TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------------------------



**BÁO CÁO** **CHUYÊN ĐỀ 2**

**Triển khai mô hình deep learning cho bài toán sentiment analysis**

**trên tập dữ liệu đánh giá sản phẩm của Amazon**

**Sinh viên thực hiện:** **LỤC TẤN ANH**

**Lớp/Khóa: CNTT K19K**

**Giảng viên hướng dẫn: ThS.HÀ THỊ THANH**

**Thái Nguyên – 2024**

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc193958852)

[**MỤC LỤC ẢNH** 4](#_Toc193958853)

[**GIỚI THIỆU** 5](#_Toc193958854)

[**CHƯƠNG 1.PHÁT BIỂU BÀI TOÁN** 6](#_Toc193958855)

[**1.1.Đọc và phân tích bài toán** 6](#_Toc193958856)

[**1.2.Phân tích bài toán và tập dữ liệu** 6](#_Toc193958857)

[**1.3.Phương pháp tiếp cận và xây dựng mô hình** 7](#_Toc193958858)

[**1.3.1. Tiền xử lý dữ liệu** 7](#_Toc193958859)

[**1.3.2.Xây dựng mô hình** 7](#_Toc193958860)

[**1.3.3. Huấn luyện và tối ưu hóa** 8](#_Toc193958861)

[**1.4.Huấn luyện, đánh giá và phân tích kết quả** 8](#_Toc193958862)

[**1.5. Thách thức và giải pháp ứng dụng** 8](#_Toc193958863)

[**CHƯƠNG 2: MÔ HÌNH TRIỂN KHAI** 10](#_Toc193958864)

[**2.1.Thu thập dữ liệu và phân tích dữ liệu** 10](#_Toc193958865)

[**2.1.1.Tổng quan về tập dữ liệu** 10](#_Toc193958866)

[**2.1.2.Thu thập dữ liệu** 10](#_Toc193958867)

[**2.1.3.Phân tích dữ liệu** 10](#_Toc193958868)

[**2.1.4.Thống kê dữ liệu** 11](#_Toc193958869)

[**2.1.5.Phương pháp sử dụng** 12](#_Toc193958870)

[**2.2.Tiền xử lý dữ liệu** 13](#_Toc193958871)

[**2.2.1. Đọc dữ liệu từ CSV** 14](#_Toc193958872)

[**2.2.2.Làm sạch dữ liệu** 15](#_Toc193958873)

[**2.2.3.Xử lý dữ liệu từ file CSV** 15](#_Toc193958874)

[**2.2.4.Tách từ (Tokenization)** 16](#_Toc193958875)

[**2.2.5.Tạo Bigram để Nâng cao Biểu diễn Ngữ nghĩa** 17](#_Toc193958876)

[**2.2.6.Chuyển đổi Văn bản thành Dữ liệu Số và Chuẩn hóa Độ dài** 18](#_Toc193958877)

[**2.2.7.Cân bằng Dữ liệu bằng SMOTE** 19](#_Toc193958878)

[**2.3. Nghiên cứu và lựa chọn mô hình deep learning** 19](#_Toc193958879)

[**2.3.1.Cơ sở lý thuyết** 20](#_Toc193958880)

[**2.3.2.Lựa chọn mô hình BiLSTM** 21](#_Toc193958881)

[**2.3.3.Lựa chọn vector Word2Vec** 21](#_Toc193958882)

[**2.3.4.Quy trình triển khai mô hình BiLSTM** 22](#_Toc193958883)

[**2.4. Xây dựng và cài đặt mô hình** 22](#_Toc193958884)

[**2.4.1.Lựa chọn framework** 22](#_Toc193958885)

[**2.4.2.Xây dựng chương trình** 24](#_Toc193958886)

[**CHƯƠNG 3: KẾT QUẢ** 31](#_Toc193958887)

[**3.1.Trình bài các độ đo áp dụng trong bài toán** 31](#_Toc193958888)

[**3.1.1.Accuracy** 31](#_Toc193958889)

[**3.1.2.Precision** 31](#_Toc193958890)

[**3.1.3.Recall** 32](#_Toc193958891)

[**3.1.4.F1-Score** 33](#_Toc193958892)

[**3.1.5.Support** 33](#_Toc193958893)

[**3.1.6.Macro Average** 33](#_Toc193958894)

[**3.1.7.Weighted Average** 34](#_Toc193958895)

[**3.2.Đánh giá mô hình trên tập test và điều chỉnh cải thiện mô hình** 35](#_Toc193958896)

[**3.2.1.Đánh giá mô hình** 35](#_Toc193958897)

[**3.2.2. Điều chỉnh mô hình cải thiện hiệu năng của mô hình** 35](#_Toc193958898)

[**Kết luận** 39](#_Toc193958899)

[**Tài liệu tham khảo** 40](#_Toc193958900)

# **MỤC LỤC ẢNH**

[Hình 1: Mình họa bài toán sentiment analysis 6](#_Toc193960120)

[Hình 2: Hình mình họa mô hình BiLSTM 21](#_Toc193960121)

[Hình 3: Hình minh họa vector Word2Vec 22](#_Toc193960122)

[Hình 4 : Công thức tính Accuracy 31](#_Toc193960123)

[Hình 5 : Công thức tính Precision 32](#_Toc193960124)

[Hình 6 : Công thức tính Recall 33](#_Toc193960125)

[Hình 7 : Công thức tính F1-Score 33](#_Toc193960126)

[Hình 8 : Công thức tính Macro Average 34](#_Toc193960127)

[Hình 9 : Công thức tính Weighted Average 34](#_Toc193960128)

[Hình 10: kết quả đánh giá mô hình trước điều chỉnh 35](#_Toc193960129)

[Hình 11: kết quả đánh giá mô hình sau điều chỉnh 36](#_Toc193960130)

[Hình 12: Hình ảnh trả về kết quả là tích cực 37](#_Toc193960131)

[Hình 13: Hình ảnh trả về kết quả là tiêu cực 37](#_Toc193960132)

[Hình 14: Hình ảnh trả về kết quả trả về với mỗi review lấy từ amazon 38](#_Toc193960133)

# **GIỚI THIỆU**

Báo cáo tốt nghiệp này hướng đến việc xây dựng và triển khai một hệ thống phân tích cảm xúc dựa trên sự kết hợp giữa mô hình BiLSTM và kỹ thuật biểu diễn từ Word2Vec. Mục tiêu chính của nghiên cứu là phát triển một giải pháp có thể nhận diện và phân loại cảm xúc từ các đánh giá sản phẩm, đặc biệt trong bối cảnh dữ liệu thực tế luôn có tính đa dạng và phức tạp. Để đạt được điều này, quá trình nghiên cứu được chia thành các giai đoạn: thu thập và xử lý dữ liệu, xây dựng mô hình, tinh chỉnh tham số, và cuối cùng là đánh giá hiệu suất của mô hình.

Việc ứng dụng mô hình BiLSTM cho phép hệ thống nắm bắt được thông tin ngữ cảnh của văn bản theo cả hai chiều, đồng thời giảm thiểu những hạn chế của các mô hình xử lý tuần tự truyền thống. Kết hợp với kỹ thuật Word2Vec, các từ được chuyển hóa thành các vector số học, giúp làm rõ mối quan hệ ngữ nghĩa giữa các từ, từ đó cải thiện độ chính xác trong việc phân loại các cảm xúc. Điều này cho thấy tiềm năng to lớn của việc ứng dụng học sâu trong xử lý ngôn ngữ tự nhiên.

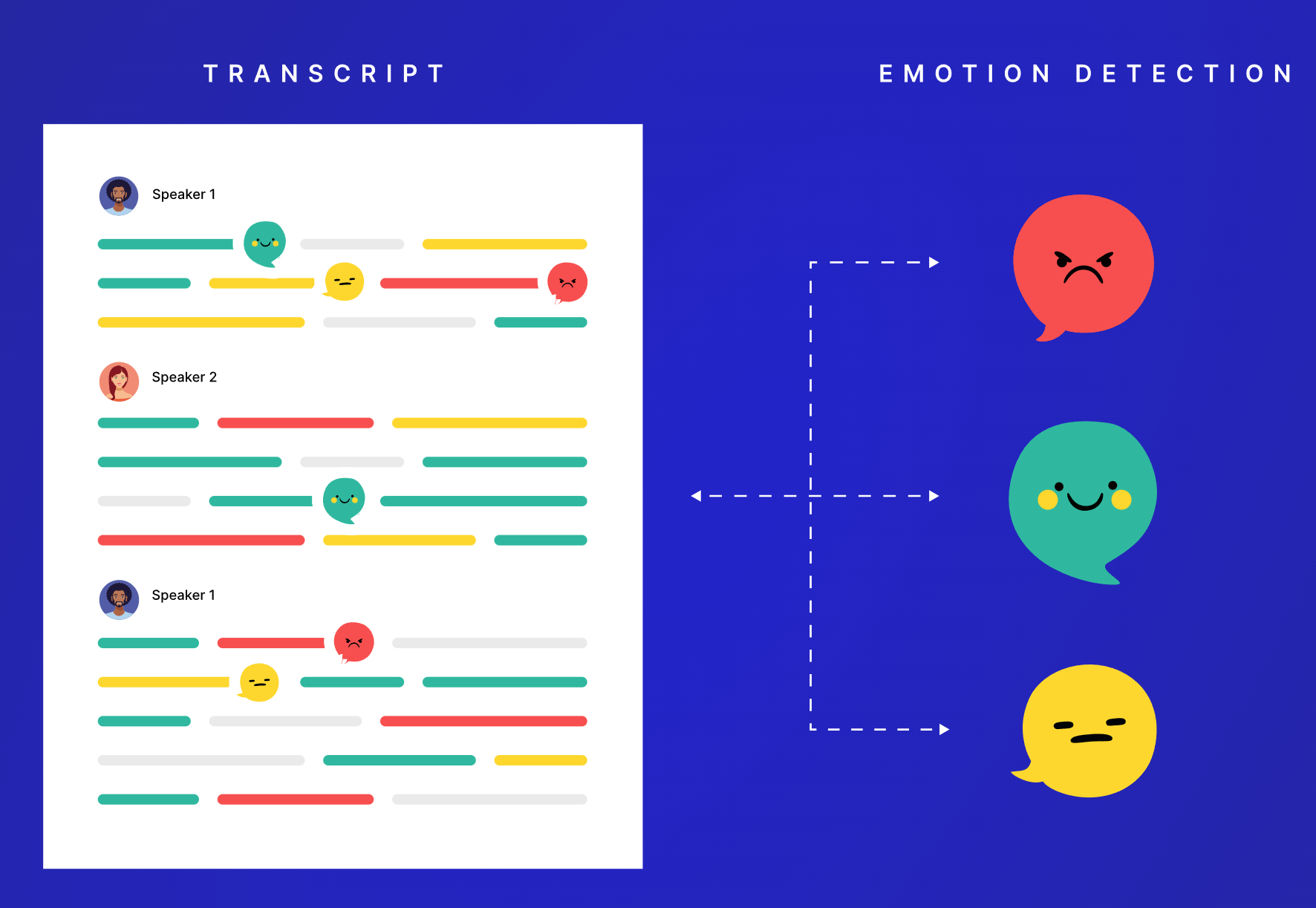
Thông qua báo cáo này, em mong muốn đóng góp một giải pháp hiệu quả và ứng dụng được trong thực tiễn, đồng thời cung cấp một cái nhìn toàn diện về những thách thức cũng như cơ hội trong lĩnh vực phân tích cảm xúc. Kết quả nghiên cứu không chỉ mang lại giá trị về mặt lý thuyết mà còn hướng đến việc hỗ trợ các doanh nghiệp tối ưu hóa quy trình thu thập và xử lý thông tin khách hàng, từ đó đưa ra những quyết định chiến lược đúng đắn.

Xin chân thành cảm ơn!

**CHƯƠNG 1.PHÁT BIỂU BÀI TOÁN**

## **1.1.Đọc và phân tích bài toán**

Trong bối cảnh thị trường thương mại điện tử ngày càng cạnh tranh, việc khai thác dữ liệu phản hồi của khách hàng trở nên vô cùng quan trọng. Đặc biệt, nhận xét đánh giá sản phẩm từ Amazon là nguồn tư liệu quý giá giúp doanh nghiệp hiểu rõ cảm nhận và kỳ vọng của người tiêu dùng. Trong báo cáo này, trình bày quá trình nghiên cứu và triển khai mô hình deep learning để phân tích cảm xúc từ các bài đánh giá, nhằm phân loại chúng thành các nhóm như tích cực, tiêu cực và trung lập. Mục đích của dự án không chỉ là tăng cường độ chính xác trong việc phân loại mà còn tìm ra các hướng tiếp cận mới nhằm tối ưu hóa hiệu quả khi ứng dụng vào thực tiễn.



Hình 1: Mình họa bài toán sentiment analysis

## **1.2.Phân tích bài toán và tập dữ liệu**

Tập dữ liệu được đánh nhãn để phân biệt cảm xúc của bài viết, các bài viết chứa thông tin chi tiết như nhãn, nội dung và thời gian đăng bài. Một số điểm đáng chú ý khi xem xét dữ liệu này là:

* Đa dạng nội dung:

Mỗi bài đánh giá có thể khác nhau về nội dung bài viết. Có những bài ngắn gọn, súc tích và có những bài dài chứa đầy cảm xúc, từ lời khen ngợi đến những lời chỉ trích chân thật. Điều này đòi hỏi các bước xử lý dữ liệu phải linh hoạt, đảm bảo không làm mất đi ý nghĩa gốc của từng bài viết.

* Tính mất cân bằng:

Nhận xét tích cực thường chiếm ưu thế so với tiêu cực. Tình trạng này không những ảnh hưởng đến quá trình huấn luyện mà còn gây khó khăn trong việc đánh giá độ tin cậy của mô hình khi áp dụng vào dữ liệu thực tế.

* Sự phong phú của ngôn ngữ:

Người dùng thể hiện cảm xúc trong bài viết qua nhiều cách khác nhau, từ việc sử dụng từ ngữ bình thường cho tới các biểu tượng cảm xúc, thậm chí là cách viết theo phong cách riêng. Do đó, hệ thống cần phải nắm bắt được những sắc thái tinh vi trong ngôn từ để phân loại đúng đắn.

## **1.3.Phương pháp tiếp cận và xây dựng mô hình**

Để giải quyết bài toán phân tích cảm xúc, quy trình nghiên cứu được chia thành các giai đoạn rõ ràng, từ xử lý dữ liệu đến huấn luyện và tối ưu mô hình:

### **1.3.1. Tiền xử lý dữ liệu**

Trước tiên, toàn bộ dữ liệu cần được xử lý để đưa vào định dạng mà mô hình có thể hiểu được. Các công việc chính bao gồm:

* Làm sạch dữ liệu: Loại bỏ ký tự đặc biệt, chuẩn hóa chữ viết (chuyển về chữ thường) và xử lý các từ ngữ không cần thiết.
* Tokenization và chuẩn hóa: Phân chia câu thành các đơn vị từ (token) và áp dụng các phương pháp như stemming hay lemmatization để chuẩn hóa các từ về dạng cơ bản.
* Chuyển đổi dữ liệu thành vector: Sử dụng các kỹ thuật như Word2Vec, GloVe,TF-IDF hoặc các mô hình hiện đại như BERT để chuyển đổi văn bản thành dạng số, từ đó thể hiện mối liên hệ ngữ nghĩa giữa các từ.

### **1.3.2.Xây dựng mô hình**

Với dữ liệu đã qua xử lý, các kiến trúc deep learning sau đây được cân nhắc triển khai:

* Mạng CNN: Có khả năng nhận diện các mẫu từ ở cấp độ cục bộ, giúp phát hiện các cụm từ đặc trưng mang tính cảm xúc.
* Mạng RNN và LSTM/GRU: Giúp nắm bắt sự liên tục của chuỗi từ, từ đó xử lý tốt các ngữ cảnh dài và phức tạp.
* Mô hình Transformer: Áp dụng cơ chế self-attention để hiểu được mối liên hệ giữa tất cả các từ trong câu, qua đó cho kết quả phân loại tinh vi hơn.

### **1.3.3. Huấn luyện và tối ưu hóa**

Quá trình huấn luyện mô hình được triển khai theo từng giai đoạn:

* Phân chia dữ liệu: Dữ liệu được chia làm tập huấn luyện, tập kiểm tra và tập đánh giá nhằm đảm bảo mô hình có khả năng tổng quát hóa cao.
* Chọn hàm loss và thuật toán tối ưu: Thông thường, Cross-entropy được sử dụng kết hợp với thuật toán tối ưu như Adam để cập nhật các trọng số của mô hình.
* Giám sát quá trình huấn luyện: Các chỉ số như độ chính xác, recall, precision và F1-score được theo dõi cẩn thận. Cùng với đó, các kỹ thuật như dropout hay early stopping được áp dụng để hạn chế hiện tượng overfitting.

## **1.4.Huấn luyện, đánh giá và phân tích kết quả**

Sau khi cấu hình xong mô hình, quá trình huấn luyện diễn ra qua nhiều epoch. Trong mỗi vòng lặp, mô hình học dần cách phân biệt các đặc trưng cảm xúc từ dữ liệu:

* Quá trình huấn luyện: Sự thay đổi trong trọng số được điều chỉnh thông qua quá trình lan truyền ngược. Mỗi epoch góp phần cải thiện khả năng nhận diện của mô hình qua từng chi tiết nhỏ.
* Đánh giá mô hình: Tập dữ liệu kiểm tra giúp đo lường hiệu quả của mô hình trên dữ liệu chưa từng gặp. Sự khác biệt giữa các chỉ số trên tập huấn luyện và tập kiểm tra là thước đo để đánh giá tính khái quát hóa.
* Phân tích sai số: Các trường hợp dự đoán không chính xác được xem xét kỹ lưỡng để rút ra bài học, từ đó cải thiện quá trình xử lý dữ liệu hoặc tinh chỉnh kiến trúc mô hình.

## **1.5. Thách thức và giải pháp ứng dụng**

Trong quá trình triển khai, một số vấn đề nổi bật được phát hiện:

* Mất cân bằng dữ liệu: Tình trạng này khiến mô hình có xu hướng thiên về các lớp chiếm đa số. Giải pháp có thể là áp dụng các phương pháp cân bằng như oversampling hoặc điều chỉnh hàm loss cho từng lớp.
* Đa dạng biểu đạt: Sự phong phú về ngôn ngữ và cách diễn đạt của người dùng tạo ra nhiều biến số khó kiểm soát. Việc sử dụng từ điển cảm xúc và kết hợp các kỹ thuật tiền xử lý chuyên sâu là cần thiết.
* Yêu cầu về tính toán: Đặc biệt với các mô hình transformer, việc huấn luyện đòi hỏi tài nguyên tính toán cao. Do đó, cân nhắc giữa hiệu năng mô hình và khả năng triển khai thực tế là yếu tố then chốt.
* Tối ưu hóa tham số: Việc lựa chọn learning rate, kích thước batch hay số lớp ẩn đòi hỏi nhiều thử nghiệm, có thể sử dụng các kỹ thuật tìm kiếm tự động như grid search hoặc Bayesian optimization để đạt được hiệu quả tốt nhất.

# **CHƯƠNG 2: MÔ HÌNH TRIỂN KHAI**

## **2.1.Thu thập dữ liệu và phân tích dữ liệu**

### **2.1.1.Tổng quan về tập dữ liệu**

Bộ dữ liệu này được tạo ra từ các tweet thu thập qua Twitter API với tổng số khoảng 1.600.000 tweet. Mục đích chính của bộ dữ liệu là phục vụ cho việc nghiên cứu phân loại cảm xúc, tức là tự động xác định xem một tweet mang cảm xúc tích cực hay tiêu cực. Một điểm đặc biệt của dataset này là nguồn gốc của các nhãn (label) được xác định theo phương pháp “distant supervision”. Cụ thể:

* Các tweet có chứa biểu tượng cảm xúc tích cực (ví dụ: “:)”) được gán nhãn là tích cực (positive – giá trị 4).
* Các tweet có chứa biểu tượng cảm xúc tiêu cực (ví dụ: “:(”) được gán nhãn là tiêu cực (negative – giá trị 0).

Lưu ý: rằng mặc dù có định nghĩa cho nhãn trung lập (neutral – giá trị 2), trong thực tế tập dữ liệu này chỉ bao gồm các tweet mang nhãn 0 và 4.

### **2.1.2.Thu thập dữ liệu**

Đây là bộ dữ liệu có sẵn trên Kaggle, Nó chứa 1.600.000 tweet được trích xuất bằng api twitter. Các tweet đã được chú thích (0 = tiêu cực, 4 = tích cực) và chúng có thể được sử dụng để phát hiện cảm xúc.

Đường dẫn: https://www.kaggle.com/datasets/kazanova/sentiment140

### **2.1.3.Phân tích dữ liệu**

Mỗi dòng dữ liệu (tương ứng với một tweet) chứa 6 trường thông tin, giúp cung cấp cái nhìn tổng thể về nguồn gốc cũng như nội dung của tweet đó. Cụ thể:

* target (Nhãn Cảm Xúc):

Trường này biểu diễn cực tính của tweet, với giá trị 0 cho cảm xúc tiêu cực và 4 cho cảm xúc tích cực. Nó được xây dựng dựa trên sự hiện diện của các biểu tượng cảm xúc trong tweet.

* 4: tweet mang cảm xúc tích cực.
* 0: tweet mang cảm xúc tiêu cực.

(Lưu ý: mặc dù có quy định cho giá trị 2 – trung lập, nhưng bộ dữ liệu này không chứa nhãn trung lập.)

* ids (ID Tweet):

Đây là một mã số duy nhất cho mỗi tweet trên Twitter. Sự duy nhất của ID giúp dễ dàng nhận dạng và truy xuất tweet từ nguồn gốc ban đầu trên Twitter.

* date (Ngày Tháng):

Trường này ghi lại thời điểm tweet được đăng tải. Thông tin về ngày tháng không chỉ giúp theo dõi xu hướng cảm xúc theo thời gian mà còn hỗ trợ việc phân tích các sự kiện theo chu kỳ (ví dụ: so sánh cảm xúc trong các dịp lễ, sự kiện chính trị, hay khủng hoảng xã hội).

* flag (Truy Vấn):

Trường này lưu trữ thông tin về truy vấn tìm kiếm được sử dụng khi lấy tweet. Trong trường hợp bộ dữ liệu này, nếu không có truy vấn cụ thể, giá trị mặc định sẽ là “NO\_QUERY”. Điều này cho thấy rằng các tweet được thu thập không dựa trên một từ khóa hay cụm từ tìm kiếm cụ thể nào, mà chủ yếu là thông qua các biểu tượng cảm xúc.

* user (User):

Đây là tên người dùng (username) đã đăng tweet. Thông tin này hữu ích để xác định nguồn gốc của tweet, từ đó có thể phân tích mức độ đóng góp của từng người dùng hoặc nhóm người dùng vào tổng thể dữ liệu.

* text (Nội Dung Tweet):

Đây là nội dung chính của tweet, tức là văn bản mà người dùng đăng tải. Mặc dù tweet có giới hạn ký tự, nhưng nội dung này chứa đựng những thông tin cảm xúc chính yếu và là đối tượng chính để thực hiện các bước xử lý ngôn ngữ tự nhiên và phân loại cảm xúc.

### **2.1.4.Thống kê dữ liệu**

Tổng dữ liệu có chứa gần 1.600.000 tweets, một số thống kê sơ bộ:

* target:
* Dataset chỉ có hai nhãn chính: 0 (negative) và 4 (positive).
* Mặc dù hệ thống ban đầu có thể định nghĩa thêm nhãn trung lập (2), nhưng do cách thu thập dựa trên biểu tượng cảm xúc, các tweet trung lập hầu như không có trong dataset.
* ids:
* Các tweet có ID khác nhau trong khoảng từ 1.47x.xxx.xxx đến 2.33x.xxx.xxx. Điều này cho thấy sự phong phú và liên tục trong quá trình thu thập dữ liệu từ Twitter.
* date:
* Số lượng khoảng 774.362 giá trị khác nhau trong trường date chứng tỏ rằng dữ liệu được thu thập qua nhiều thời điểm khác nhau.
* Điều này hữu ích trong việc phân tích xu hướng theo thời gian, giúp xác định mốc thời gian có sự biến đổi cảm xúc đáng chú ý.
* flag:
* Với duy nhất một giá trị “NO\_QUERY”, điều này xác nhận rằng không có truy vấn cụ thể nào khác được sử dụng ngoài tiêu chí biểu tượng cảm xúc, tạo nên một tập dữ liệu thuần túy.
* user:
* có 659.775 người dùng khác nhau cho thấy dữ liệu không chỉ đến từ một vài cá nhân mà thực sự đại diện cho một cộng đồng đa dạng.
* Điều này giúp giảm thiểu hiện tượng thiên lệch do quá nhiều tweet đến từ một số người dùng nhất định.
* text:
* Với 1.581.465 giá trị duy nhất, trường text phản ánh tính đa dạng về nội dung.
* Sự khác biệt trong các tweet đảm bảo rằng mô hình phân loại cảm xúc sẽ phải xử lý được nhiều kiểu ngôn ngữ và phong cách viết khác nhau, từ đó nâng cao khả năng tổng quát hóa khi áp dụng vào các tập dữ liệu khác.

### **2.1.5.Phương pháp sử dụng**

Để xây dựng bộ dữ liệu này, hai phương pháp chính đã được sử dụng:

* Distant Supervision:
* Phương pháp này tận dụng các biểu tượng cảm xúc hiện có trong tweet để tự động gán nhãn cho dữ liệu.
* Cách thực hiện:

+ Tweet có chứa biểu tượng “:)” sẽ được gán nhãn là positive sentiment (giá trị 4).

+ Tweet có chứa biểu tượng “:(” sẽ được gán nhãn là negative sentiment (giá trị 0).

* Ưu điểm của phương pháp là tiết kiệm thời gian và nguồn lực so với việc gán nhãn thủ công. Tuy nhiên, nó cũng có hạn chế, vì không phải lúc nào sự hiện diện của một biểu tượng cảm xúc cũng phản ánh đúng cảm xúc tổng thể của tweet.
* Twitter Search API:
* Dữ liệu được thu thập thông qua Twitter Search API, bằng cách sử dụng các từ khóa liên quan đến biểu tượng cảm xúc.
* Phương pháp này giúp truy xuất các tweet chứa các biểu tượng cảm xúc cần thiết, đảm bảo rằng dữ liệu thu được phù hợp với mục tiêu phân loại cảm xúc.
* Quá trình lấy dữ liệu qua API cũng giúp đảm bảo tính thời gian thực và cập nhật liên tục, mặc dù cách thu thập chủ yếu dựa vào biểu tượng cảm xúc cũng có thể dẫn đến việc bỏ sót những tweet không sử dụng biểu tượng nhưng vẫn mang thông điệp cảm xúc rõ ràng.

## **2.2.Tiền xử lý dữ liệu**

Tiền xử lý dữ liệu văn bản giúp loại bỏ những yếu tố không hữu ích trong dữ liệu và chuẩn hóa chúng để mô hình có thể xử lý tốt hơn. Các bước chính bao gồm:

* Loại bỏ URL và ký tự đặc biệt: Các URL hoặc ký tự đặc biệt không chứa thông tin cảm xúc, cần phải loại bỏ.
* Chuyển đổi văn bản thành chữ thường: Giúp loại bỏ sự phân biệt giữa chữ hoa và chữ thường.
* Tokenization: Chia văn bản thành các đơn vị nhỏ (từ hoặc cụm từ).
* Loại bỏ Stopwords: Loại bỏ các từ không mang ý nghĩa cụ thể, ví dụ như "the", "is", "at".
* Lemmatization: Là quá trình giảm từ về dạng gốc của nó, giúp giảm thiểu sự phức tạp trong dữ liệu.

### **2.2.1. Đọc dữ liệu từ CSV**

Dữ liệu tweet được lấy từ file CSV, trong đó mỗi dòng chứa các thông tin như cảm xúc (sentiment), ID, ngày, người dùng, và nội dung tweet. Hàm load\_tweets\_from\_csv() được sử dụng để đọc dữ liệu từ file CSV, chuẩn hóa nhãn cảm xúc và trích xuất các tweet cần thiết.

Hàm này thực hiện các bước sau:

* Đọc dữ liệu từ file CSV với định dạng được chỉ định và gán tên cột cho dữ liệu.
* Chuyển nhãn cảm xúc từ giá trị 0 thành -1 (tiêu cực) và từ 4 thành 1 (tích cực).
* Trích xuất các tweet và nhãn cảm xúc vào danh sách các đối tượng tweet.

Code trích dẫn:

**def load\_tweets\_from\_csv(file\_path):**

**try:**

**df = pd.read\_csv(file\_path, encoding='latin1', header=None)**

**df.columns = ["sentiment", "id", "date", "query", "user", "text"]**

**df["sentiment"] = df["sentiment"].replace({0: -1, 4: 1})**

**tweets = []**

**for index, row in df.iterrows():**

**parsed\_tweet = {"text": row["text"], "sentiment": row["sentiment"]}**

**tweets.append(parsed\_tweet)**

**return tweets**

**except Exception as e:**

**print(f"Lỗi khi đọc file CSV: {e}")**

**return None**

Giải thích chi tiết:

* pd.read\_csv(file\_path, encoding='latin1', header=None) đọc dữ liệu từ file CSV. Tham số header=None chỉ ra rằng file không có dòng tiêu đề, và encoding='latin1' giúp xử lý các ký tự đặc biệt trong dữ liệu.
* df.columns gán tên cho các cột của DataFrame.
* df["sentiment"].replace({0: -1, 4: 1}) chuyển đổi các nhãn cảm xúc từ 0 (tiêu cực) thành -1 và từ 4 (tích cực) thành 1.

### **2.2.2.Làm sạch dữ liệu**

Một trong những nhiệm vụ quan trọng trong tiền xử lý văn bản là làm sạch dữ liệu, loại bỏ các thành phần không cần thiết như tài khoản người dùng, liên kết URL, và các ký tự đặc biệt. Điều này giúp giảm bớt nhiễu và giữ lại các từ quan trọng cho phân tích.

Trong hệ thống, quá trình này được thực hiện thông qua phương thức clean\_tweet(). Cụ thể, phương thức này sử dụng biểu thức chính quy (regular expressions) để thay thế các thành phần không mong muốn trong văn bản. Cụ thể:

* Các tài khoản người dùng (dấu hiệu bắt đầu bằng @ và theo sau là các ký tự alphanumeric) sẽ được thay thế bằng một khoảng trắng.
* Các liên kết URL (dấu hiệu bắt đầu bằng http:// hoặc https://) cũng sẽ bị loại bỏ.
* Các ký tự đặc biệt khác, không phải là chữ cái, số hoặc khoảng trắng, cũng sẽ bị loại bỏ.

code của phương thức clean\_tweet() như sau:

**def clean\_tweet(self, tweet):**

**return ' '.join(re.sub(r"(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", tweet).split())**

Giải thích:

* Biểu thức chính quy (@[A-Za-z0-9]+) tìm kiếm tất cả các tài khoản người dùng bắt đầu bằng @ và theo sau là các ký tự alphanumeric.
* Biểu thức ([^0-9A-Za-z \t]) loại bỏ tất cả các ký tự không phải là chữ cái, số hoặc khoảng trắng.
* Biểu thức (\w+:\/\/\S+) tìm kiếm các liên kết URL bắt đầu bằng http:// hoặc https://.

### **2.2.3.Xử lý dữ liệu từ file CSV**

Sau khi làm sạch dữ liệu, bước tiếp theo là tiền xử lý các văn bản bằng cách chuẩn hóa và chuẩn bị dữ liệu để mô hình học máy có thể hiểu được. Các thao tác chính trong bước này bao gồm:

* Chuyển về chữ thường: Giảm thiểu sự phân biệt giữa chữ hoa và chữ thường để mô hình có thể làm việc với các từ một cách đồng nhất.
* Loại bỏ stopwords: Stopwords là những từ không mang nhiều ý nghĩa trong ngữ cảnh phân tích cảm xúc, như các từ nối, mạo từ (ví dụ: "the", "a", "and"). Việc loại bỏ các stopwords giúp giảm kích thước dữ liệu và tập trung vào các từ khóa có ý nghĩa hơn.
* Tách từ (Tokenization): Tách một câu thành các từ riêng biệt, giúp mô hình có thể phân tích chi tiết hơn các đơn vị ngữ nghĩa.
* Lemmatization: Làm giảm các từ về dạng cơ bản của chúng (lemma). Ví dụ: từ "running" sẽ được chuyển thành "run", giúp giảm thiểu sự đa dạng của từ vựng.

Các thao tác này được thực hiện trong hàm preprocess\_text():

**def preprocess\_text(text, use\_lemmatization=True):**

**text = text.lower() # Chuyển về chữ thường**

**stop\_words = set(stopwords.words('english'))**

**words = word\_tokenize(text)**

**words = [word for word in words if word not in stop\_words]**

**if use\_lemmatization:**

**lemmatizer = WordNetLemmatizer()**

**words = [lemmatizer.lemmatize(word) for word in words]**

**return ' '.join(words)**

Giải thích:

* text.lower() chuyển tất cả các ký tự trong văn bản thành chữ thường.
* word\_tokenize(text) sử dụng thư viện nltk để tách câu thành các từ riêng biệt.
* stopwords.words('english') lấy danh sách các từ dừng (stopwords) từ thư viện nltk.
* lemmatizer.lemmatize(word) sử dụng thuật toán lemmatization của thư viện nltk để giảm từ về dạng cơ bản.

### **2.2.4.Tách từ (Tokenization)**

Mục đích của tách từ là biến văn bản thành các thành phần có thể xử lý được, tạo ra các từ hoặc cụm từ (tokens) mà mô hình có thể sử dụng để học và phân tích. Tách từ giúp mô hình nhận diện được từng từ và quan hệ giữa các từ trong một câu. Đặc biệt trong ngữ cảnh phân tích cảm xúc, từng từ (hoặc cụm từ) có thể mang ý nghĩa riêng biệt và có thể quyết định cảm xúc của câu.

sử dụng hàm word\_tokenize() từ thư viện nltk để tách từ:

**from nltk.tokenize import word\_tokenize**

**words = word\_tokenize(text)**

word\_tokenize(text): Hàm này nhận vào một văn bản (dạng chuỗi) và tách nó thành một danh sách các từ. Đây là một bước rất quan trọng vì mô hình học máy cần phải làm việc với các từ riêng biệt trong văn bản, không phải toàn bộ câu

Ví dụ, ta có câu sau trong dữ liệu tweet:

"Love the new iPhone! #Amazing"

Khi áp dụng phương pháp “word\_tokenize”, kết quả sẽ là:

['Love', 'the', 'new', 'iPhone', '!', '#', 'Amazing']

Giải thích:

* Từ đơn lẻ: Mỗi từ trong câu đã được tách riêng biệt.

“ Love, the, new, iPhone, Amazing. ”

* Dấu câu và hashtag: Dấu câu như “ ! ” và các ký hiệu như “ # “ cũng được tách riêng thành các tokens riêng biệt

### **2.2.5.Tạo Bigram để Nâng cao Biểu diễn Ngữ nghĩa**

Sau khi làm sạch và token hóa, các câu được chuyển thành danh sách từ. Một bước quan trọng khác là tạo các cặp từ (bigram) nhằm nhận diện các cụm từ mang ý nghĩa riêng biệt. Ví dụ, cụm từ “not good” mang một nghĩa hoàn toàn khác so với từ “not” và “good” nếu tách riêng.

Code trích dẫn:

**# Tokenize từng câu thành danh sách các từ**

**tokenized\_sentences = [text.split() for text in processed\_texts]**

**# Tạo bigrams: kết hợp các từ đơn và cặp từ (bigram)**

**phrases = Phrases(tokenized\_sentences, min\_count=1, threshold=2) # threshold thấp tạo nhiều bigram**

**bigram = Phraser(phrases)**

**tokenized\_sentences\_with\_bigrams = [bigram[sentence] for sentence in tokenized\_sentences]**

Giải thích:

* Mỗi câu trong processed\_texts (danh sách các câu đã được xử lý trước) được tách thành danh sách các từ bằng phương thức .split().
* Sử dụng Phrases từ thư viện gensim để phát hiện và tạo bigram (cặp từ thường xuyên xuất hiện cùng nhau).
* min\_count=1: Chấp nhận cả các cặp từ xuất hiện chỉ một lần.
* threshold=2: Ngưỡng xác định mức độ phổ biến của cụm từ để nó được coi là bigram (ngưỡng thấp sẽ tạo nhiều bigram hơn).
* Phraser giúp giảm bộ nhớ và tăng tốc quá trình xử lý bằng cách lưu các bigram đã học được.
* Thay thế các cặp từ trong câu bằng bigram nếu có, ví dụ: "New York" thay vì ["New", "York"].

### **2.2.6.Chuyển đổi Văn bản thành Dữ liệu Số và Chuẩn hóa Độ dài**

Để đưa dữ liệu văn bản vào mô hình deep learning, cần phải chuyển đổi các câu thành các chuỗi số. Việc này được thực hiện thông qua Tokenizer của Keras, sau đó sử dụng pad\_sequences để chuẩn hóa độ dài của mỗi chuỗi.

Code trích dấn:

**# Sử dụng Keras Tokenizer cho bước chuyển văn bản thành chuỗi số**

**tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token="<unk>")**

**tokenizer.fit\_on\_texts(processed\_texts)**

**sequences = tokenizer.texts\_to\_sequences(processed\_texts)**

**padded\_sequences = pad\_sequences(sequences, maxlen=max\_len, padding='post', truncating='post')**

Giải thích:

* num\_words=vocab\_size: Giới hạn số lượng từ trong từ điển (chỉ giữ lại vocab\_size từ phổ biến nhất).
* oov\_token="<unk>": Gán <unk> (unknown) cho các từ không có trong từ điển.
* Xây dựng từ điển từ processed\_texts (danh sách văn bản đã qua xử lý), gán mỗi từ một chỉ số số nguyên.
* Mỗi câu trong processed\_texts được chuyển thành danh sách các số nguyên tương ứng với từ trong từ điển.
* maxlen=max\_len: Độ dài tối đa của mỗi chuỗi số. padding='post': Thêm số 0 vào cuối chuỗi nếu ngắn hơn max\_len. truncating='post': Cắt bớt từ cuối nếu chuỗi dài hơn max\_len.

### **2.2.7.Cân bằng Dữ liệu bằng SMOTE**

Trong bài toán phân loại cảm xúc, số lượng mẫu của các lớp (ví dụ: cảm xúc tích cực, tiêu cực) thường không đồng đều. Để khắc phục vấn đề này, kỹ thuật SMOTE (Synthetic Minority Over-sampling Technique) được sử dụng nhằm tạo ra các mẫu tổng hợp cho lớp ít gặp, từ đó giúp mô hình học được sự phân bố dữ liệu toàn diện hơn.

Code trích dấn:

**smote = SMOTE(random\_state=2)**

**X\_train\_resampled, y\_train\_resampled = smote.fit\_resample(X\_train, y\_train)**

Giải thích:

* SMOTE là một kỹ thuật tăng cường dữ liệu (oversampling) cho tập dữ liệu mất cân bằng bằng cách tạo mẫu dữ liệu giả từ lớp thiểu số.
* random\_state=2: Đảm bảo kết quả có thể tái lập.
* fit\_resample(X\_train, y\_train): Tạo thêm dữ liệu từ lớp thiểu số và trả về tập huấn luyện đã cân bằng.
* X\_train\_resampled: Dữ liệu đặc trưng mới sau khi áp dụng SMOTE.
* y\_train\_resampled: Nhãn tương ứng với dữ liệu mới.

## **2.3. Nghiên cứu và lựa chọn mô hình deep learning**

Trong quá trình tìm hiểu và triển khai bài toán phân tích cảm xúc từ các bài đánh giá sản phẩm, việc lựa chọn mô hình deep learning phù hợp là yếu tố then chốt để đạt được hiệu quả cao. Qua việc khảo sát các nghiên cứu trước đây và đánh giá thực nghiệm, tập trung so sánh một số mô hình như CNN, RNN (bao gồm LSTM và GRU), mô hình BiLSTM và các mô hình Transformer.

### **2.3.1.Cơ sở lý thuyết**

Trong lĩnh vực xử lý ngôn ngữ tự nhiên, đặc biệt với bài toán phân tích cảm xúc, đặc điểm của dữ liệu văn bản như thứ tự từ ngữ và ngữ cảnh đóng vai trò quan trọng. Các kiến trúc deep learning thông dụng hiện nay có những ưu, nhược điểm như sau:

* Mạng Nơ-ron Tích chập (CNN):

CNN đã được ứng dụng rộng rãi trong các bài toán xử lý ảnh và văn bản nhờ khả năng trích xuất các đặc trưng cục bộ. Tuy nhiên, trong bài toán phân tích cảm xúc, việc thiếu đi khả năng ghi nhớ thứ tự từ dài hạn có thể hạn chế hiệu quả khi cần nắm bắt các quan hệ phức tạp trong câu.

* Mạng Nơ-ron Hồi tiếp (RNN) và LSTM/GRU:

Các mô hình này có khả năng xử lý chuỗi dữ liệu theo thời gian, từ đó ghi nhớ các thông tin ngữ cảnh trong câu. Đặc biệt, LSTM và GRU đã được chứng minh là có khả năng khắc phục hạn chế về gradient khi xử lý các chuỗi dài. Tuy nhiên, mô hình đơn giản thường chỉ học được thông tin theo một chiều, dẫn đến mất đi một phần ngữ cảnh quan trọng.

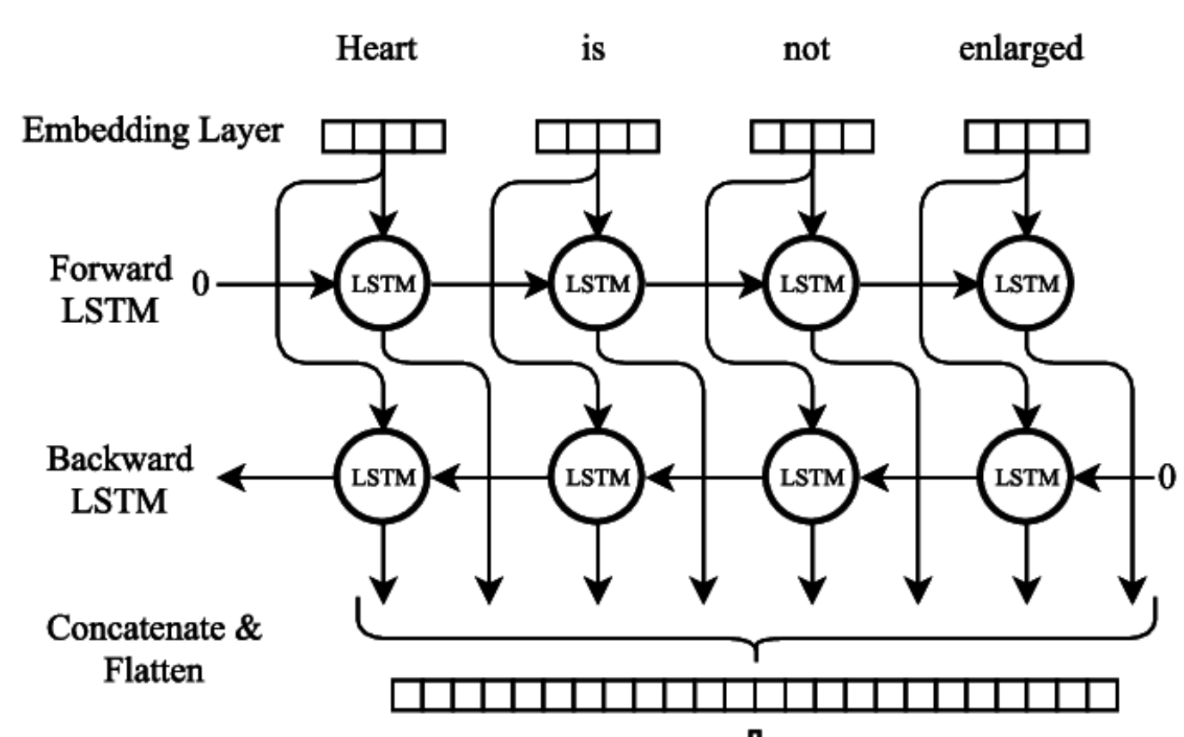
* Mô hình BiLSTM:

Mô hình BiLSTM là phiên bản mở rộng của LSTM, trong đó dữ liệu được xử lý song song theo hai chiều (tiến và lùi). Nhờ đó, mô hình có thể nắm bắt được ngữ cảnh đầy đủ hơn, không chỉ từ quá khứ mà còn từ tương lai của mỗi từ. Các nghiên cứu đã chỉ ra rằng BiLSTM thường mang lại hiệu quả vượt trội trong các bài toán phân loại văn bản, đặc biệt là khi cần hiểu rõ mối quan hệ giữa các từ trong câu.

* Transformer và các kiến trúc dựa trên cơ chế self-attention:

Các mô hình như BERT hay RoBERTa đã tạo nên bước đột phá trong lĩnh vực NLP nhờ khả năng xử lý song song và khai thác quan hệ giữa các từ một cách toàn cục. Mặc dù mang lại kết quả rất ấn tượng, nhưng chi phí tính toán và yêu cầu về tài nguyên cao có thể là rào cản khi triển khai trong các hệ thống thực tiễn với quy mô lớn.

### **2.3.2.Lựa chọn mô hình BiLSTM**



Hình 2: Hình mình họa mô hình BiLSTM

Sau quá trình nghiên cứu và so sánh, nhóm nghiên cứu đã quyết định lựa chọn mô hình BiLSTM cho bài toán phân tích cảm xúc với các lý do chính sau:

* Khả năng nắm bắt ngữ cảnh toàn diện:

BiLSTM cho phép xử lý thông tin theo hai chiều, giúp mô hình hiểu được mối liên hệ phức tạp giữa các từ trong câu. Điều này rất cần thiết khi phân tích những câu có cấu trúc ngữ pháp phức tạp hoặc chứa các biểu đạt cảm xúc gián tiếp.

* Độ ổn định và khả năng tổng quát tốt:

Các nghiên cứu đã chỉ ra rằng BiLSTM có khả năng xử lý chuỗi dữ liệu dài tốt hơn so với các mô hình RNN truyền thống. Điều này cho phép mô hình giảm thiểu hiện tượng forget information (quên thông tin) khi học từ dữ liệu có thứ tự dài.

* Hiệu quả kết hợp với phương pháp biểu diễn từ:

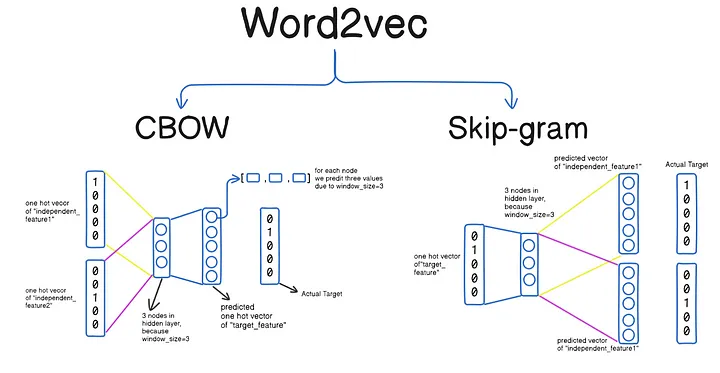
Khi kết hợp với các kỹ thuật biểu diễn từ như Word2Vec hay các embedding khác, BiLSTM có thể tận dụng được các đặc trưng ngữ nghĩa ẩn sâu của từng từ, từ đó đưa ra kết quả phân loại cảm xúc chính xác hơn.

* Sự cân bằng giữa hiệu năng và tài nguyên:

So với các mô hình Transformer, BiLSTM yêu cầu tài nguyên tính toán thấp hơn nhiều. Điều này giúp việc triển khai và huấn luyện mô hình trở nên khả thi hơn trong môi trường thực tế, nhất là với các hệ thống không có sẵn hạ tầng GPU mạnh mẽ.

### **2.3.3.Lựa chọn vector Word2Vec**

Trong quá trình xử lý dữ liệu văn bản, chuyển đổi từ ngữ thành các vector số là bước thiết yếu giúp hiểu được mối quan hệ ngữ nghĩa giữa các từ. Trong báo cáo này, lựa chọn sử dụng Word2Vec vì nó cho phép biểu diễn các từ trong không gian liên tục, theo đó các từ có ý nghĩa tương đồng sẽ có các vector gần nhau. Điều này không chỉ giúp giảm bớt số chiều của dữ liệu mà còn làm nổi bật được các mối liên hệ ngữ nghĩa một cách tự nhiên.



Hình 3: Hình minh họa vector Word2Vec

Việc áp dụng Word2Vec có một số ưu điểm nổi bật:

* Biểu diễn ngữ nghĩa mạnh mẽ: Các vector từ được huấn luyện có khả năng nắm bắt các mối liên hệ tinh tế giữa các từ, giúp mô hình deep learning dễ dàng nhận diện các đặc trưng quan trọng trong câu.
* Tăng hiệu quả huấn luyện: Khi sử dụng ma trận embedding từ Word2Vec, mô hình học được các đặc trưng ngữ nghĩa từ dữ liệu lớn mà không cần phải huấn luyện từ đầu, từ đó cải thiện tốc độ và hiệu quả của quá trình huấn luyện.
* Khả năng tổng quát cao: Các vector từ có thể chuyển đổi tốt trên nhiều bài toán NLP khác nhau, giúp hệ thống phân tích cảm xúc có thể mở rộng ứng dụng khi cần thiết.

Qua đó, Word2Vec trở thành một công cụ hữu ích trong quá trình tiền xử lý dữ liệu, góp phần tạo ra một bộ biểu diễn ngữ nghĩa chất lượng, hỗ trợ đáng kể cho quá trình xây dựng và huấn luyện mô hình deep learning cho bài toán phân tích cảm xúc.

### **2.3.4.Quy trình triển khai mô hình BiLSTM**

Trong giai đoạn triển khai, sau khi tiền xử lý dữ liệu (bao gồm làm sạch văn bản, tokenization, tạo bigram và chuyển đổi thành chuỗi số), xây dựng mô hình BiLSTM với kiến trúc gồm:

* Lớp embedding được khởi tạo từ ma trận word embedding đã được huấn luyện từ dữ liệu.
* Hai lớp LSTM chạy song song theo hai chiều (forward và backward) để thu thập thông tin ngữ cảnh đầy đủ.
* Một lớp dropout để giảm thiểu hiện tượng overfitting.
* Lớp fully-connected để chuyển đổi thông tin thu được thành dự đoán cuối cùng về nhãn cảm xúc.

Quá trình huấn luyện được thực hiện qua nhiều epoch với việc điều chỉnh các siêu tham số như kích thước embedding, số lượng đơn vị LSTM và tốc độ học. Các chỉ số đánh giá như accuracy và F1-score được theo dõi chặt chẽ để đảm bảo mô hình có khả năng tổng quát tốt trên tập dữ liệu kiểm tra.

## **2.4. Xây dựng và cài đặt mô hình**

### **2.4.1.Lựa chọn framework**

* NLTK:

Được sử dụng để xử lý các tác vụ tiền xử lý văn bản như tokenization, loại bỏ từ dừng và lemmatization. Các hàm của NLTK giúp làm sạch dữ liệu văn bản ban đầu.

* Gensim:

Thư viện này hỗ trợ huấn luyện mô hình Word2Vec, giúp chuyển đổi từ ngữ thành các vector biểu diễn với không gian liên tục. Việc này rất quan trọng để nắm bắt ngữ nghĩa giữa các từ.

* TensorFlow (Keras):

Dùng cho bước chuyển đổi văn bản thành chuỗi số và thực hiện padding (chuẩn hóa độ dài chuỗi). Điều này giúp đảm bảo dữ liệu đầu vào đồng nhất khi huấn luyện các mô hình tuần tự.

* PyTorch:

Được chọn làm framework chính để xây dựng và huấn luyện mô hình deep learning. PyTorch cho phép kiểm soát linh hoạt quá trình huấn luyện và tận dụng GPU nếu có, đặc biệt phù hợp với mô hình BiLSTM.

* Scikit-learn và Imbalanced-learn:

Hỗ trợ các công việc như chia tách tập dữ liệu, mã hóa nhãn (LabelEncoder) và cân bằng dữ liệu sử dụng SMOTE nhằm khắc phục tình trạng mất cân bằng giữa các lớp cảm xúc.

* Streamlit:

Streamlit được lựa chọn để xây dựng giao diện web cho ứng dụng phân tích cảm xúc. Streamlit giúp phát triển các ứng dụng data science một cách nhanh chóng với cú pháp đơn giản và khả năng hiển thị kết quả trực quan.

* Joblib:

Joblib được sử dụng để lưu và tải các đối tượng tiền xử lý như tokenizer, label encoder và mô hình Word2Vec. Đây là cách hiệu quả để tái sử dụng các đối tượng đã được huấn luyện mà không cần phải tái tính toán từ đầu.

* Có thể cài đặt tất cả các gói cần thiết bằng lệnh:

**pip install streamlit torch numpy tensorflow nltk joblib gensim**

### **2.4.2.Xây dựng chương trình**

#### **2.4.2.1.Cài đặt Vector Word2Vec**

Để chuyển đổi các từ thành vector số học, việc huấn luyện mô hình Word2Vec bằng Gensim. Các vector này sẽ được sử dụng làm ma trận embedding cho mô hình BiLSTM. Dưới đây là đoạn code huấn luyện Word2Vec:

**def train\_word2vec\_model(sentences, vector\_size=100, window=5, min\_count=1, workers=4):**

**model = Word2Vec(sentences, vector\_size=vector\_size, window=window, min\_count=min\_count, workers=workers)**

**return model**

Giải thích:

* Hàm train\_word2vec\_model nhận đầu vào là danh sách các câu đã được token (các câu dưới dạng danh sách từ).Các tham số:
  + vector\_size=100: kích thước của vector từ (số chiều của không gian vector).
  + window=5: kích thước cửa sổ từ, xác định phạm vi từ xung quanh từ hiện tại để học ngữ cảnh.
  + min\_count=1: số lượng tối thiểu mà một từ cần xuất hiện để được đưa vào model.
  + workers=4: số lượng luồng xử lý song song dùng cho việc huấn luyện mô hình.
* sử dụng thư viện gensim để tạo mô hình Word2Vec, huấn luyện trên dữ liệu sentences với các tham số đã chỉ định.
* Hàm trả về đối tượng mô hình Word2Vec đã được huấn luyện, cho phép sử dụng để trích xuất vector từ cho các từ trong từ điển.

#### **2.4.2.2.Xây dựng Mô hình BiLSTM**

Mô hình BiLSTM được thiết kế để khai thác thông tin ngữ cảnh từ cả hai chiều của chuỗi văn bản, giúp hiểu rõ hơn mối liên hệ giữa các từ. Trước hết, lớp embedding chuyển đổi các từ thành vector dựa trên ma trận đã được huấn luyện từ Word2Vec, giữ lại các đặc trưng ngữ nghĩa quan trọng. Lớp LSTM hai chiều sau đó xử lý chuỗi dữ liệu theo cả hướng tiến và lùi, từ đó tạo ra một vector đại diện cho toàn bộ câu. Kết quả này được chuyển qua lớp dropout để giảm hiện tượng overfitting và sau đó qua lớp fully-connected để đưa ra dự đoán cuối cùng.

Quá trình huấn luyện sử dụng hàm loss CrossEntropyLoss và thuật toán Adam, tận dụng GPU khi có để tăng tốc độ tính toán. Cấu trúc này giúp mô hình nắm bắt ngữ cảnh sâu sắc và đạt hiệu suất cao trong việc phân loại cảm xúc.

Code trích dẫn phần xây dựng mô hình:

**class BiLSTMModel(nn.Module):**

**def \_\_init\_\_(self, vocab\_size, embedding\_dim, lstm\_units, num\_classes, embedding\_matrix, max\_len):**

**super(BiLSTMModel, self).\_\_init\_\_()**

**self.embedding = nn.Embedding(vocab\_size, embedding\_dim)**

**self.embedding.weight = nn.Parameter(torch.tensor(embedding\_matrix, dtype=torch.float32))**

**self.embedding.weight.requires\_grad = False # Không cập nhật trọng số embedding**

**self.lstm = nn.LSTM(embedding\_dim, lstm\_units, num\_layers=2, batch\_first=True, bidirectional=True, dropout=0.2)**

**self.dropout = nn.Dropout(0.2)**

**self.fc = nn.Linear(lstm\_units \* 2, num\_classes)**

**def forward(self, x):**

**x = self.embedding(x)**

**output, (hn, cn) = self.lstm(x)**

**forward\_hidden = hn[-2, :, :]**

**backward\_hidden = hn[-1, :, :]**

**hidden = torch.cat((forward\_hidden, backward\_hidden), dim=1)**

**hidden = self.dropout(hidden)**

**out = self.fc(hidden)**

**return out**

Giải thích:

* Lớp embedding: Được khởi tạo với ma trận embedding từ Word2Vec. Việc khóa trọng số (requires\_grad = False) giúp giữ nguyên các thông tin ngữ nghĩa đã học được.
* Lớp LSTM: Với cấu trúc hai chiều (bidirectional), mô hình có thể nắm bắt thông tin từ quá khứ và tương lai trong chuỗi văn bản.
* Dropout và Fully Connected: Dropout được sử dụng để giảm hiện tượng overfitting, trong khi lớp fully connected chuyển đổi đầu ra của LSTM thành xác suất của các lớp cảm xúc.

#### **2.4.2.3.Huấn luyện mô hình**

Hàm train\_and\_evaluate\_bilstm trong chương trình được xây dựng để thực hiện việc huấn luyện mô hình. Chương trình được xây dựng dựa trên liên kết giữa các bước tiền xử lý, xây dựng embedding, chuyển đổi dữ liệu như sau:

* **Định nghĩa hàm và lấy dữ liệu từ**

Định nghĩa hàm: Hàm train\_and\_evaluate\_bilstm được viết để huấn luyện và đánh giá một mô hình BiLSTM cho bài toán phân loại cảm xúc từ dữ liệu tweet. Các tham số của hàm như embedding\_dim, lstm\_units, dropout\_rate, max\_len,… được truyền vào nhằm điều chỉnh cấu hình của mô hình, lưu trữ mô hình và bộ tokenizer, cũng như các tham số về batch và số epoch.

Trích xuất thông tin: Từ danh sách các tweet (mỗi tweet là một dictionary chứa các khóa như 'text' và 'sentiment'), ta tách riêng phần văn bản và nhãn cảm xúc thành 2 danh sách riêng biệt: texts chứa nội dung của tweet và sentiments chứa giá trị cảm xúc tương ứng.

* **Tiền xử lý văn bản**

Duyệt qua từng chuỗi văn bản trong texts với sự hỗ trợ của tqdm (để hiển thị tiến trình xử lý), mỗi chuỗi được đưa vào hàm preprocess\_text. Quá trình này giúp làm sạch và chuẩn hóa dữ liệu, kết quả thu được được lưu vào danh sách processed\_texts.

Mục đích của bước này là đảm bảo rằng văn bản đầu vào cho các bước xử lý sau đều ở dạng nhất quán và không có các ký tự hay từ thừa gây nhiễu.

* **Xây dựng mô hình Word2Vec có hỗ trợ bigrams**

Tokenization cơ bản: Mỗi văn bản đã qua xử lý được chuyển thành một danh sách các từ (dựa trên việc tách theo khoảng trắng bằng phương thức split() ), tạo ra danh sách tokenized\_sentences.

Phát hiện bigrams: Sử dụng công cụ Phrases từ thư viện gensim với tham số min\_count=1 (chấp nhận các từ xuất hiện ít) và threshold=2 (ngưỡng thấp giúp nhận diện nhiều cặp từ) để xác định các cặp từ có khả năng tạo thành cụm từ (bigram). Sau đó, chuyển đổi sang đối tượng Phraser nhằm tối ưu về hiệu năng. Kết quả là, danh sách tokenized\_sentences\_with\_bigrams chứa các câu mà nếu có hai từ xuất hiện liên tiếp với tần suất cao, chúng sẽ được gộp lại thành một đơn vị duy nhất.

Huấn luyện Word2Vec: Hàm train\_word2vec\_model được gọi với dữ liệu đã có cả unigrams và bigrams. Mô hình Word2Vec được huấn luyện với kích thước vector (embedding) được xác định bởi embedding\_dim.

Tạo ma trận embedding: Sau khi Word2Vec hoàn thành, ta xác định kích thước từ vựng ( vocab\_size ) và khởi tạo một ma trận embedding với kích thước (vocab\_size, embedding\_dim). Với vòng lặp, mỗi từ trong từ điển của mô hình được ánh xạ thành một vector, và vector đó được lưu vào vị trí tương ứng trong ma trận. Điều này tạo điều kiện cho bước embedding trong mô hình BiLSTM sau này.

* **Chuyển đổi văn bản thành chuỗi số thông qua Keras Tokenizer**

Khởi tạo và huấn luyện Tokenizer: Đầu tiên, khởi tạo đối tượng Tokenizer với số từ tối đa là vocab\_size và định nghĩa token đặc biệt <unk> cho từ không có trong từ điển. Sau đó, dùng fit\_on\_texts trên processed\_texts để xây dựng từ điển ánh xạ giữa từ và số nguyên.

Chuyển đổi thành chuỗi số: Sử dụng phương thức texts\_to\_sequences, mỗi câu văn được chuyển thành một dãy số theo thứ tự từ đã được gán chỉ số.

Cân bằng độ dài chuỗi: Dùng pad\_sequences để chuẩn hóa độ dài của các dãy số. Nếu chuỗi ngắn hơn max\_len, thêm số 0 ở cuối (theo tùy chọn 'post'); nếu vượt quá, cắt bớt các từ ở cuối chuỗi.

* **Mã hóa nhãn cảm xúc**

Sử dụng LabelEncoder để chuyển các nhãn cảm xúc (đã lưu trong sentiments) từ dạng chuỗi hoặc số chưa mã hóa thành dạng số nguyên từ 0 đến số lớp trừ 1.

Sau đó, kết quả được lưu vào mảng y\_encoded để thuận tiện cho việc huấn luyện mô hình.

* **Chia tập dữ liệu và xử lý mất cân bằng**

Chia dữ liệu: Dùng hàm train\_test\_split để phân chia dữ liệu đã padding (các dãy số) và nhãn đã mã hóa thành tập huấn luyện và tập kiểm tra, với tỷ lệ 80% : 20%. Việc chia này sử dụng tham số stratify để đảm bảo phân bố nhãn trong cả hai tập là tương đương.

Cân bằng dữ liệu với SMOTE: Trong trường hợp tập huấn luyện có sự mất cân bằng giữa các lớp, sử dụng kỹ thuật SMOTE (Synthetic Minority Over-sampling Technique) với random\_state=2 để tạo ra các mẫu dữ liệu mới cho lớp thiểu số, giúp cân bằng tập dữ liệu. Kết quả của bước này là X\_train\_resampled và y\_train\_resampled với phân bố nhãn đồng đều hơn.

* **Chuyển đổi dữ liệu sang dạng tensor và tạo DataLoader**

Các tập dữ liệu train và test được chuyển sang tensor của PyTorch với kiểu dữ liệu phù hợp (chẳng hạn torch.long).

Dữ liệu train được gói trong một TensorDataset và sau đó tạo ra một DataLoader với kích thước batch như đã chỉ định, đồng thời sắp xếp ngẫu nhiên các mẫu để quá trình huấn luyện hiệu quả hơn.

* **Chuẩn bị thiết bị và khởi tạo mô hình BiLSTM**

Kiểm tra xem có GPU hay không để tận dụng tối đa tài nguyên tính toán.

torch.device("cuda" if torch.cuda.is\_available() else "cpu")

Tạo mô hình BiLSTM với các tham số như: kích thước từ vựng, kích thước embedding, số đơn vị LSTM, số lớp nhãn (được xác định từ LabelEncoder), ma trận embedding đã tạo và độ dài tối đa của chuỗi. Sau đó, chuyển mô hình sang thiết bị đã chọn (GPU hoặc CPU).

* **Thiết lập hàm mất mát và bộ tối ưu hóa**

Dùng hàm mất mát CrossEntropyLoss (thích hợp cho bài toán phân loại nhiều lớp) để đo lường hiệu suất dự đoán của mô hình.

Bộ tối ưu hóa Adam được khởi tạo để cập nhật các trọng số của mô hình dựa trên gradient tính được từ hàm mất mát.

* **Quá trình huấn luyện mô hình**

Mô hình được chuyển sang chế độ huấn luyện model.train() .Vòng lặp huấn luyện được thực hiện qua từng epoch (số epoch được quy định bởi num\_epochs).

Trong mỗi epoch, DataLoader cung cấp các batch dữ liệu. Với mỗi batch, dữ liệu và nhãn được chuyển về thiết bị tính toán, sau đó:

* Các gradient được khởi tạo lại về 0.
* Mô hình xử lý batch dữ liệu và cho ra dự đoán.
* Tính toán hàm mất mát giữa dự đoán và nhãn thật.
* Thực hiện quá trình lan truyền ngược (backpropagation) và cập nhật trọng số bằng optimizer.
* Mỗi batch được theo dõi thông qua thanh tiến trình của tqdm hiển thị loss hiện tại.

Sau khi hoàn tất một epoch, tính trung bình loss trên toàn bộ dữ liệu huấn luyện và in ra màn hình.

* **Đánh giá hiệu quả mô hình trên tập test**

Chuyển mô hình sang chế độ đánh giá ( model.eval() ) để vô hiệu hóa một số tính năng như dropout.

Tạo DataLoader cho tập test nhằm đảm bảo không gặp phải vấn đề bộ nhớ trong quá trình xử lý.

Với chế độ torch.no\_grad(), dự đoán được thu thập cho từng batch trong tập test, bằng cách lấy giá trị có xác suất cao nhất từ kết quả đầu ra của mô hình.

Sử dụng LabelEncoder để chuyển đổi lại các nhãn từ dạng số về dạng ban đầu, sau đó tính toán độ chính xác (accuracy) và sinh ra báo cáo phân loại chi tiết (classification report).

* **Lưu trữ mô hình và các đối tượng liên quan**

Nếu chưa có thư mục lưu trữ, lệnh os.makedirs sẽ tạo các thư mục cần thiết.Trạng thái của mô hình (state\_dict) được lưu vào file với đường dẫn model\_path để có thể tái sử dụng hoặc triển khai sau này.

Đồng thời, các đối tượng như tokenizer, label encoder và mô hình Word2Vec được lưu trữ (bằng cách dùng hàm dump) vào file tại tokenizer\_path để phục vụ việc tiền xử lý và giải mã cho dữ liệu mới.

# **CHƯƠNG 3: KẾT QUẢ**

## **3.1.Trình bài các độ đo áp dụng trong bài toán**

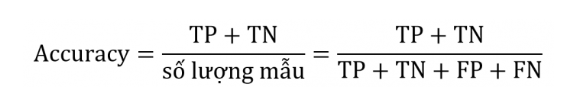
Trong quá trình xây dựng chương trình, việc đánh giá kết quả dự đoán của mô hình đóng vai trò then chốt để hiểu được khả năng nhận diện của nó. Sử dụng một số độ đo tiêu chuẩn như Accuracy, Precision, Recall, F1-Score, support, macro avg và weighted avg. Mỗi chỉ số cung cấp một góc nhìn khác nhau, từ đó giúp phát hiện và khắc phục những hạn chế của mô hình.

### **3.1.1.Accuracy**

Accuracy là tỷ lệ phần trăm mẫu mà mô hình dự đoán đúng trên tổng số mẫu. Đây là chỉ số cơ bản, giúp có cái nhìn tổng quan về khả năng của mô hình. Tuy nhiên, chỉ số này có thể gây hiểu nhầm khi dữ liệu bị mất cân bằng.

Ví dụ, nếu 90% mẫu thuộc lớp tích cực, một mô hình dự đoán tất cả đều là tích cực có thể đạt độ chính xác cao nhưng lại bỏ sót nhiều thông tin của lớp tiêu cực.

Công thức:



Hình 4 : Công thức tính Accuracy

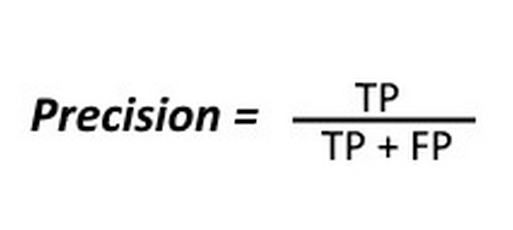
Trong đó:

* TP (True Positives): Số mẫu dương tính được dự đoán đúng.
* FP (False Positives): Số mẫu âm tính bị nhầm thành dương tính.
* FN (False Negatives): Số mẫu dương tính mà mô hình không phát hiện được.
* TN (True Negative): số lượng mẫu mà mô hình dự đoán là âm tính và thực tế cũng thuộc nhóm âm tính.

### **3.1.2.Precision**

Precision (độ chính xác dương tính) đo lường tỷ lệ các mẫu được dự đoán là dương tính mà thực sự là dương tính. Chỉ số này đặc biệt quan trọng khi việc gán nhãn sai (dự đoán dương tính giả) là cao. Nói cách khác, trong số các mẫu mà mô hình dự đoán là dương tính, bao nhiêu phần trăm là chính xác

Công thức:



Hình 5 : Công thức tính Precision

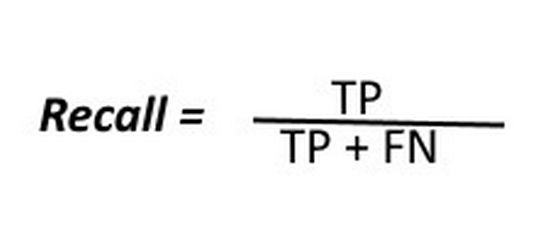
Trong đó:

* TP (True Positives): Số mẫu dương tính được dự đoán đúng.
* FP (False Positives): Số mẫu âm tính bị nhầm thành dương tính.

### **3.1.3.Recall**

Recall (độ nhạy) cho biết khả năng mô hình phát hiện đúng các mẫu dương tính trên tổng số mẫu dương tính có sẵn. Đây là chỉ số quan trọng khi việc bỏ sót mẫu dương tính (âm tính giả) có thể dẫn đến hậu quả nghiêm trọng.

Công thức:



Hình 6 : Công thức tính Recall

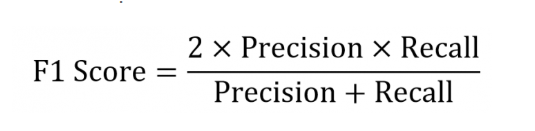
Trong đó:

* TP (True Positives): Số mẫu dương tính được dự đoán đúng.
* FN (False Negatives): Số mẫu dương tính mà mô hình không phát hiện được.

### **3.1.4.F1-Score**

F1-Score là trung bình điều hòa của Precision và Recall, cung cấp một thước đo tổng hợp về hiệu năng của mô hình. Chỉ số này rất hữu ích khi cần cân bằng giữa khả năng phát hiện mẫu dương tính và độ chính xác của các dự đoán dương tính.

Công thức:



Hình 7 : Công thức tính F1-Score

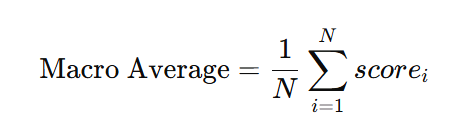
### **3.1.5.Support**

Support là số lượng mẫu thực tế thuộc mỗi lớp trong tập dữ liệu. Chỉ số này không phải là thước đo hiệu suất trực tiếp, nhưng nó rất quan trọng để hiểu được phân bố dữ liệu. Support giúp nhận biết nếu dữ liệu có sự mất cân bằng, từ đó điều chỉnh lại trọng số trong quá trình huấn luyện hoặc lựa chọn các chỉ số tổng hợp khác.

### **3.1.6.Macro Average**

Macro Average tính trung bình các giá trị của Precision, Recall và F1-Score trên tất cả các lớp mà không tính đến số lượng mẫu của từng lớp. Đây là cách đánh giá hiệu suất trung bình của mô hình trên mỗi lớp, tuy nhiên nếu dữ liệu không cân bằng, macro avg có thể không phản ánh đúng hiệu quả trên các lớp chiếm số lượng lớn.

Công thức:



Hình 8 : Công thức tính Macro Average

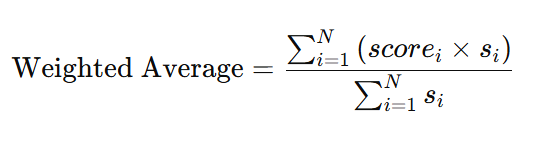
Trong đó:

* N: Số lượng các lớp trong bài toán phân loại.
* score i: Giá trị của chỉ số đánh giá (chẳng hạn như Precision, Recall hay F1-Score) đối với lớp thứ i. Mỗi lớp sẽ có một giá trị riêng cho từng chỉ số.

### **3.1.7.Weighted Average**

Weighted Average cũng tính trung bình các chỉ số nhưng trọng số của mỗi lớp được xác định bởi số mẫu (support) của lớp đó. Nhờ đó, chỉ số này thiên về các lớp có nhiều mẫu hơn, giúp tổng thể hiệu suất của mô hình được đánh giá một cách chính xác hơn khi có sự mất cân bằng dữ liệu.

Công thức:



Hình 9 : Công thức tính Weighted Average

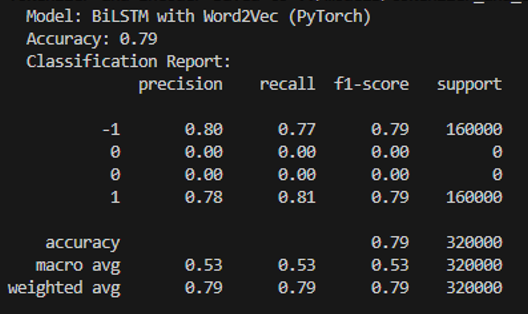
Trong đó:

* N: Số lượng các lớp trong bài toán phân loại.
* score i: Giá trị của chỉ số đánh giá (chẳng hạn như Precision, Recall hay F1-Score) đối với lớp thứ i. Mỗi lớp sẽ có một giá trị riêng cho từng chỉ số.
* **S**i (Support): Số lượng mẫu thực tế thuộc lớp thứ i. Đây là số liệu cho biết mức độ phổ biến của từng lớp trong tập dữ liệu.

## **3.2.Đánh giá mô hình trên tập test và điều chỉnh cải thiện mô hình**

### **3.2.1.Đánh giá mô hình**

Kết quả đạt được trên tập test sau quá trình huấn luyện mô hình trước khi điều chỉnh tham số:



Hình 10: kết quả đánh giá mô hình trước điều chỉnh

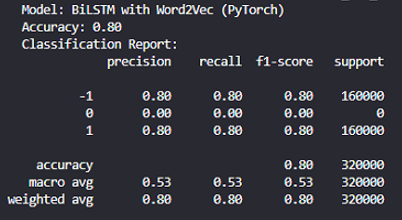
Trong đó:

* Độ chính xác (Accuracy) ~ 79%
* Cho thấy mô hình dự đoán đúng khoảng 79% tổng số tweet trong tập kiểm thử.
* Chỉ số Precision, Recall, F1-score
* Các lớp -1 (tiêu cực), 0 (trung tính), 1 (tích cực) đều có giá trị F1-score dao động trong 0.79, tương đối đồng đều.
* Điều này thể hiện mô hình không bị thiên lệch quá nhiều về một lớp nào đó.
* Macro avg và Weighted avg
* Macro avg F1 ~ 0.53, Weighted avg F1 ~ 0.79 => Mô hình đạt hiệu năng khá cân bằng trên cả ba lớp.

Lưu ý: kết quả này được tính ra khi chỉ phân tách câu thành các từ đơn.

### **3.2.2. Điều chỉnh mô hình cải thiện hiệu năng của mô hình**

Bằng việc thêm tách cặp từ trong quá trình tiền xử lý dữ liệu và huấn luyện đã cải thiện được độ chính xác của mô hình:



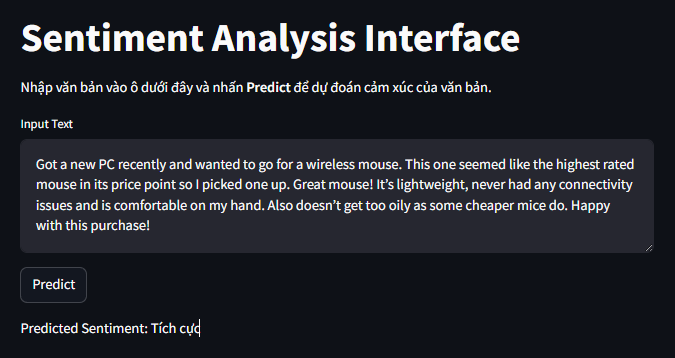
Hình 11: kết quả đánh giá mô hình sau điều chỉnh

Trong đó:

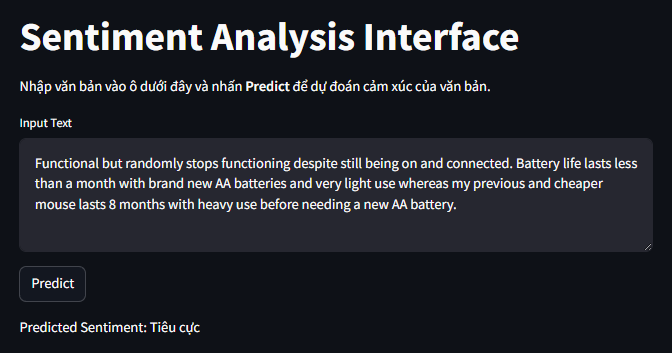
* Độ chính xác (Accuracy) ~ 80%
* Cho thấy mô hình dự đoán đúng khoảng 80% tổng số tweet trong tập kiểm thử.
* Chỉ số Precision, Recall, F1-score
* Các lớp -1 (tiêu cực), 0 (trung tính), 1 (tích cực) đều có giá trị F1-score dao động trong khoảng 0.80, tương đối đồng đều.
* Điều này thể hiện mô hình không bị thiên lệch quá nhiều về một lớp nào đó.
* Macro avg và Weighted avg
* Macro avg F1 ~ 0.53, Weighted avg F1 ~ 0.80 => Mô hình đạt hiệu năng khá cân bằng trên cả ba lớp.

3.2.4.Thử nghiệm trên dữ liệu mới

Thử nghiệm bằng việc nhập 1 đoạn văn bản bất kỳ để đánh giá cảm xúc trong văn bản đó:

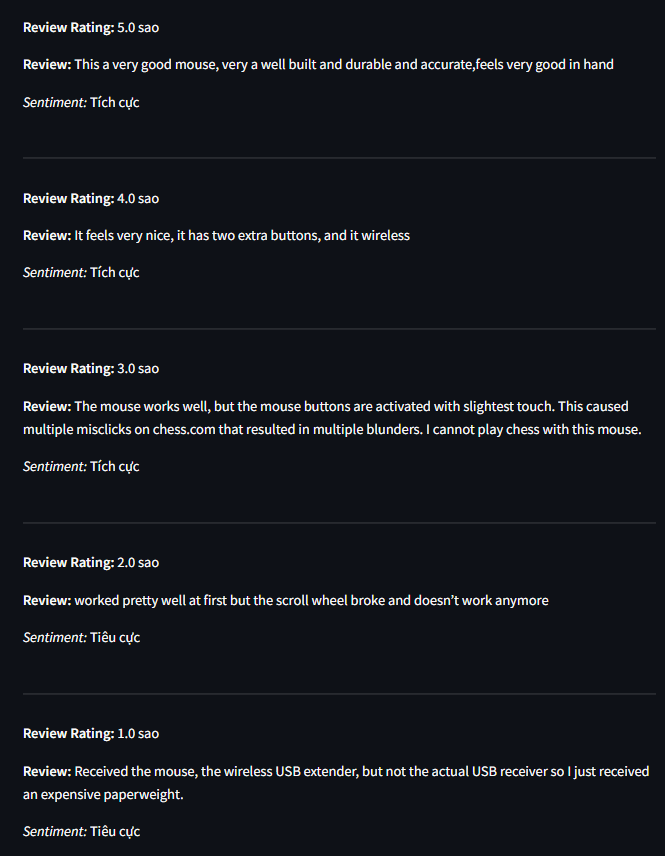


Hình 12: Hình ảnh trả về kết quả là tích cực



Hình 13: Hình ảnh trả về kết quả là tiêu cực

Thử nghiệm trên các đánh giá được lấy từ sản phầm trên trang thương mại điện tử amazon:



Hình 14: Hình ảnh trả về kết quả trả về với mỗi review lấy từ amazon

Như đã thấy trong ảnh sau khi crawl review của sản phẩm từ trang thương mại điện tử về thì mô hình đã có thể đánh giá cảm xúc trong văn bản với độ chính xác khá là cao.

# **Kết luận**

Qua quá trình khảo sát, phát triển và hoàn thiện, mô hình BiLSTM kết hợp Word2Vec đã cho thấy khả năng phân tích cảm xúc trên dữ liệu đánh giá sản phẩm một cách hiệu quả. Kết quả cuối cùng của báo cáo phân loại cho thấy mô hình đạt được độ chính xác khoảng 80% – một mức độ đáng mừng, nhất là khi so sánh với kết quả ban đầu trước khi áp dụng các kỹ thuật xử lý dữ liệu tiên tiến.

Để hiểu sâu hơn về các chỉ số đánh giá, cần phân tích chi tiết báo cáo phân loại:

* Độ chính xác (Accuracy):

Con số 80% phản ánh khả năng dự đoán đúng của mô hình trong hầu hết các trường hợp, mặc dù kết quả này có thể chịu ảnh hưởng từ sự mất cân bằng dữ liệu.

* Precision, Recall và F1-score theo từng lớp:
* Precision cho biết tỷ lệ dự đoán đúng trong số các đánh giá được xác định là một loại cảm xúc nhất định.
* Recall đo lường khả năng của mô hình trong việc nhận diện đầy đủ các mẫu thuộc một nhóm cảm xúc cụ thể.
* F1-score là chỉ số trung bình điều hòa của Precision và Recall, thể hiện sự cân đối giữa độ chính xác và khả năng nhận diện mẫu.

Với F1-score xấp xỉ 0.80 cho cả ba lớp cảm xúc (-1, 0, 1), mô hình cho thấy hiệu suất tương đối đồng đều giữa các nhóm.

Đáng lưu ý, việc tích hợp kỹ thuật tách cặp từ (bigram) đã giúp cải thiện đáng kể độ chính xác, nhấn mạnh tầm quan trọng của việc bắt trọn các cụm từ có ý nghĩa đặc biệt trong bối cảnh phân tích cảm xúc.

Thử nghiệm trên tập dữ liệu mới, bao gồm các đánh giá sản phẩm từ Amazon, đã chứng minh mô hình có khả năng khái quát hóa tốt và ứng dụng hiệu quả trong thực tiễn.

# **Tài liệu tham khảo**

1. Schuster, Mike, and Kuldip K. Paliwal. "Bidirectional recurrent neural networks." Signal Processing, IEEE Transactions on 45.11 (1997): 2673-2681.2. Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan

2. Salehinejad, Hojjat; Sankar, Sharan; Barfett, Joseph; Colak, Errol; Valaee, Shahrokh (2017). "Recent Advances in Recurrent Neural Networks". arXiv:1801.01078 [cs.NE].

3. Graves, Alex, Santiago Fernández, and Jürgen Schmidhuber. "Bidirectional LSTM networks for improved phoneme classification and recognition." Artificial Neural Networks: Formal Models and Their Applications–ICANN 2005. Springer Berlin Heidelberg, 2005. 799-804.

4. Graves, Alan, Navdeep Jaitly, and Abdel-rahman Mohamed. "Hybrid speech recognition with deep bidirectional LSTM." Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on. IEEE, 2013.

5. Sundermeyer, Martin, et al. "Translation modeling with bidirectional recurrent neural networks." Proceedings of the Conference on Empirical Methods on Natural Language Processing, October. 2014.