Machine Learning Comment Sentiment Analysis

Machine learning (Trường Đại học Bách khoa Hà Nội)

ĐẠI HỌC BÁCH KHOA HÀ NỘI

TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÁO CÁO ĐỒ ÁN MÔN HỌC

Nhập môn Học máy và Khai phá dữ liệu

Đề tài: Comment Sentiment Analysis

Danh sách thành viên:	Trần Đức Chính	20224906
	Đoàn Duy Tùng	20224906
	Ngô Đức Chung	20224827
	Hà Minh Hiếu	20220026

Hà Nội, 5-2025

Mục lục

Chapter 1

Giới thiệu

1.1 Bối cảnh vấn đề

Trong bối cảnh Internet và các nền tảng truyền thông xã hội phát triển nhanh chóng, người dùng thường xuyên bình luận trên các trang web, mạng xã hội và các dịch vụ trực tuyến. Những phản hồi này không chỉ thể hiện quan điểm cá nhân mà còn ảnh hưởng đáng kể đến danh tiếng thương hiệu, quyết định mua hàng của khách hàng và sự phát triển của các nền tảng trực tuyến. Với lượng dữ liệu khổng lồ được tạo ra mỗi ngày, việc phân tích và hiểu rõ cảm xúc của người dùng trở thành một nhiệm vụ quan trọng, giúp các doanh nghiệp và tổ chức điều chỉnh chiến lược kinh doanh, cải thiện dịch vụ và nâng cao trải nghiệm người dùng.

Phân tích cảm xúc bình luận ("Comment Sentiment Analysis") là một lĩnh vực nghiên cứu ứng dụng của xử lý ngôn ngữ tự nhiên (NLP) và học máy (Machine Learning). Kỹ thuật này giúp xác định xem nhận xét có tính chất tích cực, tiêu cực hay trung tính, từ đó cung cấp thông tin hữu ích để đánh giá chất lượng sản phẩm, dịch vụ và mức độ hài lòng của khách hàng. Ngày nay, nhiều ngành công nghiệp như thương mại điện tử, tài chính, truyền thông và chăm sóc khách hàng đều tận dụng phân tích cảm xúc để đưa ra quyết định nhanh chóng và chính xác hơn. Ngoài ra, các nền tảng mạng xã hội như Facebook, Twitter hay YouTube cũng áp dụng kỹ thuật này để kiểm soát nội dung, phát hiện xu hướng dư luận và cải thiện chính sách cộng đồng.

Mục tiêu của nhóm chúng tôi trong đề tài này là xây dựng và đánh giá các mô hình học máy để phân loại các bình luận trên các nền tảng trực tuyến với ba nhãn: tích cực, tiêu cực và trung tính. Bằng cách sử dụng các thuật toán học máy trong môn học, chúng tôi hướng đến việc tạo ra một hệ thống có khả năng tự động nhận diện và phân tích cảm xúc trong hàng loạt bình luận một cách chính xác và hiệu quả. Qua đó, hệ thống sẽ góp phần hỗ trợ các doanh nghiệp tối ưu hóa quyết định, nâng cao chất lượng dịch vụ, cải

thiện trải nghiệm người dùng và đề xuất các hành động phù hợp để đáp ứng nhu cầu của khách hàng. Trong tương lai, mô hình này có thể được mở rộng để áp dụng vào nhiều lĩnh vực khác nhau, từ phân tích dư luận xã hội, quản lý danh tiếng doanh nghiệp đến hỗ trợ hệ thống phản hồi tự động.

1.2 Về tập dữ liệu

Tập dữ liệu mà chúng tôi sử dụng được lấy từ Kaggle, được thu thập từ các bài đăng trên mạng xã hội, cụ thể là các bài đăng trên Twitter. Tập dữ liệu này chứa thông tin chi tiết về các bài đăng, bao gồm nội dung bài viết, cảm xúc của người đăng, thời gian đăng bài, độ tuổi của người dùng, quốc gia, và một số thông tin nhân khẩu học khác. Bộ dữ liệu đã được xử lý và làm sạch, đảm bảo không có bất kỳ trường thông tin nào bị thiếu sót. Đặc biệt, dữ liệu bao gồm các bài đăng với nhiều cảm xúc khác nhau như tích cực, tiêu cực và trung lập, giúp đảm bảo tính đa dạng và khách quan trong quá trình phân tích.

Bộ dữ liệu bao gồm 9 thuộc tính, trong đó có 1 thuộc tính mục tiêu là sentiment (cảm xúc) cho biết cảm xúc của bài đăng, và 8 thuộc tính khác liên quan đến nội dung và thông tin nhân khẩu học của người đăng bài. Những thuộc tính này bao phủ nhiều khía cạnh khác nhau của bài đăng và người dùng, chẳng hạn như nội dung bài viết, thời gian đăng bài, độ tuổi, và quốc gia. Dưới đây là bảng tổng quan chi tiết về các thuộc tính có trong bộ dữ liệu mà chúng tôi đang sử dụng:

Thuộc tính	Mô tả
textID	ID duy nhất của bài đăng.
text	Nội dung bài đăng trên Twitter.
sentiment	Cảm xúc của bài đăng (tích cực, tiêu cực, trung lập).
Time of Tweet	Thời gian đăng bài (sáng, trưa, tối).
Age of User	Độ tuổi của người dùng (phân loại theo nhóm tuổi).
Country	Quốc gia của người dùng.
Population - 2020	Dân số của quốc gia (năm 2020).
Land Area (Km²)	Diện tích đất của quốc gia (km²).
Density (P/Km²)	Mật độ dân số của quốc gia (người/km²).

Table 1.1: Bảng mô tả các thuộc tính của tập dữ liệu

Chapter 2

Khai phá và tiền xử lí dữ liệu

2.1 Khai phá dữ liệu

Dữ liệu gồm 27,481 dòng, mỗi dòng tương ứng với một dòng tweet có chứa thông tin văn bản, cảm xúc, thời gian đăng tải, độ tuổi người dùng và các đặc trưng về quốc gia (dân số, diện tích, mật độ dân số).

2.1.1 Phân bố các thuộc tính dữ liệu

Để hiểu rõ hơn về phân bố dữ liệu, ta xem xét tần suất xuất hiện của các thuộc tính phân loại và thống kê mô tả cho các thuộc tính số cũng như các thuộc tính văn bản:

Dựa vào biểu đồ 2.1 cho thấy các bình luận neutral chiếm tỷ lệ cao nhất (10,000 tweets), trong khi positive và negative gần như cân bằng (8,000 tweets). Phân bố độ tuổi người tweet rất đồng đều (16-17% mỗi nhóm). Thời gian tweet được đăng trải đều trong các khung giờ sáng, trưa và tối. Dữ liệu về dân số và diện tích quốc gia có phân bố lệch về trái, nghĩa là phần lớn quốc gia có dân số nhỏ và chỉ một số ít quốc gia có dân số rất cao (ví dụ: Trung Quốc, Ấn Độ). Điều này cũng tương tự với diện tích đất và mật độ dân số, khi phần lớn quốc gia có diện tích nhỏ, trong khi một số ít quốc gia có diện tích rất lớn nhưng mật độ dân số thấp. Các biểu đồ histogram phản ánh xu hướng này rõ ràng, với số lượng quốc gia đông dân cư hoặc có diện tích lớn chỉ chiếm một tỷ lệ nhỏ trong tập dữ liệu.

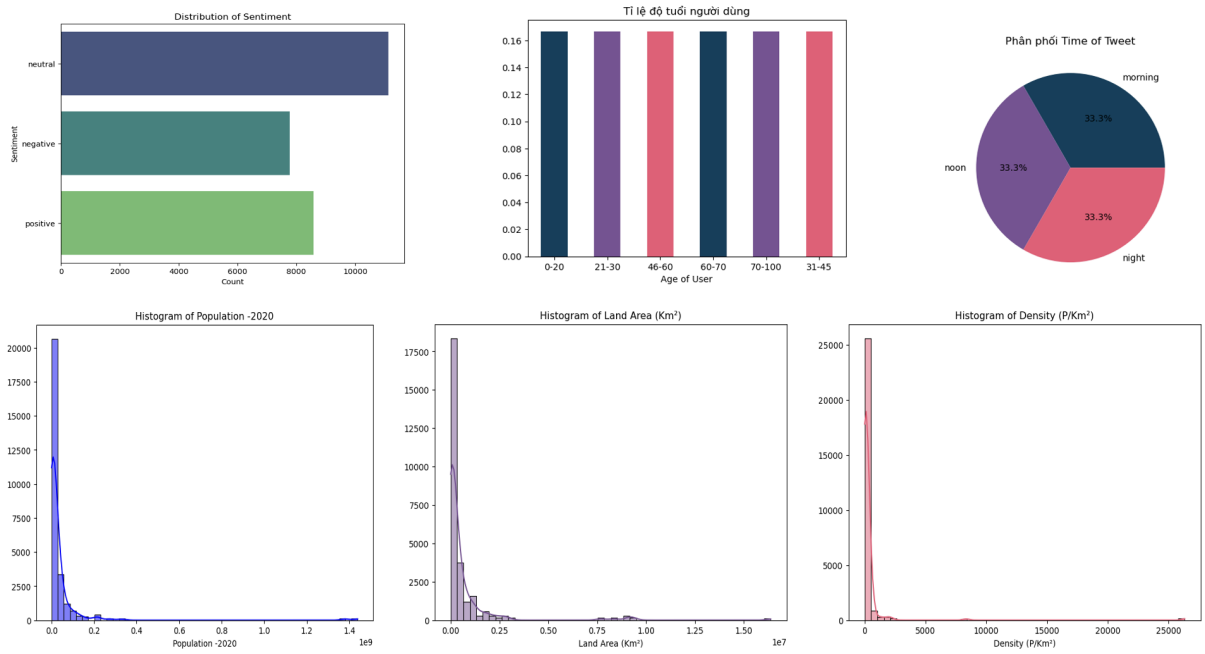

Figure 2.1: Biểu đồ phân tích các thuộc tính

2.1.2 Mối tương quan giữa thuộc tính phân loại đa giá trị và thuộc tính mục tiêu

Trong phần này, ta vẫn sử dụng biểu đồ cột chồng để tìm ra mối tương quan của các thuộc tính này với thuộc tính mục tiêu.

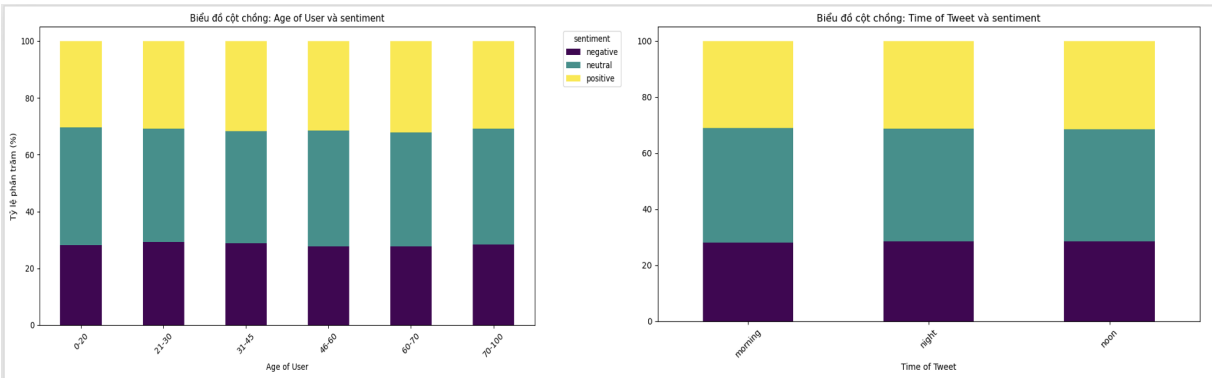

Figure 2.2: Tỉ lệ nhãn trên từng giá trị của thuộc tính đa giá trị

Dựa vào biểu đồ 2.2, có thể thấy độ tuổi và thời gian đăng tải bình luận có sự phân bố đồng đều giữa các nhãn cảm xúc, cho thấy rằng không có nhóm tuổi hoặc thời điểm cụ thể nào chiếm ưu thế rõ rệt trong việc thể hiện cảm xúc trên nền tảng.

2.1.3 Mối tương quan giữa các thuộc tính liên tục với thuộc tính mục tiêu

Trong phần này, ta sẽ sử dụng biểu đồ hộp, để phân tích mối quan hệ giữa thuộc tính liên tục với thuộc tính mục tiêu.

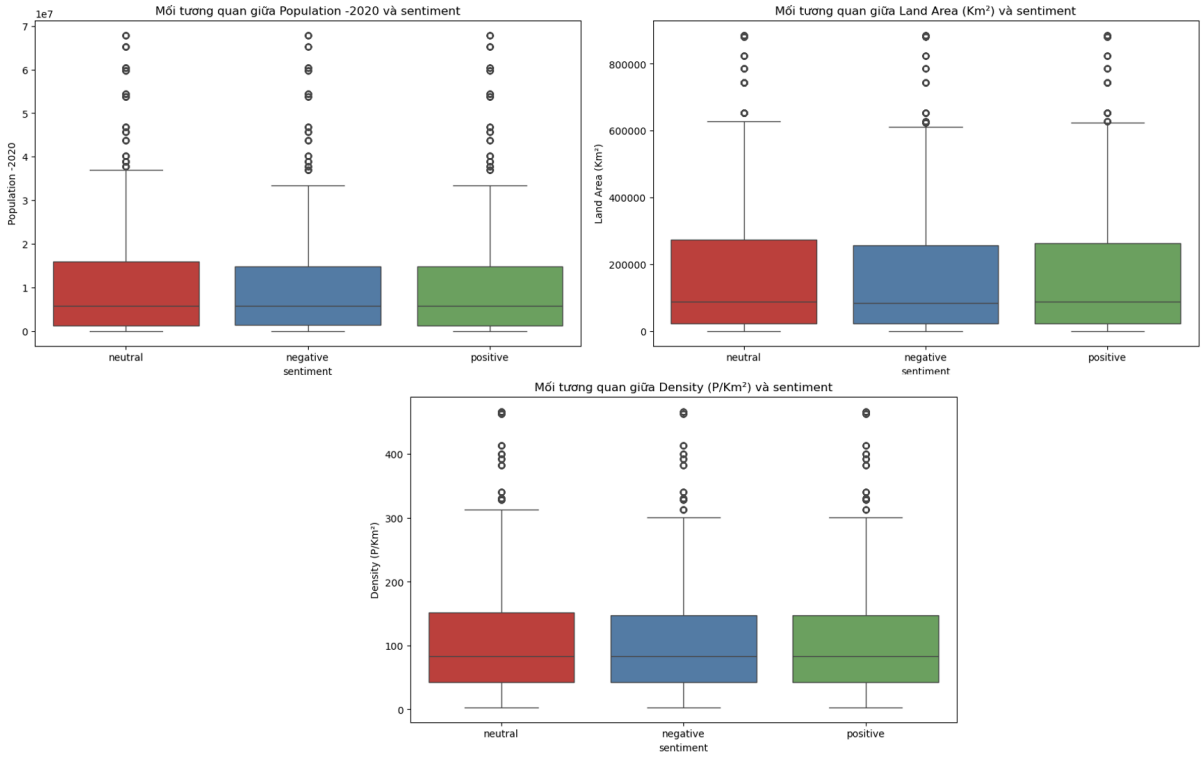

Figure 2.3: Tỉ lệ nhãn trên từng giá trị của thuộc tính liên tục

Dựa trên biểu đồ 2.3, có thể thấy rằng các thuộc tính dân số (Population - 2020), diện tích đất (Land Area) và mật độ dân số (Density) không có sự khác biệt rõ rệt giữa các nhóm cảm xúc (neutral, negative, positive). Các giá trị trung vị của ba thuộc tính này khá tương đồng giữa các nhóm, và sự phân tán dữ liệu cho thấy không có mối quan hệ rõ ràng giữa các thuộc tính này với sentiment. Điều này cho thấy rằng dân số, diện tích đất và mật độ dân số của quốc gia không ảnh hưởng đáng kể đến cảm xúc của bình luận. Ngoài ra, do sự phân bố dữ liệu khá rộng, có nhiều outlier xuất hiện, đặc biệt là ở các quốc gia có dân số hoặc diện tích lớn. Nhìn chung, các thuộc tính này có thể không đóng vai trò quan trọng trong việc phân loại cảm xúc của bình luận

2.2 Tiền xử lí dữ liệu

Trong quá trình phân tích cảm xúc bình luận (Comment Sentiment Analysis), tiền xử lý dữ liệu đóng vai trò then chốt trong việc nâng cao chất lượng dữ liệu đầu vào, giúp các thuật toán học máy hoạt động hiệu quả và chính xác hơn. Các bước tiền xử lý chủ yếu bao gồm chuẩn hóa văn bản, loại bỏ những yếu tố không cần thiết và chuyển đổi văn bản thành dạng mà các mô hình học máy có thể hiểu và xử lý.

Trong khi tiến hành khai phá dữ liệu, chúng tôi nhận thấy rằng các thuộc tính khác ngoài thuộc tính **text** (nội dung bình luận) không mang lại nhiều giá trị trong việc dự đoán cảm xúc của bình luận (sentiment). Các thuộc tính như **Age of User** (tuổi người dùng) hay **Population** (dân cư) mặc dù có thể có mối quan hệ với hành vi người dùng, nhưng trong trường hợp phân tích cảm xúc từ văn bản, chúng không có sự ảnh hưởng đáng kể đến kết quả phân loại cảm xúc. Do đó, để tối ưu hóa hiệu quả của mô hình, chúng tôi quyết định chỉ giữ lại thuộc tính **text** và loại bỏ những thuộc tính không mang lại giá trị quan trọng đối với dự đoán cảm xúc. Quá trình tiền xử lý này giúp mô hình tập trung vào những đặc trưng quan trọng, cải thiện chất lượng dữ liệu và làm giảm độ phức tạp không cần thiết trong các thuật toán học máy.

2.2.1 Làm sạch văn bản và tiền xử lí bổ sung

Quá trình chuẩn hóa văn bản giúp đảm bảo tính nhất quán của dữ liệu bằng cách loại bỏ các yếu tố không cần thiết và chuẩn hóa các từ ngữ. Cụ thể, các bước chuẩn hóa bao gồm:

- Chuyển đổi văn bản về chữ thường nhằm tránh sự phân biệt giữa các ký tự viết hoa và viết thường.
- Loại bỏ các liên kết URL, thẻ HTML và các ký tự đặc biệt để tránh nhiễu trong dữ liệu.
- Xóa các dấu câu, ký tự tab và ký tự xuống dòng để đảm bảo văn bản chỉ chứa nội dung cần thiết.
- Loại bỏ các từ chứa số nhằm đảm bảo chỉ giữ lại thông tin có ý nghĩa ngữ nghĩa.

Việc chuẩn hóa văn bản giúp loại bỏ những yếu tố không cần thiết, tạo điều kiện thuận lợi cho các phương pháp biểu diễn văn bản thành vector hoạt động chính xác hơn. Ngoài ra, chúng tôi áp dụng các phương pháp xử lý văn bản nâng cao giúp cải thiện chất lượng dữ liệu và làm giảm độ phức tạp của mô hình:

- **Loại bỏ Stop Words**: Stop words là những từ xuất hiện với tần suất cao trong văn bản nhưng không mang nhiều ý nghĩa trong việc phân biệt cảm xúc, chẳng hạn như "and", "the", "of" trong tiếng Anh. Việc loại bỏ stop words giúp giảm độ phức tạp của mô hình và tập trung vào các từ quan trọng hơn. Trong nghiên cứu này, stop words được loại bỏ dựa trên danh sách có sẵn từ thư viện xử lý ngôn ngữ tự nhiên (NLTK).
- **Lemmatization**: Lemmatization là quá trình đưa các từ về dạng gốc (lemma) nhằm giảm số lượng từ khác nhau trong tập dữ liệu mà vẫn giữ được ý nghĩa. Ví dụ, các từ "include", "includes", "included" đều được quy về dạng gốc "include". Phương pháp này giúp mô hình học máy dễ dàng tổng quát hóa hơn và cải thiện hiệu suất phân loại.
- **Loại bỏ Non-Words**: Dữ liệu bình luận thường chứa nhiều ký tự không phải từ (non-words) như số, dấu câu, ký tự tab và ký tự xuống dòng. Những ký tự này không mang nhiều ý nghĩa về mặt ngữ nghĩa và có thể gây nhiễu cho mô hình, do đó chúng được loại bỏ khỏi văn bản.

Dưới đây là một ví dụ minh họa về các bước tiền xử lý văn bản, đặt trong một khung để làm nổi bật nội dung.

Ví dụ về tiền xử lý văn bản

Câu gốc:

```
<html><body>Wow!!! The movie was amazing... but 2 hours long?!
\tCheck it out at: https://moviereview.com\n<br>It was quite a
    rollercoaster!</body></html>
```

Sau khi loại bỏ thẻ HTML, URL, dấu câu, số và xuống dòng:

```
Wow The movie was amazing but hours long
Check it out at It was quite a rollercoaster
```

Sau khi loại bỏ stop words:

```
movie amazing hours long check rollercoaster
```

Sau khi lemmatization:

```
movie amazing hour long check rollercoaster
```

Như vậy, qua các bước tiền xử lý, câu đã được làm gọn lại, giúp mô hình học máy tập trung vào thông tin quan trọng hơn.

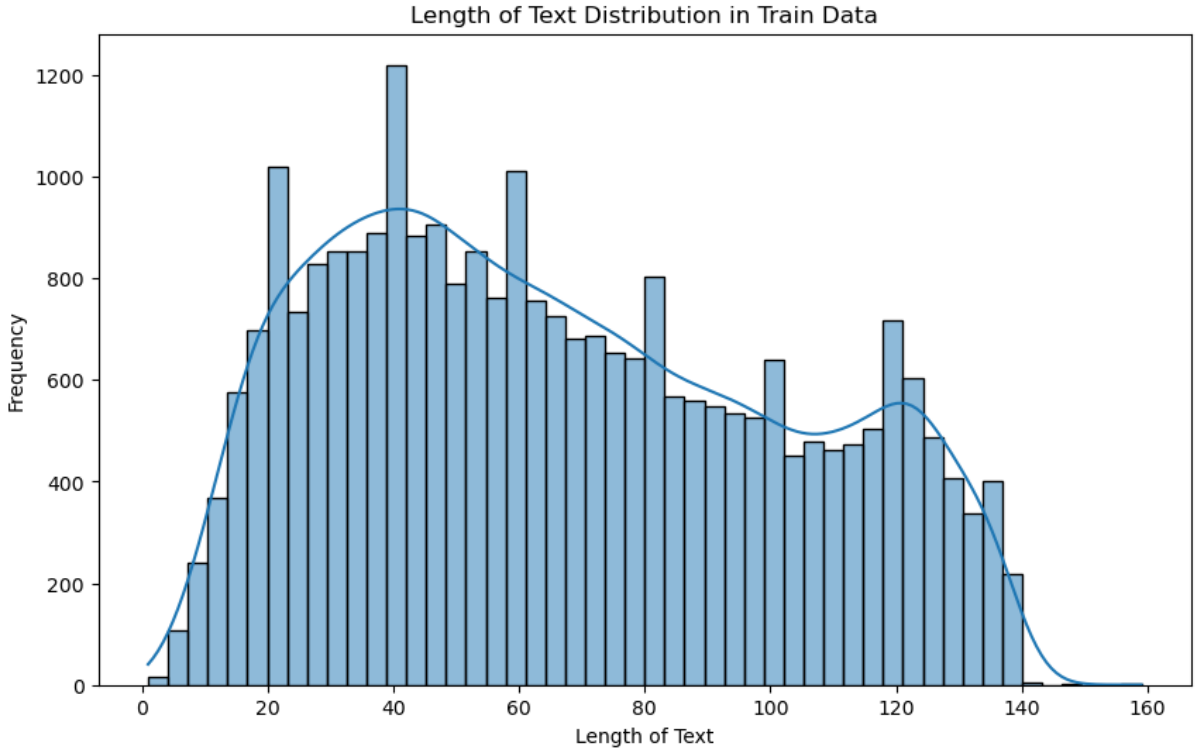

Figure 2.4: Phân bổ độ dài các đoạn text sau khi được làm sạch và tiền xử lí bổ sung

2.2.2 Chuyển đổi văn bản thành vector

Sau khi dữ liệu đã được làm sạch và chuẩn hóa, bước tiếp theo là chuyển đổi văn bản thành dạng số để có thể sử dụng trong các mô hình học máy. Dưới đây là ba phương pháp thông dụng và phổ biến để biểu diễn văn bản dưới dạng vector trong báo cáo này:

Biểu diễn nhị phân (Binary Representation)

Biểu diễn nhị phân là một trong những phương pháp đơn giản và nền tảng trong việc biến đổi văn bản thành dạng vector số để phục vụ cho các mô hình học máy. Ý tưởng chính của phương pháp này là biểu diễn mỗi tài liệu dưới dạng một vector nhị phân, trong đó mỗi phần tử tương ứng với một từ trong tập từ vựng (vocabulary).

Cụ thể, nếu một từ t_j xuất hiện trong tài liệu d_i, giá trị tại vị trí tương ứng trong vector là 1; ngược lại, nếu từ không xuất hiện, giá trị là 0. Phương pháp này chỉ quan tâm đến sự có mặt hay không của từ, hoàn toàn không xét đến tần suất xuất hiện của từ trong tài liệu.

$$V_{i,j} = \begin{cases} 1, & \text{nếu từ } t_j \text{ xuất hiện trong tài liệu } d_i \\ 0, & \text{nếu từ } t_j \text{ không xuất hiện trong tài liệu } d_i \end{cases} \quad (2.1)$$

Ví dụ minh hoạ: Giả sử chúng ta có tập từ vựng gồm 5 từ: `["học", "máy", "trí", "tuệ", "nhân tạo"]`, và tài liệu d_1 có nội dung là: *"trí tuệ nhân tạo"*.

Khi đó, vector biểu diễn nhị phân của tài liệu d_1 theo thứ tự các từ trong tập từ vựng là:

$$\mathbf{v}_{d_1} = [0,\ 0,\ 1,\ 1,\ 1]$$

Trong đó:

- "học" không xuất hiện trong $d_1 \Rightarrow 0$
- "máy" không xuất hiện trong $d_1 \Rightarrow 0$
- "trí", "tuệ", "nhân tạo" đều xuất hiện $\Rightarrow$ các giá trị tương ứng là 1

Hạn chế: Mặc dù đơn giản và dễ triển khai, phương pháp biểu diễn nhị phân tồn tại một số nhược điểm sau:

- **Không phản ánh tần suất từ:** Việc một từ xuất hiện nhiều lần trong tài liệu không làm tăng trọng số của từ đó, dẫn đến mất đi thông tin về mức độ quan trọng tương đối của từ trong ngữ cảnh tài liệu.
- **Tính phân biệt thấp:** Hai tài liệu có thể có biểu diễn giống hệt nhau nếu chúng chứa cùng một tập từ, bất kể độ dài hay nội dung chi tiết khác nhau.
- **Độ thưa cao (sparse vector):** Với tập từ vựng lớn, hầu hết các vector nhị phân đều chứa rất nhiều số 0, gây tốn bộ nhớ và giảm hiệu quả tính toán.
- **Không xử lý được ý nghĩa ngữ cảnh:** Phương pháp này không xem xét ngữ nghĩa của từ hoặc mối liên hệ giữa các từ, do đó không phù hợp cho các mô hình cần hiểu ngữ cảnh sâu sắc.

Phương pháp này thường được sử dụng như một bước khởi đầu để hiểu và thử nghiệm với dữ liệu văn bản, nhưng trong các ứng dụng thực tế yêu cầu hiệu quả cao hơn, người ta thường thay thế hoặc kết hợp nó với các kỹ thuật nâng cao hơn như *TF-IDF*, *Word2Vec*, hoặc *BERT*.

Bag of Words (BoW)

Bag of Words (BoW) là một trong những phương pháp phổ biến và nền tảng trong xử lý ngôn ngữ tự nhiên (NLP), dùng để chuyển đổi văn bản thành vector số phục vụ cho

các mô hình học máy. Ý tưởng chính của BoW là biểu diễn một tài liệu bằng cách đếm số lần xuất hiện của từng từ trong tập từ vựng, mà không quan tâm đến ngữ cảnh, vị trí hay ngữ pháp của từ đó.

Mỗi tài liệu được biểu diễn dưới dạng một vector, trong đó mỗi chiều tương ứng với một từ trong tập từ vựng. Một ma trận BoW có các dòng đại diện cho tài liệu và các cột đại diện cho từ trong toàn bộ tập dữ liệu.

$$V_{i,j} = \text{freq}(t_j, d_i) \tag{2.2}$$

Trong đó:

- $V_{i,j}$ là số lần từ t_j xuất hiện trong tài liệu d_i
- $\text{freq}(t_j, d_i)$ là hàm đếm tần suất từ t_j trong tài liệu d_i

Ví dụ minh hoạ: Giả sử tập từ vựng là: `["học", "máy", "trí", "tuệ", "nhân tạo"]`, và ta có hai tài liệu như sau:

- d_1: *"trí tuệ nhân tạo"*
- d_2: *"học máy là một nhánh của trí tuệ nhân tạo"*

Khi đó, ta có thể biểu diễn các tài liệu bằng ma trận BoW như sau:

	học	**máy**	**trí**	**tuệ**	**nhân tạo**
d_1	0	0	1	1	1
d_2	1	1	1	1	1

(Bỏ qua các từ không nằm trong tập từ vựng như "là", "một", "nhánh", "của")

Loại	Mô tả
Ưu điểm	• Đơn giản, dễ cài đặt và hoạt động hiệu quả trong nhiều trường hợp cơ bản. • Thường được sử dụng như bước đầu trong pipeline xử lý văn bản trước khi áp dụng các mô hình phức tạp hơn.
Hạn chế	• **Không xét ngữ cảnh:** Mỗi từ được xem xét độc lập, không phản ánh quan hệ cú pháp hay ngữ nghĩa. • **Không đánh giá được tầm quan trọng của từ:** Các từ phổ biến vẫn có thể chiếm trọng số lớn mặc dù không mang nhiều thông tin phân biệt. • **Vector có kích thước lớn:** Khi tập từ vựng mở rộng, vector biểu diễn trở nên dài và thưa, ảnh hưởng đến hiệu năng.

Table 2.1: Đặc điểm và hạn chế của phương pháp Bag of Words (BoW)

BoW thường được cải tiến bằng cách kết hợp với các phương pháp trọng số như TF-IDF để khắc phục các nhược điểm về ngữ nghĩa và độ quan trọng của từ.

TF-IDF (Term Frequency - Inverse Document Frequency)

TF-IDF là một phương pháp nổi bật trong xử lý ngôn ngữ tự nhiên, nhằm đánh giá tầm quan trọng của một từ trong một tài liệu cụ thể, so với toàn bộ tập tài liệu. Không giống như BoW chỉ đếm tần suất từ, TF-IDF điều chỉnh tầm quan trọng này dựa trên tần suất xuất hiện của từ trong toàn bộ tập dữ liệu. Điều này giúp làm giảm trọng số của các từ phổ biến (như "là", "của", "và",...) và làm nổi bật các từ có tính phân biệt cao.

Công thức tổng quát:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t) \tag{2.3}$$

Trong đó:

- $\mathrm{TF}(t,d)$ là tần suất tương đối của từ t trong tài liệu d:

$$\mathrm{TF}(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{2.4}$$

- $\mathrm{IDF}(t)$ là độ đo nghịch đảo của số tài liệu chứa từ t, giúp giảm trọng số của các từ xuất hiện phổ biến:

$$\mathrm{IDF}(t) = \log\left(\frac{N}{1 + |\{d \in D : t \in d\}|}\right) \tag{2.5}$$

Trong đó:

- $f_{t,d}$ là số lần từ t xuất hiện trong tài liệu d
- N là tổng số tài liệu trong tập dữ liệu
- $|\{d \in D : t \in d\}|$ là số lượng tài liệu chứa từ t

Ví dụ minh hoạ: Giả sử có 3 tài liệu:

- d_1: *"trí tuệ nhân tạo"*
- d_2: *"trí tuệ nhân tạo và học máy"*
- d_3: *"học máy là tương lai"*

Giả sử ta tính TF-IDF cho từ "trí" trong tài liệu d_2:

- $\mathrm{TF}(\text{"trí"}, d_2) = \frac{1}{6}$ (do d_2 có 6 từ)
- "trí" xuất hiện trong 2 tài liệu (d_1, d_2) $\rightarrow \mathrm{IDF}(\text{"trí"}) = \log\left(\frac{3}{1+2}\right) = \log(1) = 0$

$\Rightarrow \mathrm{TF\text{-}IDF}(\text{"trí"}, d_2) = \frac{1}{6} \times 0 = 0$

Điều này phản ánh rằng từ "trí" không mang nhiều thông tin phân biệt trong tập tài liệu, vì nó xuất hiện ở nhiều tài liệu.

Ưu điểm của TF-IDF	Hạn chế của TF-IDF
Giảm nhiễu từ phổ biến: TF-IDF gán trọng số thấp cho những từ xuất hiện rộng rãi trong nhiều tài liệu, giúp làm nổi bật từ quan trọng hơn trong tài liệu cụ thể.	**Không xét ngữ cảnh:** Phương pháp không quan tâm đến vị trí, thứ tự hay ngữ nghĩa của từ trong văn bản.
Tăng độ phân biệt: Các từ mang tính đặc trưng trong một tài liệu riêng lẻ sẽ được đánh trọng số cao hơn, giúp phân biệt giữa các tài liệu tốt hơn.	**Không xử lý từ đồng nghĩa/đa nghĩa:** Những từ khác nhau nhưng có cùng nghĩa vẫn được coi là riêng biệt.
Hiệu quả thực tiễn cao: TF-IDF đã được kiểm chứng trong nhiều ứng dụng như phân loại văn bản, truy xuất thông tin, phân cụm, v.v.	**Vector thưa (sparse vector):** Với tập từ vựng lớn, biểu diễn TF-IDF vẫn dẫn đến không gian vector thưa, ảnh hưởng đến hiệu suất tính toán.

Table 2.2: So sánh ưu điểm và hạn chế của phương pháp TF-IDF

TF-IDF là cầu nối giữa các phương pháp biểu diễn văn bản đơn giản (như BoW) và các phương pháp nâng cao (như Word2Vec, BERT), cung cấp một cách tiếp cận hiệu quả để xử lý văn bản với độ phức tạp trung bình.

Chapter 3

Các mô hình Học máy

3.1 Logistic Regession

3.1.1 Tổng quan

Logistic Regression là một thuật toán học máy có giám sát, chủ yếu dùng cho bài toán phân loại nhị phân (ví dụ: phân loại email là spam hoặc không spam). Thuật toán dự đoán xác suất một điểm dữ liệu thuộc vào một lớp cụ thể (0 hoặc 1) dựa trên các đặc trưng của nó. Bằng cách sử dụng hàm sigmoid, Logistic Regression chuyển đổi giá trị đầu ra thành xác suất trong khoảng từ 0 đến 1, từ đó gán nhãn lớp cho dữ liệu. Mặc dù dựa trên tổ hợp tuyến tính của đặc trưng, nó khác với hồi quy tuyến tính vì tập trung vào phân loại thay vì dự đoán giá trị liên tục. Thuật toán đơn giản, dễ triển khai và hiệu quả cho các bài toán phân loại cơ bản.

3.1.2 Nguyên lý hoạt động

Đầu vào:

- **Tập dữ liệu huấn luyện:** Bao gồm các điểm dữ liệu với đặc trưng (biến độc lập) và nhãn tương ứng (0 hoặc 1 trong phân loại nhị phân).
- **Điểm dữ liệu mới:** Một điểm dữ liệu cần được dự đoán lớp.
- **Tham số:**
  - **Tốc độ học (learning rate):** Điều chỉnh kích thước bước trong quá trình tối ưu gradient descent.
  - **Số lần lặp:** Số lần thuật toán cập nhật tham số mô hình.

  - **Tham số điều chuẩn (tùy chọn):** Kiểm soát sự cân bằng giữa việc khớp dữ liệu huấn luyện và giữ mô hình đơn giản để tránh quá khớp (ví dụ: điều chuẩn L1 hoặc L2).

- **Hàm mất mát:** Thường sử dụng hàm log-loss (cross-entropy loss), đo lường sự khác biệt giữa xác suất dự đoán và nhãn thực tế.

3.1.3 Triển khai

Chúng tôi đã tiến hành đánh giá hiệu suất của thuật toán Logistic Regression trên tập dữ liệu của mình với các siêu tham số sau:

- **Tốc độ học (learning rate):** Đây là tham số quan trọng, ảnh hưởng đến tốc độ hội tụ của thuật toán. Giá trị phù hợp giúp mô hình đạt được hiệu suất tốt mà không bị kẹt ở các điểm tối ưu cục bộ.

- **Số lần lặp:** Quyết định số lần mô hình cập nhật trọng số và độ chệch, ảnh hưởng đến mức độ khớp của mô hình với dữ liệu.

- **Điều chuẩn (regularization):** Chúng tôi sử dụng các phương pháp điều chuẩn phổ biến như:

  - **L1 (Lasso):** Thêm tổng giá trị tuyệt đối của các trọng số vào hàm mất mát, giúp chọn lọc đặc trưng.
  - **L2 (Ridge):** Thêm tổng bình phương của các trọng số, giúp giảm độ lớn của trọng số và tránh quá khớp.

- **Ngưỡng phân loại:** Thông thường đặt ở mức 0.5, nhưng có thể điều chỉnh tùy thuộc vào bài toán để tối ưu hóa các chỉ số như precision hoặc recall.

3.2 K-Means Clustering

3.2.1 Tổng quan

Để giới thiệu về K-Means, trước tiên ta đề cập đến phương pháp học không giám sát dùng để phân cụm dữ liệu. Nguyên tắc chính của phương pháp này là nhóm các điểm dữ liệu có đặc trưng giống nhau vào cùng một cụm dựa trên khoảng cách tới các tâm cụm. Khi chọn số lượng cụm là k, thuật toán K-Means sẽ tự động phân chia dữ liệu thành k nhóm, tối ưu hóa vị trí các tâm cụm để giảm thiểu tổng bình phương khoảng cách trong

mỗi cụm. K-Means là một thuật toán đơn giản, hiệu quả, thường được sử dụng trong các bài toán như phân đoạn khách hàng hoặc phân tích hình ảnh.

3.2.2 Nguyên lý hoạt động

Đầu vào:

- Một tập dữ liệu gồm các điểm dữ liệu với các đặc trưng (không cần nhãn).
- Số lượng cụm k (tham số do người dùng chọn).
- Độ đo khoảng cách (thường là khoảng cách Euclid).
- Số lần lặp tối đa hoặc ngưỡng hội tụ.

Đầu ra:

- Nhãn cụm cho mỗi điểm dữ liệu (từ 0 đến k–1).
- Tọa độ của k tâm cụm.

3.2.3 Triển khai

Chúng tôi đã triển khai và đánh giá hiệu suất của thuật toán K-Means trên tập dữ liệu với các siêu tham số sau:

- **Số lượng cụm *k*:** Đây là tham số quan trọng, ảnh hưởng trực tiếp đến chất lượng phân cụm. Việc chọn k phù hợp thường dựa trên phương pháp Elbow hoặc Silhouette score.
- **Độ đo khoảng cách:** Sử dụng khoảng cách Euclid để đo sự tương đồng giữa các điểm dữ liệu và tâm cụm.
- **Khởi tạo tâm cụm:** Sử dụng phương pháp K-Means++ để chọn các tâm ban đầu, giúp cải thiện tốc độ hội tụ và chất lượng cụm.
- **Số lần lặp tối đa:** Đặt là 500 để đảm bảo thuật toán hội tụ trong hầu hết các trường hợp.

3.3 K-Nearest Neighbors (k-NN)

3.3.1 Tổng quan

Để giới thiệu về KNN, trước tiên ta đề cập đến phương pháp học dựa trên hàng xóm. Nguyên tắc chính của phương pháp này là những điểm dữ liệu có thuộc tính giống nhau thường có nhãn tương tự. Khi giới hạn số lượng hàng xóm của một điểm dữ liệu mới là k, ta có thể dự đoán nhãn của điểm đó, từ đó hình thành thuật toán k láng giềng gần nhất (KNN). KNN còn được gọi là Lazy learning do pha huấn luyện của thuật toán chỉ cần lưu trữ các điểm dữ liệu.

3.3.2 Nguyên lý hoạt động

Đầu vào:

- Một tập dữ liệu huấn luyện gồm các điểm dữ liệu bao gồm đặc trưng và nhãn.
- Một điểm dữ liệu mới cần phân loại hoặc dự đoán.
- Số lượng lân cận k (tham số do người dùng chọn).
- Độ đo tương đồng.

Đầu ra: Nhãn của điểm dữ liệu mới.

Cách dự đoán:

- Tính khoảng cách từ điểm dữ liệu mới đến tất cả các điểm trong tập huấn luyện (thường dùng khoảng cách Euclid).
- Chọn k điểm gần nhất dựa trên khoảng cách đã tính.
- Chọn nhãn phổ biến nhất trong k điểm.

3.3.3 Triển khai

Chúng tôi đã tiến hành đánh giá hiệu suất của thuật toán này trên tập dữ liệu của mình bằng cách sử dụng các siêu tham số sau.

- Số lượng lân cận k: Đây là một tham số quan trọng của mô hình, ảnh hưởng trực tiếp đến hiệu suất dự đoán. Việc lựa chọn giá trị k phù hợp đóng vai trò quyết định trong độ chính xác của thuật toán.

- Độ đo tương đồng: Việc lựa chọn độ đo để đánh giá độ tương đồng giữa các điểm dữ liệu góp phần ảnh hưởng tới hiệu quả thuật toán. Chúng tôi sẽ sử dụng các độ đo phổ biến như: Euclidean, Manhattan, Cosine.
- Cách đánh trọng số đối với điểm dữ liệu lân cận: Chúng tôi triển khai đánh trọng số cho lân cận bằng 2 độ đo là Uniform (Các lân cận có trọng số như nhau) và Distance (Các lân cận có khoảng cách càng gần sẽ có trọng số càng cao).

3.4 Random Forest

3.4.1 Tổng quan

Random Forest là một thuật toán học máy được phát triển từ thuật toán Decision Tree, nhưng thay vì chỉ sử dụng một cây quyết định duy nhất, Random Forest xây dựng và kết hợp nhiều cây quyết định (hình thành một "rừng" các cây), giúp cải thiện độ chính xác và giảm thiểu hiện tượng overfitting.

Desicion tree (Cây quyết định) là một mô hình học có giám sát, trong đó mỗi nút đại diện cho một biến. Các đường nối giữa các nút cho thấy giá trị cụ thể của biến đó. Mỗi nút lá biểu thị giá trị dự đoán của biến mục tiêu, dựa trên các giá trị của các biến đã được xác định bởi đường đi từ nút gốc đến nút lá. Theo một cách dễ hiểu, Decision tree là tập hợp của các quy luật IF-ELSE.

3.4.2 Cây quyết định

Nguyên lý hoạt động của cây quyết định dựa trên quá trình phân chia dữ liệu thành các nhánh (branches) và các nút lá (leaf nodes) qua các quyết định dựa trên giá trị của các đặc trưng.

Cấu tạo của 1 cây quyết định bao gồm:

- Gốc: Là điểm bắt đầu của cây quyết định, nơi toàn bộ dữ liệu huấn luyện hoặc dữ liệu cần dự đoán được đưa vào để xử lý.
- Nút quyết định: Là các điểm trong cây, nơi dữ liệu được phân chia dựa trên các điều kiện (hoặc tiêu chí). Mỗi điều kiện tương ứng với một thuộc tính cụ thể của mẫu dữ liệu.
- Nhánh: Là các đường nối giữa các nút, thể hiện kết quả của quá trình phân chia dữ liệu tại các nút quyết định.

- Lá: Là các điểm cuối cùng trong cây, nơi quá trình phán đoán kết thúc, và mẫu dữ liệu được gán nhãn hoặc giá trị dự đoán.

Các bước cơ bản trong quá trình xây dựng cây quyết định bao gồm:

- Chọn đặc trưng tại các nút để chia tách dữ liệu: Tại mỗi nút, mô hình sẽ lựa chọn một đặc trưng để phân chia dữ liệu sao cho dữ liệu được phân tách sao cho hiệu quả nhất. Các tiêu chí thường được sử dụng bao gồm:
  - Entropy: là một tiêu chí được sử dụng trong cây quyết định để đo mức độ hỗn loạn trong dữ liệu. Entropy càng thấp có nghĩa là dữ liệu càng đồng nhất. Công thức tính Entropy như sau:

$$Entropy(t) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

Trong đó: p_i là tỷ lệ của lớp i trong tập dữ liệu tại nút t.
k là số lượng lớp trong tập dữ liệu.

Entropy thường được sử dụng trong cây ID3 nhằm chọn đặc trưng sao cho $Information\ Gain$ là lớn nhất.

  - Gini index: là một tiêu chí được sử dụng chủ yếu trong phân loại để đo mức độ không đồng nhất của một tập dữ liệu. Giá trị Gini càng thấp, độ đồng nhất của dữ liệu càng cao. Công thức tính Gini như sau:

$$Gini(t) = 1 - \sum_{i=1}^{k} p_i^2$$

- Chia tách dữ liệu: Dữ liệu sẽ được phân chia thành các nhánh con dựa trên thuộc tính được chọn.
- Lặp lại quá trình trên đến khi đạt 1 trong các điều kiện:
  - Số lượng dữ liệu tại nút con không đủ lớn.
  - Cây đạt độ sâu tối đa.
  - Không còn đặc trưng nào có thể phân chia hiệu quả.

3.4.3 Nguyên lý hoạt động

Random Forest hoạt động dựa trên 3 nguyên tắc chính:

- Bagging(Bootstrap Aggregating): là một phương pháp tạo ra nhiều bộ dữ liệu con bằng cách lấy ngẫu nhiên một phần từ bộ dữ liệu huấn luyện ban đầu (1 mẫu dữ liệu có thể xuất hiện nhiều lần trong 1 bộ dữ liệu con).
- Chọn ngẫu nhiên đặc trưng: Khi xây dựng mỗi cây quyết định, thay vì xem xét tất cả các đặc trưng có sẵn để chọn điểm phân chia, Random Forest chỉ xem xét một tập con ngẫu nhiên của các đặc trưng tại mỗi nút phân chia. Điều này giúp làm giảm sự tương quan giữa các cây trong rừng, khiến cho mô hình tổng thể mạnh mẽ và ít bị overfitting.
- Kết hợp nhiều cây quyết định để đưa ra dự đoán: Dựa trên cơ chế đa số, lớp được đa số các cây lựa chọn sẽ là dự đoán cuối cùng của mô hình.

3.4.4 Triển khai

Chúng tôi đã tiến hành đánh giá hiệu suất của thuật toán này trên tập dữ liệu của mình bằng cách sử dụng các siêu tham số sau:

- Số lượng cây quyết định.
- Tiêu chí chọn đặc trưng.
- Độ sâu tối đa của cây.
- Số mẫu tối thiểu để chia một nút.
- Số mẫu tối thiểu cần thiết tại mỗi lá.

3.5 Support Vector Machines - SVM

3.5.1 Tổng quan

SVM là thuật toán học có giám sát, hoạt động bằng cách tìm siêu phẳng tối ưu để phân tách các lớp dữ liệu sao cho khoảng cách giữa chúng là lớn nhất.

3.5.2 Nguyên lý hoạt động

SVM tìm kiếm một siêu phẳng tối ưu để phân chia các điểm dữ liệu của hai lớp sao cho khoảng cách giữa siêu phẳng và các điểm dữ liệu gần nhất của mỗi lớp là lớn nhất. Siêu phẳng trong không gian n chiều sẽ là mặt phẳng có $n-1$ chiều.

Các điểm dữ liệu gần nhất này gọi là support vectors, chúng là những điểm quyết định vị trí của siêu phẳng. Thực tế, chỉ có các điểm này ảnh hưởng đến quá trình huấn luyện, các điểm còn lại không ảnh hưởng gì đến vị trí của siêu phẳng. Khoảng cách từ siêu phẳng đến các điểm dữ liệu gần nhất của mỗi lớp được gọi là margins. Mục tiêu của SVM là tối đa hóa margin.

Khi dữ liệu không thể phân chia được bằng một siêu phẳng trong không gian dữ liệu ban đầu, SVM sử dụng kernel trick để ánh xạ dữ liệu vào không gian có chiều cao hơn, nơi dữ liệu có thể phân chia được bằng một siêu phẳng. Một số kernel phổ biến bao gồm:

- Linear kernel
- Polynomial kernel
- RBF kernel

3.5.3 Triển khai

Chúng tôi đã tiến hành đánh giá hiệu suất của thuật toán này trên tập dữ liệu của mình bằng cách sử dụng các siêu tham số sau:

- *C*: Hệ số điều chỉnh giữa độ chính xác và margin.
- kernel: Các kernel trong SVM.

3.6 Long-Short Term Memory - LSTM

3.6.1 Tổng quan

LSTM (Long Short-Term Memory) là một biến thể của mạng nơ-ron hồi tiếp (RNN) được thiết kế để xử lý các bài toán liên quan đến chuỗi dữ liệu, chẳng hạn như xử lý ngôn ngữ tự nhiên và phân tích cảm xúc. Ưu điểm nổi bật của LSTM là khả năng ghi nhớ thông tin trong khoảng thời gian dài, qua đó giải quyết vấn đề "gradient vanish" thường gặp ở RNN thông thường. Ở đây, chúng ta sẽ áp dụng mô hình LSTM để phân loại cảm xúc của các đoạn văn bản.

3.6.2 Nguyên lý hoạt động

Quá trình xử lý dữ liệu và xây dựng mô hình LSTM có thể được thể hiện như sau:

1. Tokenization và Padding Trong các phương pháp truyền thống như TF-IDF, BoW hay biểu diễn nhị phân, mỗi văn bản được chuyển thành một vector đặc trưng có chiều cố định, nhưng mất đi thông tin về thứ tự từ ngữ trong câu. Những phương pháp này phù hợp với các mô hình tuyến tính (Logistic Regression, SVM) nhưng không thích hợp cho các kiến trúc học sâu như LSTM, vốn yêu cầu chuỗi dữ liệu có thứ tự thời gian hoặc ngữ nghĩa để khai thác ngữ cảnh cục bộ và toàn cục. Do đó, chúng tôi không sử dụng các phương pháp vector hóa truyền thống, mà thiết kế một tokenizer riêng biệt (tạm gọi **Index-based Tokenizer**) phù hợp cho LSTM như sau:
Giả sử tập dữ liệu văn bản là

$$D = \{d_i \mid i = 1, \ldots, N\},$$

với mỗi văn bản d_i được biểu diễn dưới dạng chuỗi các từ:

$$d_i = (w_{i,1}, w_{i,2}, \ldots, w_{i,T_i}),$$

trong đó T_i là số từ trong d_i.

Ta xây dựng từ điển $\mathcal{V}$ gồm V từ xuất hiện trong tập dữ liệu. Mỗi từ $w \in \mathcal{V}$ được ánh xạ thành một số nguyên qua hàm tokenization $T(\cdot)$:

$$T(w) = \begin{cases} index(w) & \text{nếu } w \in \mathcal{V}, \\ 0 & \text{nếu } w \notin \mathcal{V}. \end{cases}$$

Do đó, mỗi văn bản d_i được chuyển thành chuỗi số:

$$T(d_i) = (T(w_{i,1}), T(w_{i,2}), \ldots, T(w_{i,T_i})).$$

Để đảm bảo tất cả các chuỗi có độ dài cố định $L_{\max}$ (được lấy từ chuỗi dài nhất của tập huấn luyện), ta tiến hành padding:

$$\tilde{d_i} = \begin{cases} (T(w_{i,1}), \ldots, T(w_{i,T_i}), 0, \ldots, 0) & \text{nếu } T_i < L_{\max}, \\ (T(w_{i,1}), \ldots, T(w_{i,L_{\max}})) & \text{nếu } T_i \geq L_{\max}. \end{cases}$$

2. Biểu diễn Embedding Mỗi từ trong từ điển được biểu diễn bằng một vector thông qua ma trận embedding:

$$E \in \mathbb{R}^{V \times d},$$

với d là kích thước vector embedding (ví dụ, $d = 256$). Với một chuỗi đã được padding $\tilde{d_i} = (i_1, i_2, \ldots, i_{L_{\max}})$, ta có dãy vector:

$$\mathbf{x}_i = (E[i_1], E[i_2], \ldots, E[i_{L_{\max}}]) \in \mathbb{R}^{L_{\max} \times d}.$$

3. Mô hình LSTM Mô hình LSTM xử lý chuỗi vector $\mathbf{x}_i$ theo các bước thời gian $t = 1, 2, \ldots, L_{\max}$. Ở mỗi bước thời gian, với đầu vào $\mathbf{x}_t \in \mathbb{R}^d$, trạng thái ẩn $\mathbf{h}_{t-1} \in \mathbb{R}^n$ và trạng thái tế bào $\mathbf{C}_{t-1} \in \mathbb{R}^n$, các cổng của LSTM được tính như sau:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(W_f[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_f), \\
\mathbf{i}_t &= \sigma(W_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i), \\
\tilde{\mathbf{C}}_t &= \tanh(W_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C), \\
\mathbf{C}_t &= \mathbf{f}_t \odot \mathbf{C}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{C}}_t, \\
\mathbf{o}_t &= \sigma(W_o[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_o), \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{C}_t),
\end{aligned}$$

trong đó:

- σ là hàm sigmoid,
- tanh là hàm hyperbolic tangent, $\tanh(x) = \dfrac{e^x - e^{-x}}{e^x + e^{-x}}$
- $\odot$ là phép nhân từng phần,
- W_f, W_i, W_C, W_o là các ma trận trọng số và $\mathbf{b}_f, \mathbf{b}_i, \mathbf{b}_C, \mathbf{b}_o$ là các vector bias.

Sau khi chuỗi vector $\mathbf{x}_i$ được xử lý qua $L_{\max}$ bước thời gian trong mạng LSTM, đầu ra cuối cùng là trạng thái ẩn tại bước thời gian cuối cùng $\mathbf{h}_{L_{\max}}$. Đây là vector đặc trưng tổng hợp toàn bộ thông tin của chuỗi và sẽ được sử dụng làm đầu vào cho các tầng phân loại phía sau nhằm dự đoán xác suất thuộc về các lớp cảm xúc.

4. Phân loại Vector đặc trưng thu được từ LSTM, $\mathbf{h}_{L_{\max}}$, được đưa qua một hoặc nhiều lớp Dense để phân loại. Ví dụ, ta có thể có một lớp Dense với hàm kích hoạt ReLU như sau:

$$\mathbf{z} = \text{ReLU}(W_d \mathbf{h}_{L_{\max}} + \mathbf{b}_d),$$

và sau đó, lớp phân loại cuối cùng với 3 đơn vị tương ứng với 3 lớp (tích cực, trung tính và tiêu cực) sử dụng hàm softmax:

$$\hat{\mathbf{y}} = \text{softmax}(W_o' \mathbf{z} + \mathbf{b}_o'),$$

trong đó hàm softmax được định nghĩa bởi:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}}.$$

3.6.3 Triển khai

- Tokenization: Văn bản được chuyển về chữ thường và tách từ bằng regex; tất cả các từ trong dữ liệu huấn luyện được thu thập để xây dựng từ điển, ánh xạ mỗi từ sang một số nguyên (bắt đầu từ 1, số 0 dành cho padding); các chuỗi số được padding về độ dài cố định.
- Model: Đầu tiên, văn bản được chuyển sang dạng vector 256 chiều. Tiếp đó, thông tin được xử lý qua ba tầng LSTM xếp chồng liên tiếp. Cuối cùng, đặc trưng này được đưa qua một tầng fully-connected với hàm kích hoạt ReLU, rồi qua tầng phân loại cuối cùng với 3 đơn vị để dự đoán ba nhãn cảm xúc.

3.7 Neural Network

3.7.1 Tổng quan

Neural Network (mạng nơ-ron nhân tạo) là một mô hình học máy lấy cảm hứng từ cấu trúc thần kinh sinh học, được thiết kế để học ánh xạ giữa đầu vào và đầu ra thông qua các lớp kết nối. Trong số đó, các mạng lan truyền tiến (feedforward neural networks) là một trong những kiến trúc nền tảng và phổ biến nhất, đặc biệt phù hợp với các bài toán phân loại. Hai kiến trúc tiêu biểu là Single-layer Perceptron và Multilayer Perceptron (MLP). Single-layer Perceptron là dạng đơn giản nhất, chỉ gồm một lớp đầu vào và một lớp đầu ra, phù hợp với các bài toán tuyến tính. Trong khi đó, MLP là một mở rộng với một hoặc nhiều lớp ẩn và hàm kích hoạt phi tuyến, cho phép mô hình học được các quan hệ phi tuyến phức tạp trong dữ liệu. Đặc biệt, MLP đã được chứng minh là một bộ xấp xỉ toàn năng (universal approximator), tức là với cấu trúc đủ lớn, nó có thể xấp xỉ bất kỳ hàm liên tục nào trên miền compact. Điều này khiến MLP trở thành lựa chọn mạnh mẽ và linh hoạt trong nhiều ứng dụng thực tế, đặc biệt là các bài toán phân loại văn bản.

3.7.2 Nguyên lý hoạt động

Multilayer Perceptron (MLP) là một mạng lan truyền tiến gồm nhiều lớp tuyến tính kết hợp với các hàm kích hoạt phi tuyến, cho phép mô hình học được các quan hệ phức tạp giữa đầu vào và đầu ra. Với một đầu vào $\mathbf{x} \in \mathbb{R}^d$, MLP thực hiện ánh xạ qua một chuỗi các phép biến đổi tuyến tính và phi tuyến dưới dạng:

$$\mathbf{h}^{(1)} = \sigma(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}), \quad \mathbf{h}^{(2)} = \sigma(\mathbf{W}^{(2)}\mathbf{h}^{(1)} + \mathbf{b}^{(2)}), \quad \ldots, \quad z = \mathbf{W}^{(L)}\mathbf{h}^{(L-1)} + \mathbf{b}^{(L)}$$

Trong đó, $\sigma(\cdot)$ là hàm kích hoạt phi tuyến (chẳng hạn ReLU), $\mathbf{W}^{(l)}$ và $\mathbf{b}^{(l)}$ lần lượt là trọng số và độ lệch của lớp thứ l, và $\hat{\mathbf{y}}$ là vector đầu ra dạng logits. Trong bài toán phân loại, đầu ra này được đưa vào hàm mất mát `CrossEntropyLoss` để đo độ sai lệch so với nhãn thật.

$$\mathcal{L} = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

Trong đó:

- C là số lượng lớp.
- $y_i \in \{0, 1\}$ là giá trị nhị phân biểu thị nhãn đúng (one-hot encoding) của lớp thứ i.
- $\hat{y}_i$ là xác suất mô hình dự đoán rằng đầu vào thuộc lớp i, thu được bằng cách áp dụng hàm **softmax** lên đầu ra logits:

$$\hat{y}_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)}$$

Quá trình huấn luyện MLP được thực hiện thông qua thuật toán lan truyền ngược (backpropagation) kết hợp tối ưu hoá bằng gradient descent. Mục tiêu là tìm tập trọng số tối ưu sao cho tổng sai số trên toàn bộ tập huấn luyện là nhỏ nhất. Việc sử dụng các hàm kích hoạt phi tuyến tại mỗi lớp ẩn là điều kiện cần thiết để đảm bảo khả năng xấp xỉ các hàm phi tuyến phức tạp trong không gian đầu vào.

Với cấu hình phù hợp, MLP có thể học được các biểu diễn đặc trưng hiệu quả từ văn bản đã được vector hóa, giúp cải thiện hiệu suất trong các tác vụ phân loại.

3.7.3 Triển khai

Chúng tôi đã tiến hành đánh giá hiệu suất của thuật toán này bằng cách thử nghiệm với nhiều cấu hình siêu tham số khác nhau. Cụ thể, văn bản được biểu diễn thông qua ba phương pháp vector hóa phổ biến: TF-IDF, Bag-of-Words (BoW), và biểu diễn nhị phân. Mỗi phương pháp vector hóa được kết hợp với các giá trị khác nhau của số vòng lặp huấn luyện (epoch)

Chapter 4

Kết quả thực nghiệm

4.1 Cài đặt thực nghiệm

Dưới đây là danh sách các cấu hình siêu tham số được sử dụng cho từng phương pháp. Chúng tôi tiến hành thử nghiệm toàn bộ các tổ hợp cấu hình này và lựa chọn ra một cấu hình tối ưu nhất cho mỗi phương pháp dựa trên hiệu suất trên tập kiểm thử. Các phương pháp được đánh giá bao gồm Logistic Regression, K-Nearest Neighbors (k-NN), Random Forest, Support Vector Machines (SVM), Long Short-Term Memory (LSTM), và Neural Networks. Quá trình vector hóa dữ liệu văn bản sử dụng ba phương pháp: TF–IDF, Bag of Words, và Binary Representation (biểu diễn nhị phân), ngoại trừ LSTM sử dụng Index-based Tokenizer như đã đề cập.

- **Logistic Regression**
  - **C (Độ mạnh điều chuẩn):** 0.01, 0.1, 1, 10
  - **solver (Thuật toán tối ưu):** liblinear, lbfgs
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation
- **K-Means Clustering**
  - **n_cluster(Số cụm):** 3
  - **init:** k-means++, random
  - **n_init:** 5, 10, 20
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation
- **K-Nearest Neighbors (k-NN)**
  - **n_neighbors (Số hàng xóm):** 1–100

  - **metric (Độ đo tương đồng):** euclidean, manhattan, cosine
  - **weights (Trọng số lân cận):** uniform, distance
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation

- **Random Forest**
  - **n_estimators (Số lượng cây):** 50, 100, 150, 200
  - **criterion (Tiêu chí chia):** gini, entropy
  - **max_depth (Độ sâu tối đa):** None, 5, 10, 15
  - **min_samples_split:** 2, 5, 10
  - **min_samples_leaf:** 1, 2, 4
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation

- **Support Vector Machines (SVM)**
  - **C (Hệ số điều chỉnh):** 0.1, 1, 10, 100
  - **kernel:** linear, rbf, poly, sigmoid
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation

- **Long Short-Term Memory (LSTM)**
  - **Epochs:** 5, 10, 20, 50

- **Neural Networks**
  - **Epochs:** 5, 10, 20, 50
  - **Vectorization:** TF–IDF, Bag of Words, Binary Representation

4.2 Kết quả thực nghiệm

Để đánh giá hiệu suất của các mô hình phân loại cảm xúc, chúng tôi sử dụng ba chỉ số phổ biến trong lĩnh vực học máy: **Precision**, **Recall** và **F1-score**. Các chỉ số này được xây dựng dựa trên ba thành phần cơ bản: **True Positive (TP)**, tức số mẫu thực sự thuộc một lớp và được mô hình dự đoán đúng; **False Positive (FP)**, là số mẫu không thuộc lớp đó nhưng bị dự đoán nhầm vào; và **False Negative (FN)**, là số mẫu thực sự thuộc lớp nhưng lại bị bỏ sót, dự đoán sai sang lớp khác.

Trên cơ sở đó, **Precision** được định nghĩa là tỷ lệ giữa số lượng dự đoán đúng trên tổng số mẫu mà mô hình dự đoán là thuộc lớp:

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

Chỉ số này phản ánh mức độ "tinh khiết" trong các dự đoán của mô hình cho một lớp cụ thể — precision càng cao thì càng ít có dự đoán sai lệch vào lớp đó.

Recall được tính bằng tỷ lệ giữa số lượng dự đoán đúng trên tổng số mẫu thực sự thuộc lớp:

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

Recall cho thấy khả năng bao phủ của mô hình — recall cao đồng nghĩa với việc mô hình ít bỏ sót các mẫu quan trọng.

Tuy nhiên, trong nhiều trường hợp, Precision và Recall có thể mâu thuẫn với nhau. Vì vậy, **F1-score** được đưa ra như là trung bình điều hòa của hai chỉ số này:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-score là một thước đo cân bằng giữa độ chính xác và khả năng bao phủ, đặc biệt hữu ích trong các tình huống dữ liệu mất cân bằng.

Vì bài toán phân loại cảm xúc là một bài toán đa lớp gồm ba nhãn (Negative, Neutral, Positive), mỗi chỉ số trên sẽ được tính riêng cho từng lớp. Để có cái nhìn tổng thể, chúng tôi sử dụng phương pháp **macro average**, tức là lấy trung bình không trọng số của các chỉ số qua tất cả các lớp. Với $X \in \{\text{Precision}, \text{Recall}, \text{F1}\}$, công thức macro average được biểu diễn như sau:

$$\text{Macro-}X = \frac{1}{C} \sum_{i=1}^{C} X_i$$

trong đó $C = 3$ là số lượng lớp. Macro-average giúp đảm bảo rằng mỗi lớp có ảnh hưởng ngang nhau đến kết quả tổng hợp, bất kể quy mô của lớp đó trong dữ liệu.

Kết quả đánh giá chi tiết cho từng mô hình được thể hiện trong Bảng 4.1.

Table 4.1: Kết quả phân loại của các phương pháp

Model	**Precision**	**Recall**	**F1-score**
Logistic Regression	0.72	0.71	0.71
K-Means-Clustering	0.40	0.45	0.33
K-Nearest Neighbors	0.56	0.52	0.53
Random Forest	0.73	0.72	0.73
Support Vector Machines	0.73	0.70	0.71
LSTM	0.66	0.68	0.67
Neural Networks	0.70	0.70	0.70

4.3 Nhận xét

Qua bảng tổng kết các chỉ số Precision, Recall và F1-score, mô hình Random Forest có chỉ số đánh giá F1-score cao nhất (0.73), tiếp theo là Logistic Regression và Support Vector Machines với F1-score bằng 0.71. Trong bài toán phân loại cảm xúc này, chúng tôi quan tâm đến việc cân bằng giữa Precision và Recall để vừa phát hiện chính xác các cảm xúc, vừa hạn chế bỏ sót. Do đó, chỉ số F1-score được chọn làm tiêu chí đánh giá chính vì đây là chỉ số tổng hợp phản ánh sự cân bằng giữa Precision và Recall.

Bên cạnh đó, do số lượng nhãn cảm xúc trong tập dữ liệu được cân bằng tương đối, nên các giá trị Precision và Recall của các mô hình là khá gần nhau, từ đó việc lựa chọn F1-score làm tiêu chí đánh giá chính là hợp lý.

Từ các kết quả, mô hình Random Forest được xem là phù hợp nhất với mục tiêu phân loại cảm xúc của chúng tôi nhờ khả năng đạt hiệu suất tổng thể tốt nhất. Các mô hình Logistic Regression và SVM cũng thể hiện hiệu quả tương đối, có thể được cân nhắc tùy theo yêu cầu cụ thể về tốc độ và tính đơn giản của mô hình.

Ngoài ra, các mô hình LSTM và Neural Networks tuy có kết quả trung bình nhưng chưa vượt trội so với các mô hình truyền thống, điều này có thể do cấu trúc dữ liệu hoặc cách trích xuất đặc trưng chưa tối ưu cho mô hình sâu.

Một điểm lưu ý là hiệu suất chung của các mô hình vẫn còn ở mức trung bình, với F1-score cao nhất là 0.73. Điều này có thể xuất phát từ các nguyên nhân sau:

- Tính chất đặc thù của tập dữ liệu cảm xúc, khi các biểu cảm có thể mang tính phức tạp và chồng chéo nên việc phân biệt rõ ràng giữa các nhãn là khó khăn.
- Dữ liệu có thể có nhiễu hoặc sự đa dạng lớn trong cách biểu hiện cảm xúc, gây khó khăn cho mô hình trong việc học các đặc trưng hiệu quả.
- Kích thước và chất lượng của tập dữ liệu có thể chưa đủ để các mô hình phức tạp khai thác hết tiềm năng.

Do đó, để cải thiện hiệu quả, nhóm có thể cân nhắc việc làm sạch dữ liệu, tăng kích thước tập huấn luyện, cũng như thử nghiệm các kỹ thuật trích xuất đặc trưng hoặc mô hình sâu nâng cao hơn.

Chapter 5

Kết luận

Trong báo cáo này, chúng tôi đã triển khai và đánh giá hiệu quả của các mô hình học máy và học sâu cho bài toán phân loại cảm xúc văn bản tiếng Việt. Dữ liệu đầu vào là các câu văn thể hiện cảm xúc, được gán nhãn theo ba mức độ: *tích cực*, *trung lập* và *tiêu cực*. Quá trình thực hiện bao gồm các bước chính như tiền xử lý văn bản, xây dựng tokenizer phù hợp với mạng LSTM, trích xuất đặc trưng bằng TF-IDF, BoW, và embedding, sau đó áp dụng các mô hình khác nhau: Logistic Regression, Support Vector Machines, Random Forest, MLP và LSTM.

Kết quả thực nghiệm cho thấy mô hình **Random Forest** đạt hiệu suất tốt nhất với chỉ số **F1-score đạt 0.73**. Các mô hình Logistic Regression và SVM cũng cho kết quả khá ổn định với F1-score khoảng 0.71. Trong khi đó, các mô hình học sâu như LSTM và MLP chưa đạt được hiệu quả cao hơn so với các mô hình truyền thống. Nguyên nhân có thể đến từ kích thước và chất lượng của tập dữ liệu chưa đủ lớn để phát huy được sức mạnh của các mô hình phức tạp hơn.

Trong thời gian tới, để cải thiện hiệu quả phân loại cảm xúc, một số định hướng có thể được xem xét. Trước hết, việc mở rộng và làm sạch tập dữ liệu là rất cần thiết nhằm giảm nhiễu và tăng khả năng học của mô hình. Bên cạnh đó, nhóm có thể tìm hiểu và ứng dụng các mô hình ngôn ngữ tiên tiến hơn như BERT hoặc Transformer, vốn đã được chứng minh hiệu quả vượt trội trong nhiều tác vụ xử lý ngôn ngữ tự nhiên. Ngoài ra, việc kết hợp thêm các đặc trưng ngôn ngữ học như cú pháp, ngữ nghĩa hoặc thông tin từ ngữ cảnh cũng là một hướng đi tiềm năng nhằm nâng cao khả năng phân biệt cảm xúc của mô hình.

Qua quá trình thực hiện, chúng tôi đã hiểu rõ hơn về các bước xây dựng một hệ thống phân loại cảm xúc, từ xử lý dữ liệu đến lựa chọn mô hình và đánh giá hiệu quả. Đây là cơ sở quan trọng cho việc phát triển các ứng dụng xử lý ngôn ngữ tự nhiên trong thực tiễn.