TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA KHOA HỌC MÁY TÍNH

BÁO CÁO ĐỒ ÁN MÔN HỌC
XỬ LÝ NGÔN NGỮ TỰ NHIÊN

CS221.O11

ĐỀ TÀI

PHÂN LOẠI TIÊU ĐỀ VÀ MÔ TẢ CỦA BÀI BÁO

GVHD: Nguyễn Trọng Chỉnh

GVHDTH: Nguyễn Đức Vũ

GVHDTH: Đặng Văn Thìn

Nhóm thực hiện:

1. Hồ Đức Trưởng	21522730
2. Nguyễn Công Nguyên	21521200
3. Nguyễn Phương Tùng	21520524

Thành phố Hồ Chí Minh, Tháng 1 - 2024

Mục lục

Chương1

GIỚI THIỆU BÀI TOÁN

1.1 Bối cảnh và bài toán

Trong thế giới ngày nay, thông tin ngày càng trở nên dồi dào và phức tạp hơn với sự lan rộng của internet. Việc tiếp cận và xử lý lượng lớn thông tin, đặc biệt là các tiêu đề bài báo, đang trở thành một thách thức quan trọng. Internet và các nguồn thông tin trực tuyến ngày càng tăng cường sự hiện diện của tin tức, đòi hỏi khả năng tự động phân loại, sắp xếp thông tin để dễ dàng tìm kiếm và hiểu biết. Việc tự động phân loại các tiêu đề bài báo theo các lĩnh vực khác nhau trở nên cần thiết để giúp người đọc tiếp cận thông tin một cách dễ dàng và nhanh chóng, đồng thời cung cấp cái nhìn tổng quan về đa dạng của thông tin được cung cấp trên internet.

Bài toán phân loại tiêu đề và mô tả của bài báo vào các lĩnh vực tin tức cụ thể là một yếu tố quan trọng trong việc xử lý thông tin. Nó giúp tách riêng các tiêu đề và mô tả của bài báo thành các nhóm nhỏ tương ứng với loại tin tức như Thế giới, Thể thao, Kinh doanh, Khoa học/Công nghệ. Việc này tạo ra sự thuận tiện cho người đọc khi tìm kiếm thông tin cụ thể và đồng thời cung cấp cái nhìn tổng quan về nội dung tin tức đa dạng được cung cấp trên internet.

1.2 Mục tiêu

Mục tiêu chính của đồ án là xây dựng một mô hình thông minh có khả năng tự động phân loại tiêu đề bài báo vào các lĩnh vực tin tức như Thế giới, Thể thao, Kinh doanh, và Khoa học/Công nghệ. Mục đích là tối ưu hóa việc sắp xếp thông tin, mang lại trải nghiệm thuận tiện và nhanh chóng cho người dùng. Đồng thời, mô hình này cũng tạo nền tảng cần thiết để phân tích và tiếp cận thông tin một cách hiệu quả.

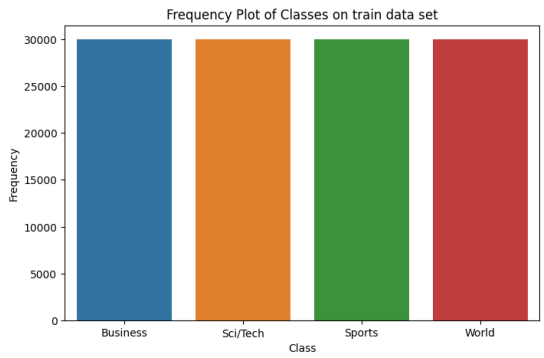
1.3 Ngữ liệu

AG là một bộ sưu tập gồm hơn 1 triệu bài báo. Những bài báo này được thu thập từ hơn 2000 nguồn tin tức khác nhau bởi ComeToMyHead trong hơn 1 năm hoạt động. ComeToMyHead là một công cụ tìm kiếm tin tức học thuật đã hoạt động từ tháng 7 năm 2004.

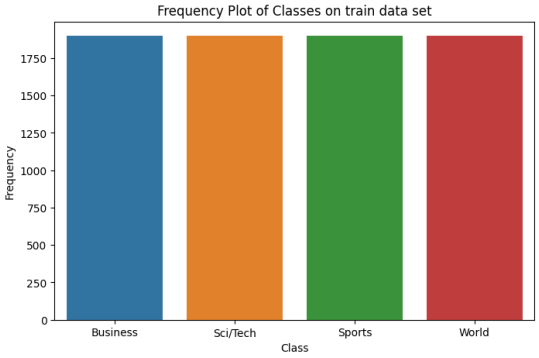
Nguồn: `http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html`

Chúng tôi sử dụng một bộ dữ liệu con từ bộ dữ liệu tin tức AG, bao gồm tiêu đề và mô tả của các bài báo từ 4 loại tin tức lớn nhất. ("Thế giới", "Thể thao", "Kinh doanh", "Khoa học/Công nghệ").

Bộ dữ liệu này bao gồm 30.000 mẫu huấn luyện và 1.900 mẫu kiểm tra cho mỗi loại. Tức trên toàn bộ dữ liệu, số mẫu train là 120.000 và số mẫu test là 7.600.



Hình 1.1: Biểu đồ tần suất của các lớp trên tập dữ liệu huấn luyện (train)



Hình 1.2: Biểu đồ tần suất của các lớp trên tập dữ liệu kiểm tra (test)

Nhận xét: Có thể thấy rằng phân phối số lượng mẫu giữa các lớp trên cả tập dữ liệu huấn luyện và kiểm tra là đồng đều.

Các file train.csv và test.csv đều gồm 3 cột:

- Title: chứa tiêu đề của các bài báo tin tức.
- Description: chứa mô tả về các bài báo tin tức.
- Class Index: bao gồm các ID từ 1 đến 4 tương ứng với một loại tin tức cụ thể: 1-Thế giới, 2-Thể thao, 3-Kinh doanh, 4-Khoa học/Công nghệ.

Hầu hết các thông tin trong tập dữ liệu này đều được viết bằng tiếng Anh.

Trong bộ dữ liệu này, có thể xuất hiện các thẻ HTML và thực thể của HTML như < (hoặc <), > (hoặc >), và & (hoặc &). Những thành phần này không chứa thông tin liên quan đến nội dung của bài báo, mà chúng chỉ tạo thêm nhiễu và không cần thiết trong quá trình phân loại lĩnh vực tin tức.

Ví dụ: Một phần trong phần mô tả của bài báo tin tức kinh doanh có nội dung như sau: `<A HREF="http://www.investor.reuters.com/FullQuote.aspx?ticker=GGP.N target=/stocks/quicki nfo/fullquote">GGP.N</A>`, the No. 2 U.S. shopping mall owner, on Friday said it would buy Rouse for $7.2 billion. Câu này có các thực thể HTML là `<` đại diện cho ký tự "<". `>` đại diện cho ký tự ">". `&` đại diện cho ký tự "&". Và thẻ HTML sau khi giải mã là `\<A HREF="http://www.investor.reuters.com/FullQuote.aspx?ticker=GGP.N target=/stocks/quicki nfo/fullquote">`.

Ngoài ra, còn có thể có chứa các Hyperlink (nhiều trường hợp Hyperlink nằm trong thẻ HTML), chúng cũng giống như các thẻ HTML, chúng không mang lại đóng góp gì vào quá trình phân loại bài báo tin tức.

Các con số xuất hiện trong bộ dữ liệu thường không cung cấp thông tin ngữ cảnh quan trọng mà thường chỉ là các dạng số liệu, ngày tháng, hoặc thông tin chi tiết không ảnh hưởng đáng kể đến việc phân loại lĩnh vực tin tức.

Tính không đồng nhất trong việc sử dụng viết hoa và viết thường có thể gây ra sự mập mờ trong việc nhận diện từ vựng. Việc chuẩn hóa chúng về một dạng cụ thể cần thiết để tăng tính nhất quán và hiệu suất.

Sự không nhất quán của từ ngữ do các từ chưa được rút gọn trong bộ dữ liệu có thể làm giảm độ chính xác của việc phân loại, vì chúng tạo ra sự mơ hồ và phức tạp trong quá trình đánh giá.

Các từ stopword như "and", "are", "a", "at", thường không đóng góp nhiều ý nghĩa cụ thể cho việc phân loại văn bản, chúng thường xuất hiện rất phổ biến mà không có nhiều thông tin quan trọng.

Dấu câu xuất hiện rộng rãi trong bộ dữ liệu. Mặc dù không ảnh hưởng lớn đến ý nghĩa tổng thể của văn bản, việc loại bỏ chúng có thể giúp làm sạch văn bản và tập trung vào thông điệp chính.

1.4 Mô tả bài toán

Input: Tiêu đề và phần mô tả của một bài báo.

Output: Một nhãn thể hiện một trong 4 lĩnh vực tin tức cụ thể: Thế giới, Thể thao, Kinh doanh hoặc Khoa học/Công nghệ.

Ví dụ 1:
Input:
- Title: Stocks Set to Open Slightly Lower at Open
- Description: NEW YORK - U.S. stocks are seen slightly lower at the open Thursday as investors take some cash off the table after Wednesday's sharp rally...
Output: 1 (tương ứng với loại tin tức "Thế giới")

Giải thích: Dự đoán lĩnh vực "Thế giới" dựa trên các từ khóa như "U.S. stocks," "NEW YORK," "investors," và "cash off the table" trong tiêu đề và mô tả. Các từ này thường liên quan đến diễn biến thị trường, tài chính toàn cầu, cho thấy mối quan tâm về diễn biến kinh tế quốc tế, từ đó, dẫn đến việc phân loại vào lĩnh vực "Thế giới".

Ví dụ 2:
Input:
- Title: India weightlifter tests positive
- Description: An Indian weightlifter has tested positive for drugs before the start of the Athens Games, the International Olympic Committee has announced.
Output: 2 (tương ứng với loại tin tức "Thể thao")

Giải thích: Dự đoán lĩnh vực "Thể thao" dựa trên các từ khóa như "India weightlifter," "Athens Games," "International Olympic Committee," và "tested positive for drugs". Các từ này thường liên quan đến vấn đề doping trong thể thao, cho thấy tiêu đề và mô tả liên quan đến sự kiện thể thao quốc tế, do đó được phân loại vào lĩnh vực "Thể thao".

Ví dụ 3:
Input:
- Title: A broken record: Oil up again
- Description: New market set as demand in China, India intensifies supply worries, Iraq lends price support. NEW YORK (CNN/Money) - Oil prices blazed past $47.50 a barrel and set fresh highs early Thursday as demand growth in China and India exacerbated supply shortage ...
Output: 3 (Tương ứng với loại tin tức "Kinh doanh")

Giải thích: Dự đoán lĩnh vực "Kinh doanh" dựa trên các từ khóa như "Oil," "demand in China, India," "supply worries," và "price support". Vì các từ này cho thấy mối liên quan đến tình hình thị trường dầu, nhu cầu và cung ứng, thường là các yếu tố ảnh hưởng đến lĩnh vực kinh doanh. Vì vậy, tiêu đề và mô tả được dự đoán thuộc lĩnh vực "Kinh doanh".

Ví dụ 4:
Input:
- Title: Toshiba, Memory-Tech Develop New DVD (AP)
- Description: AP - Two Japanese companies said Tuesday they have developed a DVD that can play on both existing machines and the upcoming high-definition players, raising hopes for a smooth transition as more people dump old TV sets for better screens.
Output: 4 (Tương ứng với loại tin tức "Khoa học/Công nghệ")

Giải thích: Dự đoán lĩnh vực "Khoa học/Công nghệ" dựa trên các từ khóa như "Toshiba," "Memory-Tech," "developed a DVD," "existing machines," "high-definition players," và "dump old TV sets for better screens". Các từ này thường liên quan đến các sản phẩm công nghệ, phát triển thiết bị điện tử và công nghệ thông tin. Do đó, tiêu đề và mô tả của bài báo được dự đoán thuộc lĩnh vực "Khoa học/Công nghệ".

1.5 Hướng tiếp cận

Chúng tôi áp dụng nhiều kỹ thuật khác nhau trong việc xử lý dữ liệu văn bản như BOW (Bag of Words), Word Embedding sử dụng các phương pháp như word2vec.

Đối với việc xây dựng mô hình máy học, chúng tôi sử dụng một loạt các thuật toán như Naïve Bayes, Logistic Regression, Multi-Layer Perceptron,...

Chương2

PHƯƠNG PHÁP

2.1 Tiền xử lý

Trước khi bắt đầu quá trình xử lý dữ liệu, chúng tôi quyết định kết hợp hai đặc trưng "Title" và "Description" thành một đặc trưng duy nhất. Việc này giúp chúng tôi xử lý dữ liệu một cách nhất quán và liên tục từ đầu đến cuối quá trình tiền xử lý.

Chúng tôi sẽ sử dụng một ví dụ trong suốt quá trình tiền xử lý, đây là một ví dụ về phần mô tả của một bài báo thuộc loại tin tức "Kinh doanh":

> " This QUARTER's EARNINGS report EXCEEDED expectations with a 30 percent INCREASE in revenue compared to LAST YEAR. The company's innovative marketing STRATEGY, outlined HERE at https://www.example.com"> THIS LINK, contributed significantly to this SUCCESS. Our STOCK PRICE ROSE sharply following the ANNOUNCEMENT. Visit OUR WEBSITE here for more DETAILS. "

2.1.1 Giải mã và loại bỏ các thẻ HTML

Tập dữ liệu của chúng tôi bao gồm các đoạn văn chứa mã HTML. Đầu tiên, chúng tôi loại bỏ các phần mã này. Các phần mã này không có ích cho việc phân loại hay cung cấp thông tin quan trọng. Việc loại bỏ chúng giúp làm sạch dữ liệu và tập trung vào những thông tin quan trọng hơn, giúp việc hiểu và phân loại dễ dàng hơn.

Sau đó, chúng tôi giải mã các phần mã HTML trong văn bản, chuyển đổi chúng trở lại thành dạng văn bản thông thường. Tiếp theo, chúng tôi loại bỏ hoàn toàn các phần mã HTML bằng cách loại bỏ các chuỗi nằm trong dấu <> trong văn bản.

Để thực hiện việc loại bỏ liên kết web và thẻ HTML, chúng tôi sử dụng thư viện `html` và `re` trong Python.

> This QUARTER's EARNINGS report EXCEEDED expectations with a 30 percent INCREASE in revenue compared to LAST YEAR. The company's innovative marketing STRATEGY, outlined HERE at https://www.example.com"> THIS LINK, contributed significantly to this SUCCESS. Our STOCK PRICE ROSE sharply following the ANNOUNCEMENT. Visit OUR WEBSITE here for more DETAILS.

2.1.2 Loại bỏ Hyperlink

Loại bỏ các Hyperlink là một phần của quá trình làm sạch dữ liệu trước khi phân loại. Tương tự như các thẻ HTML, các liên kết web không cung cấp thông tin quan trọng cho quá trình phân loại. Chúng tôi thực hiện việc loại bỏ chúng để tập trung vào các nội dung chính, giúp quá trình phân loại trở nên đơn giản và chính xác hơn.

Để loại bỏ các Hyperlink, chúng tôi sử dụng thư viện 're' kết hợp với biểu thức chính quy (regular expression) trong Python.

Ví dụ: Kết quả sau khi loại bỏ các Hyperlink:

> This QUARTER's EARNINGS report EXCEEDED expectations with a 30 percent INCREASE in revenue compared to LAST YEAR. The company's innovative marketing STRATEGY, outlined HERE at THIS LINK, contributed significantly to this SUCCESS. Our STOCK PRICE ROSE sharply following the ANNOUNCEMENT. Visit OUR WEBSITE here for more DETAILS.

2.1.3 Loại bỏ các con số

Việc xóa bỏ các con số trong quá trình phân loại tiêu đề và mô tả của bài báo giúp tập trung vào thông tin nội dung văn bản quan trọng hơn. Các con số thường không mang lại thông tin ngữ cảnh, mà thường chỉ là dạng số liệu, ngày tháng, hay thông tin cụ thể không cần thiết đối với việc phân loại lĩnh vực tin tức.

Khi loại bỏ các con số, chúng tôi nhằm làm sạch dữ liệu để tập trung vào những từ ngữ, cụm từ, hoặc ngữ cảnh chính trong tiêu đề và mô tả, từ đó giúp mô hình phân loại tập trung vào các yếu tố quan trọng hơn để hiểu nội dung và xác định lĩnh vực tin tức một cách chính xác.

Để loại bỏ các Hyperlink, chúng tôi sử dụng thư viện 're' kết hợp với biểu thức chính quy (regular expression) trong Python.

Ví dụ: Kết quả sau khi loại bỏ các con số:

> This QUARTER's EARNINGS report EXCEEDED expectations with a percent INCREASE in revenue compared to LAST YEAR. The company's innovative marketing STRATEGY, outlined HERE at THIS LINK, contributed significantly to this SUCCESS. Our STOCK PRICE ROSE sharply following the ANNOUNCEMENT. Visit OUR WEBSITE here for more DETAILS.

2.1.4 Lowercase

Bước Lowercase là quá trình chuyển đổi tất cả các ký tự trong văn bản về dạng chữ thường. Quá trình này giúp làm sạch và chuẩn hóa dữ liệu văn bản trước khi tiến hành các bước xử lý và phân loại.

Việc chuyển đổi văn bản về dạng chữ thường có ích trong việc loại bỏ sự phân biệt về viết hoa và viết thường giữa các từ. Điều này giúp mô hình học máy hoặc các thuật toán xử lý ngôn ngữ tự nhiên không phân biệt các từ giống nhau chỉ khác nhau ở viết hoa và viết thường. Kỹ thuật này cũng giúp giảm kích thước của từ vựng, giảm sự phức tạp của dữ liệu và tăng tính nhất quán trong quá trình phân loại.

Sử dụng phương thức lower() để chuyển đổi văn bản thành dạng chữ thường.

Ví dụ: Kết quả sau khi thực hiện bước Lowercase:

> this quarter's earnings report exceeded expectations with a percent increase in revenue compared to last year. the company's innovative marketing strategy, outlined here at this link, contributed significantly to this success. our stock price rose sharply following the announcement. visit our website here for more details.

2.1.5 Tokenize

Bước tiền xử lý Tokenization là quá trình chia nhỏ văn bản thành các đơn vị nhỏ hơn gọi là "token". Các token có thể là từ, cụm từ, hoặc ký tự đơn nhỏ nhất có ý nghĩa trong văn bản. Quá trình này giúp tách dữ liệu văn bản thành các phần nhỏ hơn để dễ dàng xử lý và phân tích.

Trong bài toán phân loại tiêu đề và mô tả bài báo, Tokenization có ích trong việc biến đổi văn bản thành các từ hoặc cụm từ riêng biệt, từ đó tạo thành các "đơn vị ý nghĩa" để mô hình máy học có thể hiểu và phân tích. Việc này giúp xây dựng từ vựng, chuẩn bị dữ liệu cho các bước xử lý ngôn ngữ tự nhiên như vector hóa hoặc xử lý ngữ cảnh để có thể áp dụng các thuật toán phân loại và dự đoán chính xác lĩnh vực tin tức từ tiêu đề và mô tả.

Dòng mã này sử dụng phương thức findall() từ thư viện re để tìm và trích xuất các từ hoặc dấu câu từ văn bản đã cho.

Ví dụ: Kết quả sau khi thực hiện bước Tokenize:

['this', 'quarter', 's', 'earnings', 'report', 'exceeded', 'expectations', 'with', 'a', 'percent', 'increase', 'in', 'revenue', 'compared', 'to', 'last', 'year', '.', 'the', 'company', 's', 'innovative', 'marketing', 'strategy', ',', 'outlined', 'here', 'at', 'this', 'link', ',', 'contributed', 'significantly', 'to', 'this', 'success', '.', 'our', 'stock', 'price', 'rose', 'sharply', 'following', 'the', 'announcement', '.', 'visit', 'our', 'website', 'here', 'for', 'more', 'details', '.']

2.1.6 Lemmatization

Bước Lemmatization là quá trình chuẩn hóa một số từ về dạng từ gốc, gọi là "lemma". Lemmatization giúp chuyển đổi một số từ về dạng chuẩn để tạo ra các từ gốc chung có ý nghĩa, giúp giảm thiểu sự biến đổi và tăng tính nhất quán trong dữ liệu.

Trong bài toán phân loại tiêu đề và mô tả bài báo, Lemmatization hữu ích để chuẩn hóa một số từ về dạng gốc chung, từ đó giúp mô hình học máy hiểu được các từ có cùng nguồn gốc mặc dù viết tắt, dạng số nhiều, thì hiện tại hoặc quá khứ. Việc này giúp làm sạch dữ liệu và tăng khả năng phân tích văn bản, giúp mô hình hiểu được nội dung một cách chính xác hơn và tạo ra dự đoán tốt hơn khi phân loại lĩnh vực tin tức từ tiêu đề và mô tả của bài báo.

Bước Lemmatization sử dụng thư viện WordNetLemmatizer để chuẩn hóa một số từ về dạng từ gốc.

2.1.7 Loại bỏ stopword

Stopword là những từ phổ biến, thông thường không mang lại nhiều ý nghĩa hoặc không cần thiết trong việc hiểu ý nghĩa của một đoạn văn. Các từ như "and", "the", "is", "of" thường được coi là stopwords.

Khi loại bỏ stop words trong bài toán phân loại, chúng giúp làm sạch dữ liệu, tập trung vào các từ khóa quan trọng hơn để nắm bắt ý nghĩa và tính chất phân loại của văn bản. Việc này cũng giúp giảm kích thước của tập dữ liệu, cải thiện hiệu suất và giảm thiểu ảnh hưởng của các từ không quan trọng đối với việc phân loại.

Chúng tôi loại bỏ các từ stopword bằng cách sử dụng danh sách từ stopwords có sẵn trong thư viện xử lý ngôn ngữ tự nhiên như NLTK (Natural Language Toolkit). Đầu tiên, chúng tôi tải danh sách từ stopwords cho ngôn ngữ cụ thể (ví dụ: tiếng Anh) từ thư viện. Sau đó, chúng tôi lặp qua các từ trong danh sách và loại bỏ những từ stopword này.

Ví dụ: Sau khi loại bỏ stopwords, danh sách từ đã thay đổi. Các từ stopwords thường bị loại bỏ để tập trung vào các từ quan trọng hơn trong văn bản. Có thể loại bỏ các từ như "this", "with", "a", "in", "to", "the", "at", "following", "our", "for" vì chúng thường là các từ phổ biến và không cần thiết trong việc hiểu ý nghĩa của đoạn văn.

Danh sách sau khi loại bỏ stopword:

['quarter', 'earnings', 'report', 'exceeded', 'expectation', 'percent', 'increase', 'revenue', 'compared', 'last', 'year', '.', 'company', 'innovative', 'marketing', 'strategy', ',', 'outlined', 'link', ',', 'contributed', 'significantly', 'success', '.', 'stock', 'price', 'rose', 'sharply', 'following', 'announcement', '.', 'visit', 'website', 'detail', '.']

2.1.8 Loại bỏ dấu câu

Các dấu câu (punctuation) bao gồm các ký tự như dấu chấm, dấu phẩy, dấu chấm hỏi, dấu ngoặc, dấu gạch ngang, và các ký tự tương tự. Chúng được sử dụng để đánh dấu và phân cách các câu, từ hoặc đoạn văn trong văn bản.

Trong bài toán phân loại văn bản, loại bỏ các dấu câu có thể giúp tập trung vào ý nghĩa của từ và đoạn văn, giúp mô hình học máy không bị ảnh hưởng bởi các ký tự không mang ý nghĩa ngôn ngữ mà tập trung vào các từ và cấu trúc văn phạm. Điều này giúp cải thiện hiệu suất và chất lượng của mô hình trong quá trình phân loại.

Để loại bỏ các dấu câu, chúng tôi sử dụng thư viện chuẩn có sẵn trong python là string. Chúng tôi kiểm tra từng từ trong văn bản nếu không nằm trong danh sách các dấu câu sẽ giữ lại, ngược lại sẽ loại bỏ. *Ví dụ:* Kết quả sau khi loại bỏ dấu câu là:

['quarter', 'earnings', 'report', 'exceeded', 'expectation', 'percent', 'increase', 'revenue', 'compared', 'last', 'year', 'company', 'innovative', 'marketing', 'strategy', 'outlined', 'link', 'contributed', 'significantly', 'success', 'stock', 'price', 'rose', 'sharply', 'following', 'announcement', 'visit', 'website', 'detail']

2.2 Trích xuất từ vựng

Quá trình trích xuất từ vựng (Vocabulary) là bước quan trọng xác định danh sách từng từ có mặt trong tập dữ liệu. Điều này giúp mô hình hiểu và xử lý văn bản một cách chính xác. Mỗi từ được lựa chọn và thêm vào danh sách từ vựng mà mô hình sẽ sử dụng khi tạo các vector từ.

Điểm đặc biệt là chúng tôi quyết định chỉ sử dụng các từ xuất hiện ít nhất 5 lần trong tập dữ liệu. Việc này loại bỏ các từ ít quan trọng, giúp tinh chỉnh Vocabulary để tập trung vào các từ có ý nghĩa và quan trọng trong dữ liệu huấn luyện. Nhờ điều này, mô hình có thể nắm bắt và hiểu ngữ cảnh của văn bản một cách tốt hơn, từ đó tăng cường khả năng xử lý và hiểu văn bản một cách chính xác.

Ví dụ: Phần mô tả trong bài báo thuộc lĩnh vực Kinh doanh là: Unions representing workers at Turner Newall say they are 'disappointed' after talks with stricken parent firm Federal Mogul. Sau khi thực hiện trích xuất từ vựng thì kết quả thu được (trong ví dụ này giả sử tất cả các từ đều xuất hiện ít nhất 5 lần):

["Unions", "representing", "workers", "at", "Turner", "Newall", "say", "they", "'disappointed'", "after", "talks", "with", "stricken", "parent", "firm", "Federal", "Mogul"]

2.3 Trích xuất feature

2.3.1 Bag of Words(BoW)

Bag-of-words, viết tắt là BoW, có nghĩa là túi từ. Theo phương pháp bag-of-word chúng ta sẽ mã hoá các từ trong câu thành một vector có độ dài bằng số lượng các từ trong từ điển (tức là bằng với kích thước của bộ từ điển) và đếm tần suất xuất hiện của các từ. Tần suất của từ thứ i trong từ điển sẽ chính bằng phần tử thứ i trong véc tơ.

I have a great AI book
I have to read twice times

I	2
AI	1
a	1
about	1
book	1
deep	0
great	1
is	0
this	0
machine	0
learning	0
have	2
to	1
read	1
twice	1
times	1

*Hình 2.1: Văn bản ở bên trái được mã hoá thành véc tơ tần suất từ ở bên phải. Các từ **'I'** và **'have'** lặp lại 2 lần nên có tần suất là 2. Những từ không xuất hiện trong câu nhưng có trong từ điển như **'deep', 'is', 'this', 'machine', 'learning'** thì có giá trị là 0.*

Như vậy theo phương pháp bag-of-words thì mỗi từ sẽ trở thành một chiều biểu diễn trong không gian của véc tơ đầu ra. Khi số lượng các từ rất lớn thì kết quả mã hoá có thể tạo thành một vector có độ dài rất lớn. Thông thường đây sẽ là một véc tơ thưa thớt (sparse vector) có hầu hết các giá trị bằng 0. Số lượng chiều lớn khiến việc biểu diễn các vector mã hoá trên không gian gặp khó khăn. Việc lưu trữ này không hiệu quả mà còn làm tăng độ phức tạp của các mô hình máy học gây mất thời gian để huấn luyện

và đưa ra các dự đoán. Để hạn chế vấn đề này, chúng tôi đã xây dựng bộ từ vựng với các từ xuất hiện ít nhất 5 lần trong tập dữ liệu.

Nhược điểm lớn nhất của BoW là nó không mang thông tin về thứ tự của các từ và nghĩa thực của mỗi từ và độ tương tự giữa các từ khác nhau.

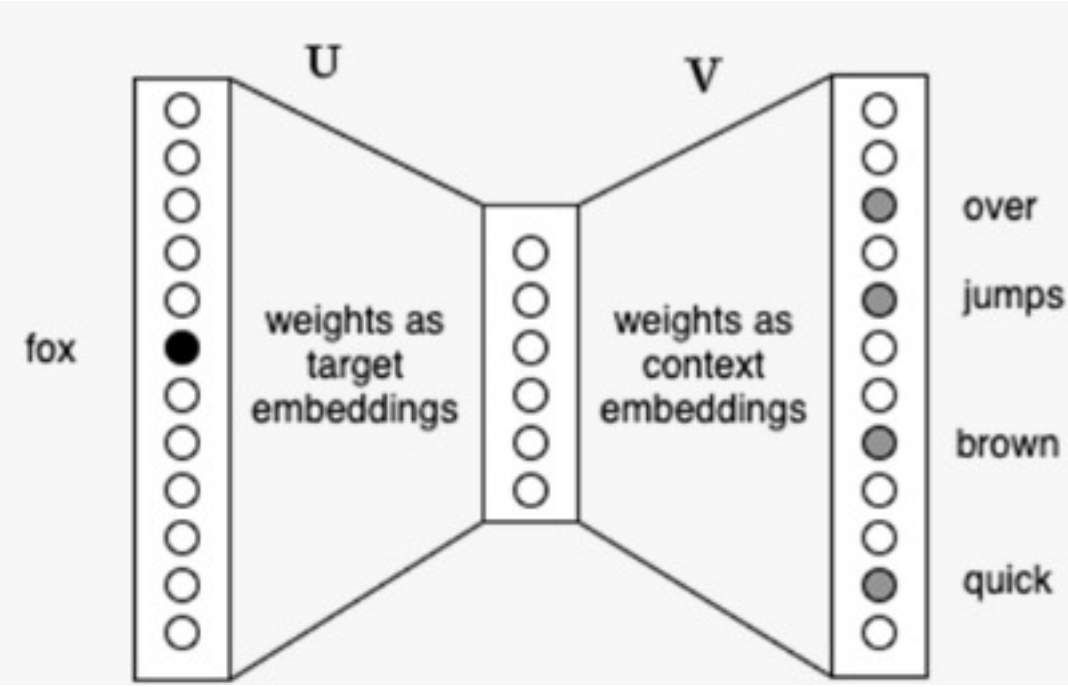
Các record sẽ được biểu diễn thông qua BoW và sẽ là đầu vào của mô hình Naïve Bayes.

2.3.2 Word2vec -Skip Gram model

Mô hình Skip-gram nhằm mục đích dự đoán các từ xung quanh (context words) dựa trên một từ đặc trưng (target word). Ý tưởng chính của Skip-gram là từ mục tiêu (target word) sẽ được đưa vào mô hình để dự đoán các từ xung quanh. Mỗi từ trong từ điển sẽ được biểu diễn dưới dạng vector đặc trưng trong không gian nhiều chiều. Mục tiêu là học được biểu diễn vector sao cho các từ có ý nghĩa tương tự sẽ có biểu diễn gần nhau, độ tương đồng được tính toán bằng Cosine Similarity.

$$Similarity = \cos(\theta) = \frac{w_1 \cdot w_2}{\|w_1\|\|w_2\|}$$

Skip-gram word2vec là một mạng neural vô cùng đơn giản với chỉ một tầng ẩn không có hàm kích hoạt.



Hình 2.2: Minh họa Skip-gram dưới dạng mạng neural.

Ngoài Skip-gram thì Word2vec còn có thể được xây dựng bằng Continuous Bag of Words model.

Sau khi huấn luyện xong, chúng ta có thể sử dụng vector từ đã học để biểu diễn từ bất kỳ trong từ điển.

Trong đồ án này, chúng tôi đã sử dụng mô hình 300-dimension Word2vec đã được pretrained trên tập dữ liệu Google News dataset (khoảng 100 tỷ từ). Các record sẽ được biểu diễn thông qua Word2vec và sẽ là đầu vào của các classifiers như Logistic Regression, Multi-layer Perceptron,...

2.4 Modeling

Chúng tôi sẽ sử dụng mô hình học máy bao gồm Naive Bayes, Logistic Regression, MLP (Multi-Layer Perceptron), và một số classifiers khác chúng tôi sẽ triển khai sau này nữa.

2.4.1 Naive Bayes

Trong bài toán phân loại đa lớp, Naive Bayes được gọi là Naive Bayes đa thức (Multinomial Naive Bayes). Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng **Bags of**

Words. Thuật toán này được sử dụng để phân loại các đối tượng vào nhiều lớp khác nhau dựa trên các đặc trưng của chúng. Để sử dụng Naive Bayes đa thức, ta cần tính xác suất của mỗi lớp dựa trên các đặc trưng của đối tượng đó.

Như vậy, đầu vào của mô hình là một vector đặc trưng biểu diễn một văn bản, trong đó mỗi phần tử của vector đại diện cho số lần xuất hiện của từ tương ứng trong văn bản. Đầu ra của mô hình Naive Bayes đa thức trong bài toán phân loại văn bản với Bags of Words là một dự đoán cho lớp của văn bản. Cụ thể, mô hình sẽ xuất ra xác suất cho mỗi lớp (thế giới, thể thao, kinh doanh, khoa học/công nghệ). Đầu ra cuối cùng sẽ là lớp được dự đoán cho văn bản đó, là lớp có xác suất cao nhất.

Quá trình biến đổi từ đầu vào Bags of Words thành đầu ra (dự đoán lớp) trong mô hình Naive Bayes đa thức bao gồm các bước cụ thể. Đầu tiên, ta tính xác suất xuất hiện của từng đặc trưng trong từng lớp, đánh giá mức độ quan trọng của từ đối với mỗi lớp. Tiếp theo, sử dụng xác suất của các đặc trưng và giả định độc lập, ta tính xác suất của mỗi lớp cho một văn bản thông qua công thức Bayes. Cuối cùng, dựa vào xác suất của từng lớp, mô hình chọn lớp có xác suất cao nhất là lớp dự đoán cho văn bản đó. Tổng cộng, quá trình này giúp mô hình đưa ra dự đoán chính xác về lớp của mỗi văn bản dựa trên xác suất tính toán từ các đặc trưng của chúng.

Công thức Bayes được sử dụng để tính xác suất của một lớp dựa trên xác suất của các đặc trưng của đối tượng đó. Giả sử chúng ta đang xử lý bài toán phân loại với C lớp, và giả sử có một điểm dữ liệu $x \in R^d$. Xác suất này được tính bằng công thức sau $p(c|x)$, tức là xác suất để đầu ra là class c biết đầu vào là vector x. Biểu thức này, nếu tính được, sẽ giúp chúng ta xác định được xác suất để điểm dữ liệu rơi vào mỗi class. Từ đó có thể giúp xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:

$$c = \underset{c}{\operatorname{argmax}}\, p(c|x) = \underset{c}{\operatorname{argmax}} \frac{p(x|c) \cdot p(c)}{p(x)} = \underset{c}{\operatorname{argmax}}\, p(x|c) \cdot p(c)$$

Naive Bayes đa thức giả định rằng các đặc trưng của đối tượng đó độc lập với nhau. Tức là, xác suất của một đặc trưng không phụ thuộc vào các đặc trưng khác. Giả định này giúp cho việc tính toán xác suất trở nên đơn giản hơn. Tức là:

$$p(x|c) = p(x_1, x_2, \ldots, x_d|c) = \prod_{i=1}^{d} p(x_i|c)$$

Ở bước training, các phân phối $p(c)$ và $p(x_i|c), i = 1, ..., d$ sẽ được xác định dựa vào training data. Cụ thể:

$$p(c) = \frac{\text{số mẫu tiên nghiệm thuộc lớp } c}{\text{tổng số lượng mẫu huấn luyện}}$$

$$\lambda_{ci} = p(x_i|c) = \frac{N_{ci}}{N_c}$$

Trong đó:

N_{ci} = Tổng số lần từ thứ i xuất hiện trong các văn bản của class c,

được tính là tổng của các thành phần thứ i của các feature vectors ứng với class c.

N_c = Tổng số từ (bao gồm cả lặp) xuất hiện trong class c.

Nói cách khác, nó bằng tổng độ dài của toàn bộ các văn bản thuộc class c.

Có thể suy ra rằng $N_c = \sum_{i=1}^{d} N_{ci}$, từ đó $\sum_{i=1}^{d} \lambda_{ci} = 1$.

Cách tính này có một hạn chế là nếu có một từ chưa bao giờ xuất hiện trong class c thì $p(x_i|c)$ sẽ bằng 0. Điều này sẽ dẫn đến $p(x|c)$ bằng 0 bất kể các giá trị còn lại lớn thế nào, dẫn đến kết quả không chính xác. Để giải quyết ta sẽ dùng kĩ thuật **Laplace smoothing**: $\hat{\lambda}_{ci} = \frac{N_{ci}+\alpha}{N_c+d\alpha}$

Với α là một số dương, thường bằng 1, để tránh trường hợp tử số bằng 0, mẫu số được cộng với $d\alpha$ để đảm bảo tổng xác suất: $\sum_{i=1}^{d} \hat{\lambda}_{ci} = 1$

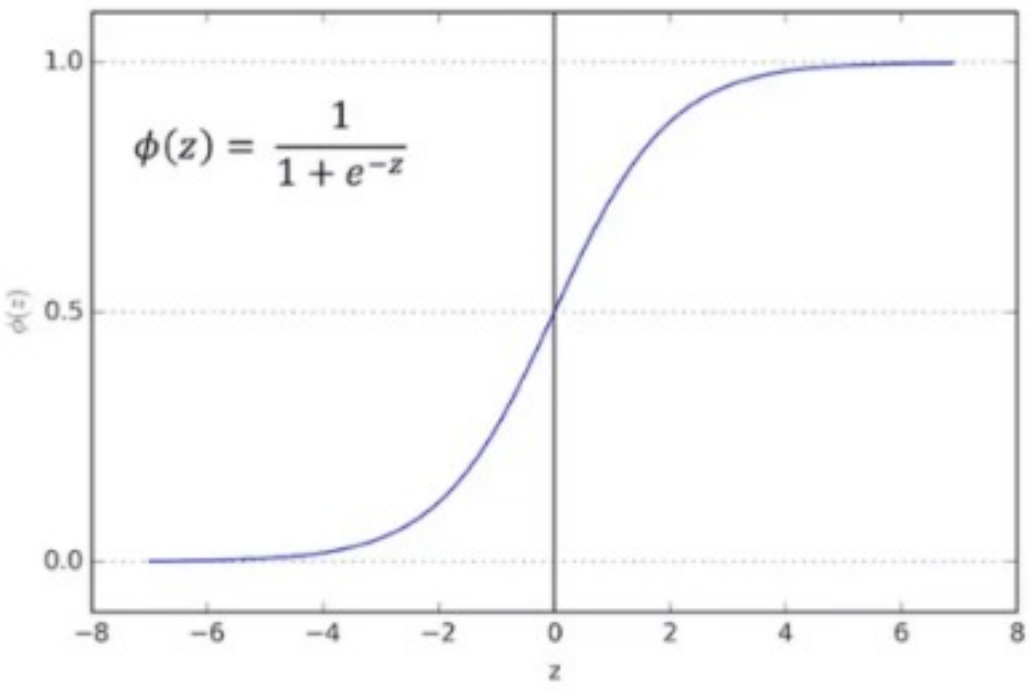
Như vậy, mỗi class c sẽ được mô tả bởi bộ các số dương có tổng bằng 1: $\hat{\lambda}_c = \{\hat{\lambda}_{c1}, \ldots, \hat{\lambda}_{cd}\}$.

Ở bước **test**, với một số điểm dữ liệu mới $\mathbf{x}$, class của nó sẽ được xác định bởi:

$$c = \underset{c \in \{1,\ldots,C\}}{\operatorname{argmax}}\, p(c) \prod_{i=1}^{d} p(x_i|c)$$

2.4.2 Logistic Regression

Hồi quy Logistic sử dụng hàm logistic (hoặc hàm sigmoid) để chuyển đổi đầu ra thành giá trị nằm trong khoảng từ 0 đến 1. Hàm sigmoid được định nghĩa bởi $\phi(z) = \frac{1}{1+e^{-z}}$, trong đó z là một tổ hợp tuyến tính của các đặc trưng.



Hình 2.3: Độ thị hàm sigmoid

Hàm mất mát: Cross-Entropy Loss dùng để đo sự khác biệt giữa kết quả dự đoán so với giá trị thực tế

$$\text{logloss} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=0}^{C-1}[y_i = c] \cdot \log(\hat{p}_c(X_i))$$

Trong đó:

- N là số điểm dữ liệu hoặc số lượng mẫu.
- C là số lượng lớp hoặc số lượng phân loại.

Trong đó, $y \in \{1, ..., C\}$ là giá trị thực, còn còn $\hat{p}_c(X_i)$ là giá trị dự đoán thông qua hàm Softmax, được xác định bằng công thức:

$$P(y_i = c|X_i) = \hat{p}_c(X_i) = \frac{e^{X_i \cdot W_c + W_{0,c}}}{\sum_{l=0}^{C-1} e^{X_i \cdot W_l + W_{0,l}}}$$

Mục tiêu: Tìm giá trị nhỏ nhất của hàm loss

Tìm w tối ưu để hàm loss nhỏ nhất

Để tối ưu hóa người ta thường sử dụng Gradient Descent trong mỗi vòng lặp các tham số được cập nhật cho đến khi đạo hàm tiến về 0.

Quá trình tối ưu tham số sẽ dừng lại cho đến khi hàm Loss hội tụ hoặc đạt đến số lần lặp được đặt trước.

Sau khi kết thúc quá trình training để tìm giá trị w_i và b tối ưu. Mô hình sẽ tính toán n giá trị xác suất (bằng với số lượng classes). Giá trị dự đoán $P(y = k)$ là xác suất của dữ liệu thuộc lớp k. Sau đó, mô hình sử dụng Sofmax để tìm lớp có xác suất cao nhất, nhãn dự đoán sẽ tương ứng với lớp đó.

2.4.3 MLP- Multi-Layer Perceptron

Mạng nơ-ron đa tầng (Multi-layer Perceptron - MLP) là một dạng phổ biến của mạng nơ-ron nhân tạo, được xây dựng trên cơ sở mô hình perceptron. MLP thường bao gồm ít nhất ba tầng: tầng đầu vào, tầng ẩn và tầng đầu ra.

- **Tầng đầu vào (Input Layer)**: Nhận giá trị đầu vào và truyền chúng đến tầng ẩn. Mỗi nút trong tầng này đại diện cho một đặc trưng của dữ liệu đầu vào.
- **Tầng ẩn (Hidden Layer)**: Chứa các nút nơ-ron ẩn, thực hiện các phép tính trung gian dựa trên trọng số và hàm kích thích (activation function). Mỗi nút trong tầng ẩn nhận giá trị từ tầng đầu vào và tạo ra đầu ra cho tầng đầu ra.
- **Tầng đầu ra (Output Layer)**: Tạo ra đầu ra của mô hình dự đoán. Mỗi nút trong tầng này thường tương ứng với một lớp đối tượng hoặc một giá trị đầu ra.

Mỗi output của một unit (trừ các input units) được tính dựa vào công thức:

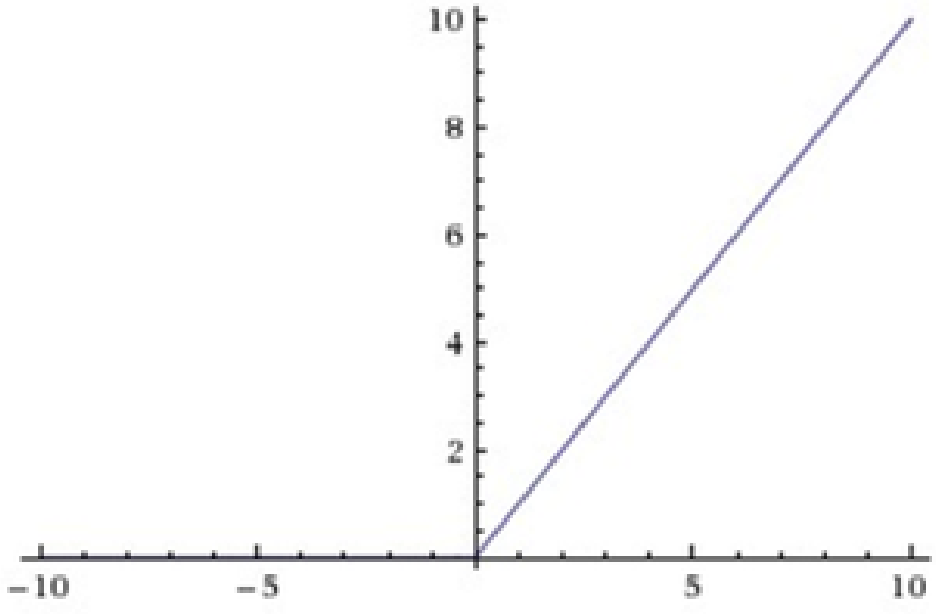
$$a_i^{(l)} = f(w_i^{(l)T} \cdot a^{(l-1)} + b_i^{(l)})$$

Trong đó $f(.)$ là một (nonlinear) activation function. Ở dạng vector, biểu thức bên trên được viết là:

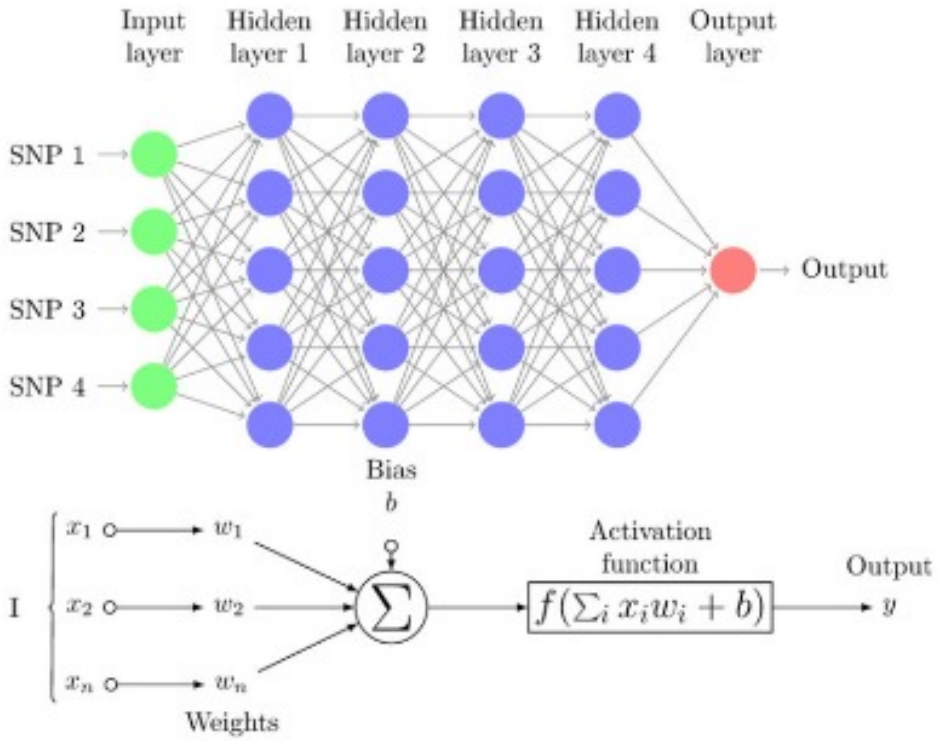
$$a^{(l)} = f(W^{(l)T} \cdot a^{(l-1)} + b^{(l)})$$

Trước đây, sigmoid và tanh là các hàm kích hoạt thường được sử dụng. Tuy nhiên, những năm gần đây hàm ReLU (Rectified Linear Unit) được sử dụng rộng rãi vì tính đơn giản, giúp giảm chi phí tính toán, giúp mô hình hội tụ nhanh hơn và giảm nguy cơ vanishing gradient so với sigmoid và tanh.

Công thức toán học: $f(s) = \max(0, s)$



Hình 2.4: Đồ thị hàm ReLU



Hình 2.5: Minh họa: Sơ đồ mạng Multi-Layer Perceptron (MLP) với bốn tầng ẩn

```
Model: "multilayer_perceptron"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               154112

 dense_1 (Dense)             (None, 256)               131328

 dense_2 (Dense)             (None, 128)               32896

 dense_3 (Dense)             (None, 4)                 516

=================================================================
Total params: 318852 (1.22 MB)
Trainable params: 318852 (1.22 MB)
Non-trainable params: 0 (0.00 Byte)
```

Hình 2.6: Cấu hình của MLP

Quá trình học của MLP thường dựa trên thuật toán lan truyền ngược (backpropagation). Trong quá trình huấn luyện, mô hình điều chỉnh các trọng số để giảm sai số giữa đầu ra thực tế và đầu ra dự đoán. Điều này thường được thực hiện thông qua quá trình tối ưu hóa hàm mất mát (Cross-entropy loss) bằng các phương pháp như Gradient Descent.

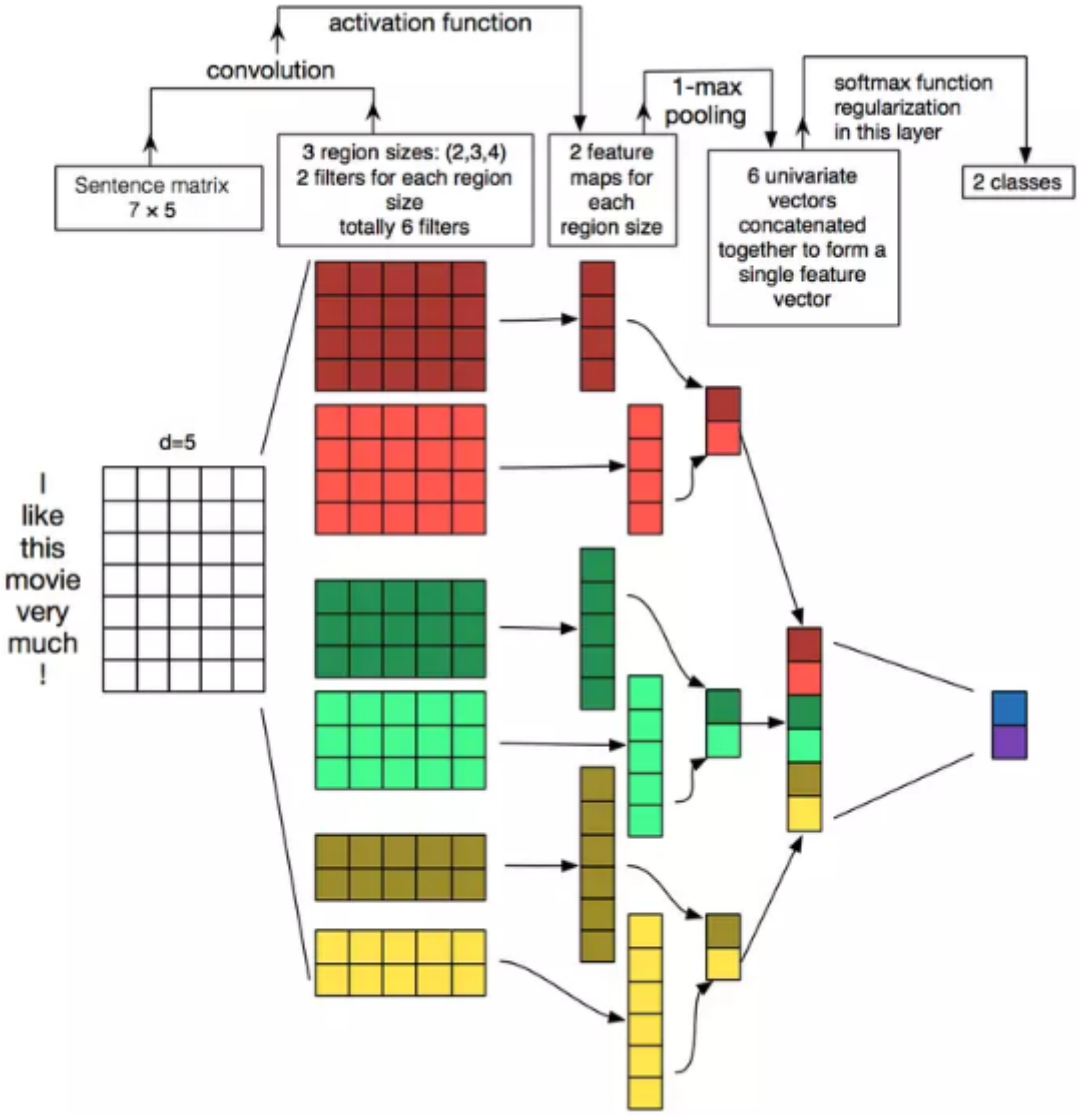
- Bắt đầu từ lớp đầu vào, truyền dữ liệu tới lớp đầu ra. Bước này là sự lan truyền về phía trước.(áp dụng hàm kích hoạt cho đầu vào z)
- Dựa trên kết quả đầu ra, tính toán sai số (chênh lệch giữa kết quả dự đoán và kết quả đã biết). Lỗi cần được giảm thiểu.

- Truyền ngược lỗi. Tìm đạo hàm của nó theo từng trọng số trong mạng và cập nhật mô hình.
- Lặp lại ba bước nêu trên qua nhiều epochs để tìm trọng số lý tưởng.

Cuối cùng, đầu ra được lấy thông qua Softmax để thu được nhãn lớp dự đoán.

2.4.4 Text CNN

Text CNN (Convolutional Neural Network) là một kiến trúc mạng nơ-ron sử dụng các tầng tích chập để xử lý dữ liệu văn bản. Các tầng tích chập giúp mô hình tự động học được các đặc trưng cấp độ cao từ các phần nhỏ của văn bản, giúp nó hiệu quả trong các nhiệm vụ như phân loại văn bản, phân tích cảm xúc, và dự đoán chuỗi thời gian văn bản.



Hình 2.7: Hình ảnh minh họa một mô hình Text CNN với 2 lớp phân loại

Kiến trúc mạng Text CNN có 5 lớp, bao gồm:

- Lớp Convolutional 1 chiều: Có 16 filter, giúp mô hình học các đặc trưng cấp độ cao từ dữ liệu văn bản.
- Lớp Pooling 1 chiều: Không có node. Có thể giảm kích thước của đầu ra từ lớp convolutional và giữ lại thông tin quan trọng.
- Lớp Flatten: Không có node. Chuyển đổi đầu ra từ các lớp trước đó thành một vectơ 1 chiều để đưa vào lớp fully connected.

- Lớp Dropout: Không có node. Được sử dụng để ngẫu nhiên loại bỏ một số node trong quá trình huấn luyện, giúp tránh tình trạng quá mức học.
- Lớp Fully Connected: Có 4 node, tương ứng với số lớp phân loại - số lượng lĩnh vực tin tức phân loại. Lớp này kết nối đầu vào từ lớp flatten và tạo ra đầu ra cuối cùng của mô hình.

Trong quá trình xử lý, chúng tôi áp dụng hàm kích hoạt **ReLU** cho cả các lớp convolutional và fully connected. Quyết định này nhằm tối ưu hóa khả năng học của mô hình, giúp nó nắm bắt các đặc trưng phức tạp và phi tuyến tính từ dữ liệu văn bản một cách hiệu quả.

Cho việc đánh giá hiệu suất của mô hình trong việc phân loại nhiều lớp, chúng tôi sử dụng hàm mất mát là **categorical_crossentropy**. Hàm này được điều chỉnh đặc biệt để phù hợp với bài toán của chúng tôi, đo lường khoảng cách giữa dự đoán và nhãn thực tế.

Mô hình được huấn luyện trong 230 epoch với batch size là 512 và learning rate là 1e-3. Trong mỗi chu kỳ huấn luyện, mô hình đi qua toàn bộ dữ liệu, cập nhật trọng số dựa trên một lượng dữ liệu nhỏ (batch). Learning rate là tham số quyết định độ lớn của bước cập nhật và quan trọng cho tốc độ học của mô hình.

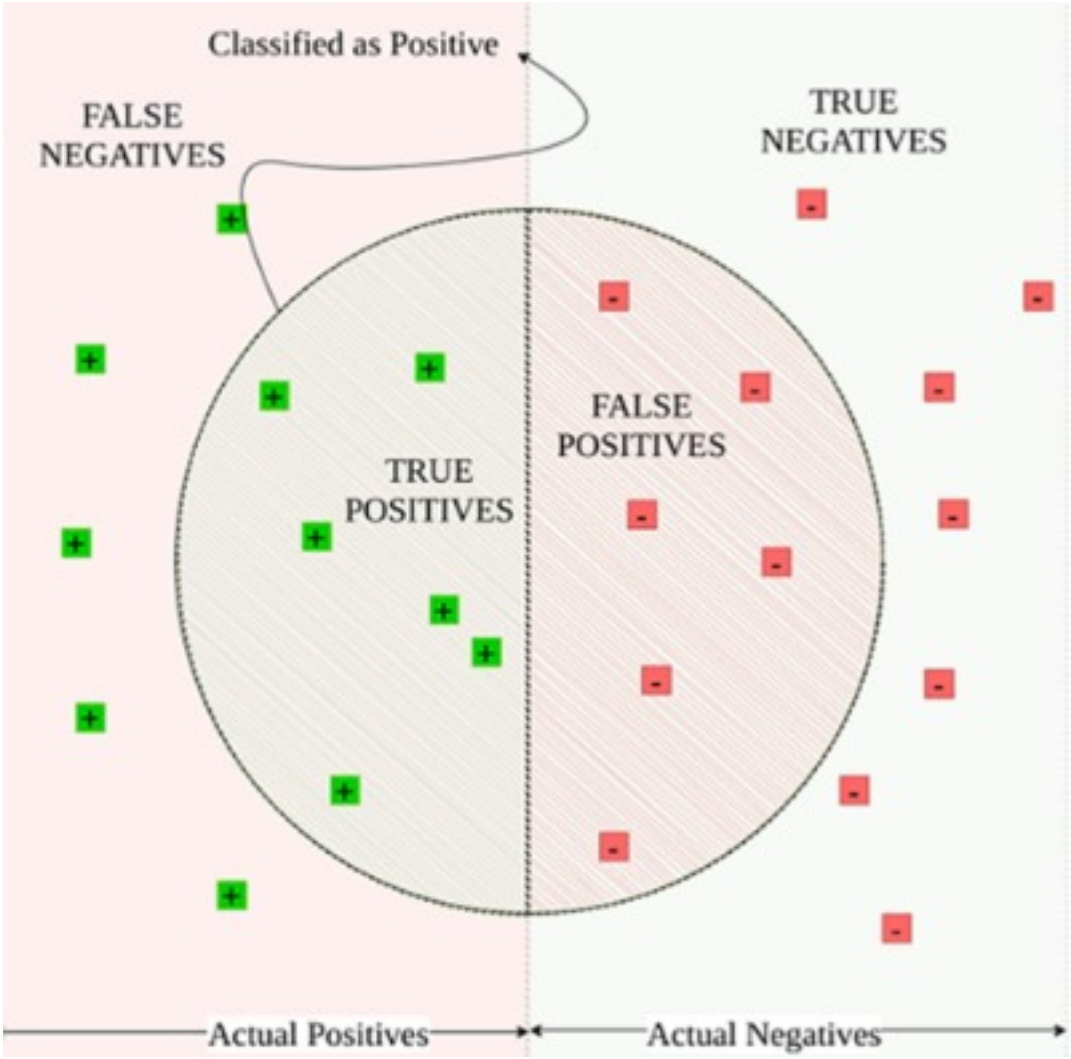
Trong Text CNN, cấu trúc đầu vào và đầu ra của mô hình được mô tả như sau:

- **Đầu vào:** Là một mảng 3 chiều có kích thước (batch_size, sequence_length, word_embedding_size), với:
  - `batch_size`: Số lượng mẫu trong một batch.
  - `sequence_length`: Số từ trong mỗi mẫu.
  - `word_embedding_size`: Kích thước của vector word embedding.
- **Đầu ra:** Là một mảng 1 chiều có kích thước (batch_size, num_classes), trong đó:
  - `batch_size`: Số lượng mẫu trong một batch.
  - `num_classes`: Là số lớp phân loại mà mô hình được huấn luyện để dự đoán.

Xử lý đầu ra: Kết quả đầu ra của lớp fully connected là một mảng 1 chiều với số phần tử bằng số lớp phân loại. Để thu được kết quả phân loại cuối cùng, chúng ta sử dụng hàm softmax, giúp chuyển đổi đầu ra thành một vector xác suất. Phần tử có xác suất cao nhất trong vector này sẽ là kết quả phân loại của mỗi mẫu.

2.5 Đánh giá mô hình

Trong đề tài này, chúng tôi lựa chọn đánh giá kết quả phân loại thông qua qua ba độ đo Accuracy, Precision, Recall, F1-score trên bộ dữ liệu test.



Hình 2.8: Minh họa confusion matrix.

Trong đó:

- TP là số lượng True Positives (dự đoán đúng là positive).
- TN là số lượng True Negatives (dự đoán đúng là negative).
- FP là số lượng False Positives (dự đoán là positive nhưng thực tế là negative).
- FN là số lượng False Negatives (dự đoán là negative nhưng thực tế là positive).

Accuracy (Độ chính xác) là tỷ lệ giữa số mẫu được phân loại chính xác trên tổng số mẫu. Công thức tính accuracy như sau:

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

Precision là tỉ lệ số lượng các điểm dữ liệu được phân loại đúng thành một lớp cụ thể so với tổng số điểm dữ liệu được phân loại vào lớp đó, bao gồm cả các trường hợp dự đoán false positives.

Công thức Precision cho lớp i trong mô hình đa lớp:

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

Recall là tỉ lệ số lượng các điểm dữ liệu được phân loại đúng thành một lớp cụ thể so với tổng số điểm thực tế thuộc vào lớp đó, bao gồm cả các trường hợp dự đoán false negatives.

Công thức Recall cho lớp i trong mô hình đa lớp:

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

$$\Rightarrow \text{Mô hình phân lớp tốt thì Precision và Recall đều cao}$$

F1 Score là một số đo kết hợp giữa Precision và Recall, được tính dựa trên trung bình điều hòa của chúng:

$$\text{F1 Score}_i = \frac{2 \times \text{Precision}_i \times \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}$$

F1 Score càng cao tương ứng precision và recall càng cao, mô hình phân loại càng tốt.

Chương3

CÀI ĐẶT

3.1 Hàm tiền xử lý

Hàm tiền xử lý của chúng tôi thực hiện các công việc: Giải mã các thực thể HTML và loại bỏ các thẻ HTML đó; loại bỏ kí hiệu chứng khoán, hyperlinks, các con số, hashtag; chuyển đổi về chữ thường; tokenize văn bản,...

```
    def preprocess_text(text):
        # Unescape HTML entities
        text = html.unescape(text)

        # Remove HTML tags
        text = re.sub(r'<.*?>', '', text)

        # Remove stock market tickers like $GE
        text = re.sub(r'\$\w*', '', text)

        # Remove hyperlinks
        text = re.sub(r'https?://\S+', '', text)

        # Remove numbers
        text = re.sub(r'\d+', '', text)

        # Remove hashtags (only the # sign)
        text = re.sub(r'#', '', text)

        # Lowercasing the text
        text = text.lower()

        # Tokenize the text only punctuation and character a-z pass
        tokens = re.findall(r'\b\w+\b|[.,;!?]', text)

        # Initialize a list to store the cleaned and lemmatized words
        cleaned_tokens = []
        # Initialize the WordNet Lemmatizer and English stopwords
        lemmatizer = WordNetLemmatizer()
        stopwords_english = stopwords.words('english')

        # Iterate through each word in the tokens
        for word in tokens:
            if (word not in stopwords_english and  # Remove stopwords
                    word not in string.punctuation):  # Remove punctuation
                lemmatized_word = lemmatizer.lemmatize(word)  # Lemmatize the
                                                                word
                cleaned_tokens.append(lemmatized_word)

        return cleaned_tokens

```

Chúng tôi tiến hành thực hiện tiền xử lý văn bản cho tập huấn luyện và tập kiểm tra.

```
    X_preprocess_train=[preprocess_text(text) for text in X_train]
    X_preprocess_test=[preprocess_text(text) for text in X_test]

```

3.2 Bag of words

Đầu tiên, chúng tôi tạo ra một vectorizer để chuyển đổi văn bản thành ma trận tần suất từ vựng (Bag-of-Words) dựa trên tần suất xuất hiện của từng từ trong văn bản.

```
    from sklearn.feature_extraction.text import CountVectorizer
    vectorizer = CountVectorizer(min_df=5)
    vectorizer.fit([' '.join(tokens) for tokens in X_preprocess_train])

```

Sau đó, chúng tôi chuyển đổi tập dữ liệu huấn luyện và kiểm tra thành các ma trận tần suất từ vựng đã được xây dựng từ tập huấn luyện.

```
    X_bow_train=vectorizer.transform([' '.join(tokens) for tokens in
                                          X_preprocess_train])
    X_bow_test=vectorizer.transform([' '.join(tokens) for tokens in
                                          X_preprocess_test])

```

3.3 Word2vec

Đầu tiên, tải mô hình Word2Vec đã được đào tạo từ tệp tin.

```
    from gensim.models import KeyedVectors
    word2vec_model  = KeyedVectors.load_word2vec_format('/content/drive/
    MyDrive/GoogleNews-vectors-negative300.bin.gz', binary=True)

```

Sau đó, tạo ra các vectơ Word2Vec từ văn bản đã được tiền xử lý. Các từ không có sẵn trong từ điển của mô hình Word2Vec được loại bỏ.

```

    X_word2vec_train=[[word2vec_model.get_vector(token) for token in tokens
              if token in word2vec_model] for tokens in X_preprocess_train]
    X_word2vec_test=[[word2vec_model.get_vector(token) for token in tokens
             if token in word2vec_model] for tokens in X_preprocess_test]

```

Cuối cùng, tính trung bình của các vectơ Word2Vec để tạo ra các vectơ đại diện cho mỗi văn bản trong tập huấn luyện và kiểm tra.

```
    X_avg_train=np.array([np.mean(X_word2vec_train[i],axis=0) for i in range
                                          (len(X_word2vec_train))])
    X_avg_test=np.array([np.mean(X_word2vec_test[i],axis=0) for i in range
                                          (len(X_word2vec_test))])

```

3.4 Hàm đánh giá

Trong hàm đánh giá, chúng tôi thực hiện các công việc như: Tính confusion matrix từ các giá trị thực tế và dự đoán; vẽ heatmap để trực quan hóa; in ra báo cáo phân loại bao gồm precision, recall và f1-score; xác định và in ra một số trường hợp bị phân loại sai.

```
    def evaluation(y_true, y_pred, class_name, X):
        # Compute the confusion matrix
        conf_matrix = confusion_matrix(y_true, y_pred)

        def plot_confusion_matrix(conf_matrix, class_names):
            fig, ax = plt.subplots(figsize=(8, 6))
            sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues',
                        xticklabels=class_names, yticklabels=class_names)
            plt.title('Confusion Matrix')
            plt.xlabel('Predicted')
            plt.ylabel('True')
            plt.show()

        # Plot the heatmap of the confusion matrix
        plot_confusion_matrix(conf_matrix, class_names=list(class_name.
                                                            values()))

        # Print the classification report
        print("\nClassification Report:")
        print(classification_report(y_true, y_pred, labels=list(class_name.
                           keys()), target_names=list(class_name.values())))

        # Identify misclassified examples
        errors = [(y_true[i], y_pred[i], X[i]) for i in range(len(y_true))
                                               if y_true[i] != y_pred[i]]

        # Print some misclassified examples
        print("\nSome Misclassified Examples:")
        for true_label, predicted_label, example in errors[:min(5,
                                                        len(errors))]:
            print(f"True Label: {class_name[true_label]}, Predicted Label:
                        {class_name[predicted_label]}, Example: {example}")


```

3.5 Naive Bayes

Với mô hình Naive Bayes, chúng tôi đã thực hiện các công việc sau: import thư viện, khởi tạo mô hình, huấn luyện, dự đoán và đánh giá hiệu suất.

```
    #Import the library
    from sklearn.naive_bayes import MultinomialNB
    #Initialize the Naive Bayes model
    Naive_Bayes_model = MultinomialNB(alpha=0.01,fit_prior=True)
    #Train the model on the training set
    Naive_Bayes_model .fit(X_bow_train, y_train)
    #Predict on the test set
    y_pred_test=Naive_Bayes_model.predict(X_bow_test)
    #Evaluate performance
    evaluation(y_test,y_pred_test,class_name,X_test)

```

3.6 Logistic Regression

Với mô hình Logistic Regression, chúng tôi đã thực hiện các công việc sau: import thư viện, khởi tạo mô hình, huấn luyện, dự đoán và đánh giá hiệu suất.

```
    #Import the library
    from sklearn.linear_model import LogisticRegression
    #Initialize the Logistic Regression model
    Logistic_Regression_model = LogisticRegression(random_state=0,max_iter
    =10000)
    #Train the model on the training set
    Logistic_Regression_model.fit(X_avg_train,y_train)
    #Predict on the test set
    y_pred_test=Logistic_Regression_model.predict(X_avg_test)
    #Evaluate performance
    evaluation(y_test,y_pred_test,class_name,X_test)

```

3.7 Multi-Layer Perceptron

Với mô hình Multi-Layer Perceptron, chúng tôi import các thư viện sau:

```
    from tensorflow.keras import Sequential
    from tensorflow.keras.layers import Dense
    import tensorflow as tf

```

Thực hiện khởi tạo mô hình:

```
    MLP_model = Sequential([
            tf.keras.Input(shape=(300,)),
            Dense(units=512,kernel_regularizer=l2(0.01), activation='relu'),
            Dense(units=256, activation='relu'),
            Dense(units=128, activation='relu'),
            Dense(units=4, activation='softmax')
                ],name='multilayer_perceptron')

```

Sau đó, chúng tôi cấu hình quá trình huấn luyện cho MLP như sau:

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

```

Chúng tôi thực hiện mã hóa one-hot (one-hot encoding) cho nhãn lớp trong tập huấn luyện và tập kiểm tra.

```
    y_OHE_train = to_categorical(y_train-1)
    y_OHE_test = to_categorical(y_test-1)

```

Thực hiện quá trình huấn luyện, dự đoán và đánh giá hiệu suất của mô hình:

```
    #Train the model on the training set
    history=MLP_model.fit(X_avg_train, y_OHE_train, batch_size=128, epochs
                                                =100,validation_split=0.2)
    #Predict on the test set
    y_pred=np.argmax(MLP_model.predict(X_avg_test),axis=1)+1
    #Evaluate performance
    evaluation(y_test,y_pred_test,class_name,X_test)

```

3.8 Text CNN

Với mô hình Text CNN, chúng tôi import các thư viện sau:

```
    from tensorflow.keras import Sequential
    from tensorflow.keras.layers import Dense,Conv1D, MaxPooling1D, Flatten,
                                                Input,RNN,Masking,Dropout
    from tensorflow.keras.optimizers import Adam

```

Thực hiện khởi tạo mô hình:

```
    def create_textCNN_model(num_classes,learning_rate):
        model=Sequential([
            Conv1D(filters=16, kernel_size=3, activation='relu'),
            MaxPooling1D(pool_size=2),
            Flatten(),
            Dropout(rate=0.1),
            Dense(num_classes, activation='softmax')])
        model.compile(optimizer=Adam(learning_rate=learning_rate),
                      loss='categorical_crossentropy',
                       metrics=['categorical_accuracy'])
        return model

```

Hàm **data_generator** để tạo các batch dữ liệu:

```
    def data_generator(data, batch_size):
        # Infinite loop for creating batches
        while True:
            # Randomly choose batch indices
            batch_indices = np.random.choice(len(data[0]), batch_size,
                                                        replace=False)
            # Create a batch from selected indices
            batch = (data[0][batch_indices], data[1][batch_indices])
            # Yield the batch for training
            yield batch


```

Đoạn code thực hiện quá trình huấn luyện mô hình sử dụng generator và lưu trọng số sau mỗi epoch:

```
    def training(model, batch_size, epochs, training_data,
    initial_checkpoint_path, final_checkpoint_path):
        # Load initial checkpoint if provided
        if initial_checkpoint_path is not None:
            model.load_weights(initial_checkpoint_path)
        # Create a generator from the training data
        train_generator = data_generator(training_data, batch_size)
        # Train the model and save the final weights
        history = model.fit(
            x=train_generator,
            steps_per_epoch=len(training_data[0]) // batch_size,
            epochs=epochs,)
        model.save_weights(final_checkpoint_path)

        return model, history

```

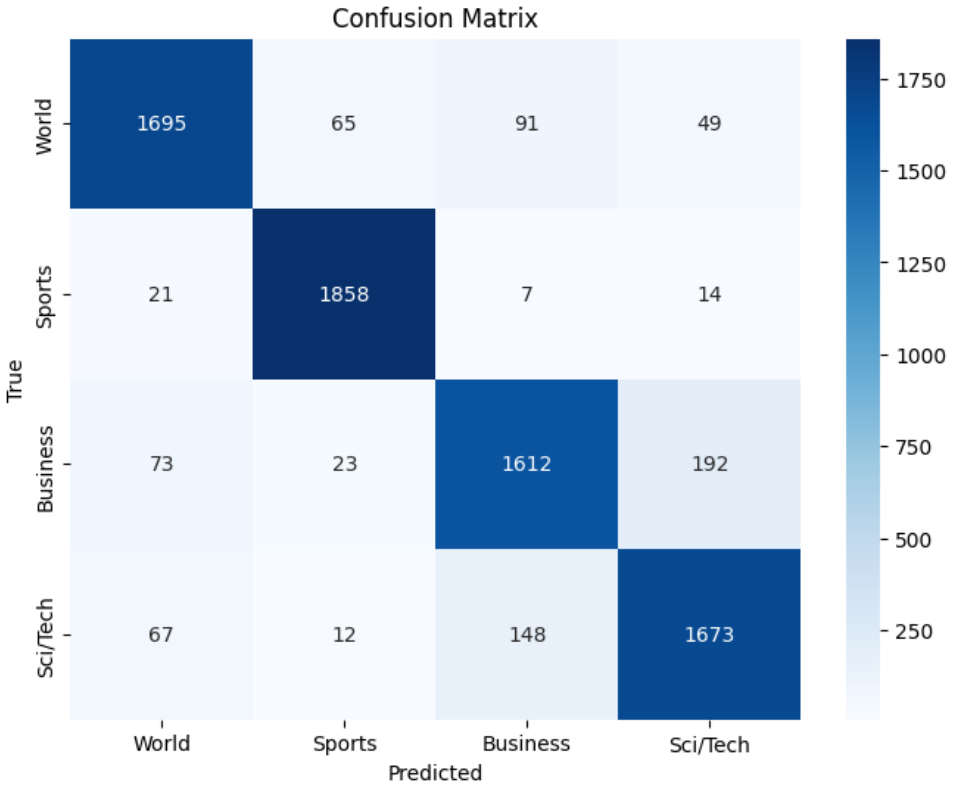
Dự đoán và đánh giá hiệu suất:

```
    #Predict on the test set
    y_pred=np.argmax(textCNN.predict(X_word2vec_test_padded),axis=1)+1
    #Evaluate performance
    evaluation(y_test,y_pred,class_name,X_test)

```

Chương4

KẾT QUẢ, NHẬN XÉT

4.1 Naive Bayes



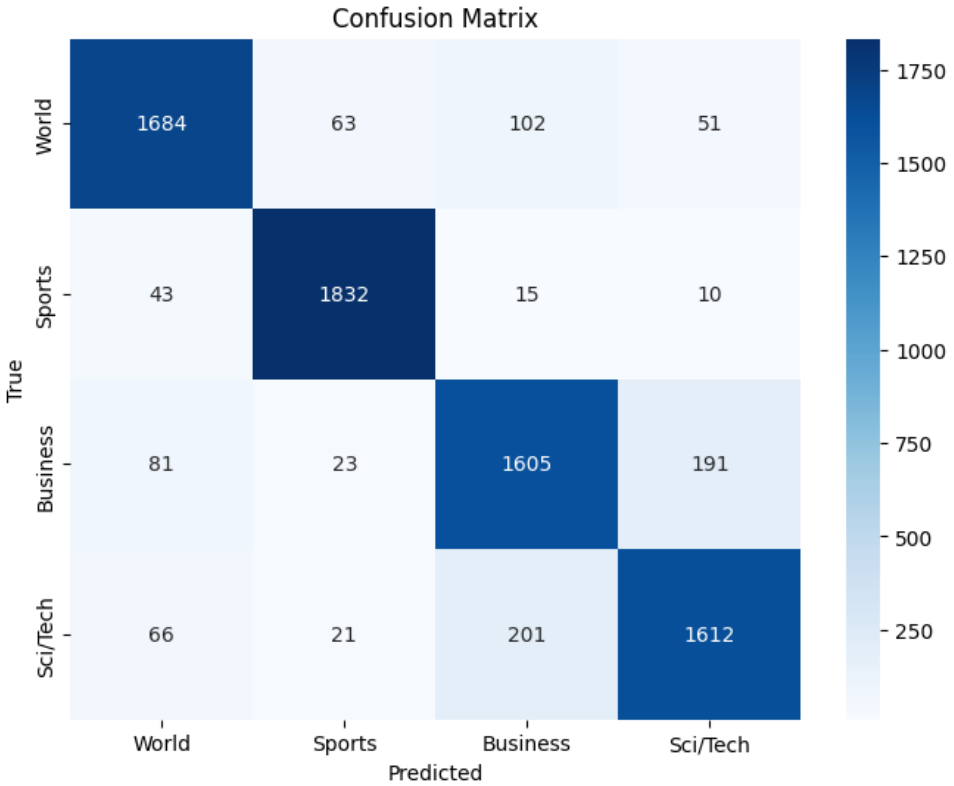
Hình 4.1: Confusion Matrix của Naive Bayes

```
               Classification Report:
                 precision    recall  f1-score   support

         World       0.91      0.89      0.90      1900
        Sports       0.95      0.98      0.96      1900
      Business       0.87      0.85      0.86      1900
      Sci/Tech       0.87      0.88      0.87      1900

      accuracy                           0.90      7600
     macro avg       0.90      0.90      0.90      7600
  weighted avg       0.90      0.90      0.90      7600
```

Hình 4.2: Báo cáo phân loại của Naive Bayes

Nhận xét: Mô hình Naive Bayes gặp khó khăn khi phân loại tin tức khoa học/công nghệ và tin tức kinh doanh. Các dự đoán sai chủ yếu xuất phát từ sự mơ hóa giữa ngôn ngữ kinh doanh và công nghệ, cũng như sự chú ý quá mức vào tiêu đề mà bỏ qua nội dung thực sự của bài báo. Để cải thiện, cần bổ sung dữ liệu đa dạng và cải thiện khả năng hiểu biết của mô hình về nội dung bài báo, đồng thời điều chỉnh tham số mô hình.

4.2 Logistic Regression



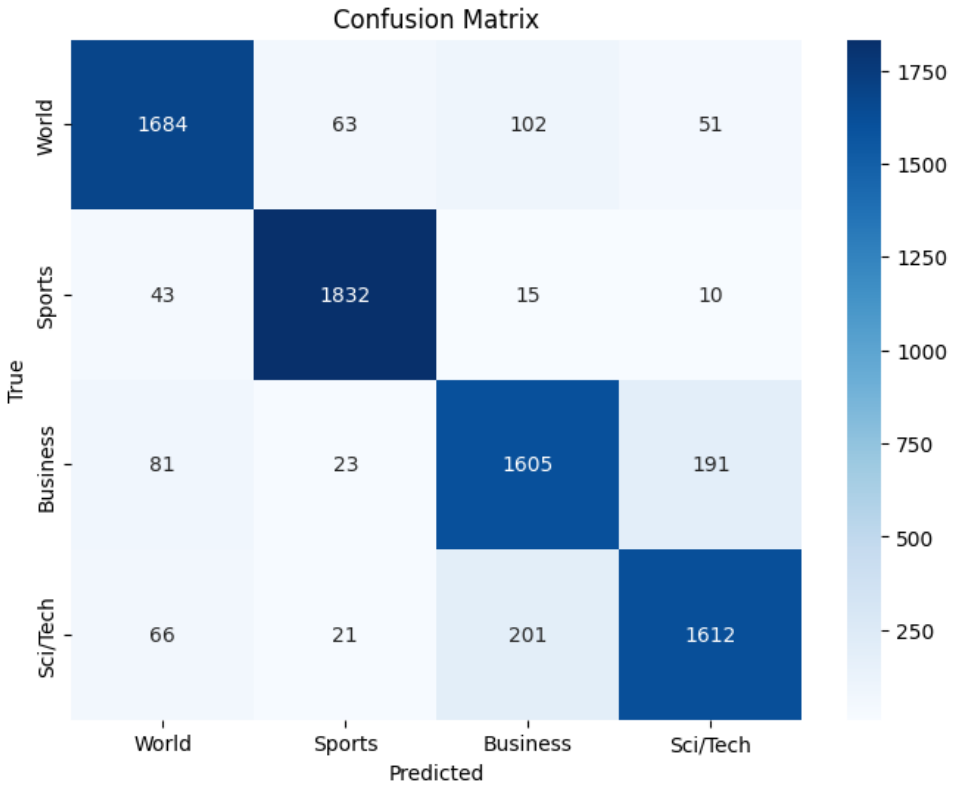
Hình 4.3: Confusion Matrix của Logistic Regression

```
               Classification Report:
                 precision    recall  f1-score   support

          World       0.90      0.89      0.89      1900
         Sports       0.94      0.96      0.95      1900
       Business       0.83      0.84      0.84      1900
       Sci/Tech       0.86      0.85      0.86      1900

       accuracy                           0.89      7600
      macro avg       0.89      0.89      0.89      7600
   weighted avg       0.89      0.89      0.89      7600
```

Hình 4.4: Báo cáo phân loại của Logistic Regression

Nhận xét: Mô hình Logistic Regression đang mắc phải việc dự đoán sai với các bài báo khoa học/công nghệ, nhưng mô hình đang nhầm lẫn với tin tức kinh doanh do sự mập mờ giữa các ngữ cảnh ngôn ngữ tài chính và nội dung khoa học/công nghệ. Cải thiện có thể đạt được thông qua việc sử dụng dữ liệu đa dạng hơn và điều chỉnh mô hình để nhận biết rõ ràng hơn giữa các thể loại này.

4.3 Multi-Layer Perceptron



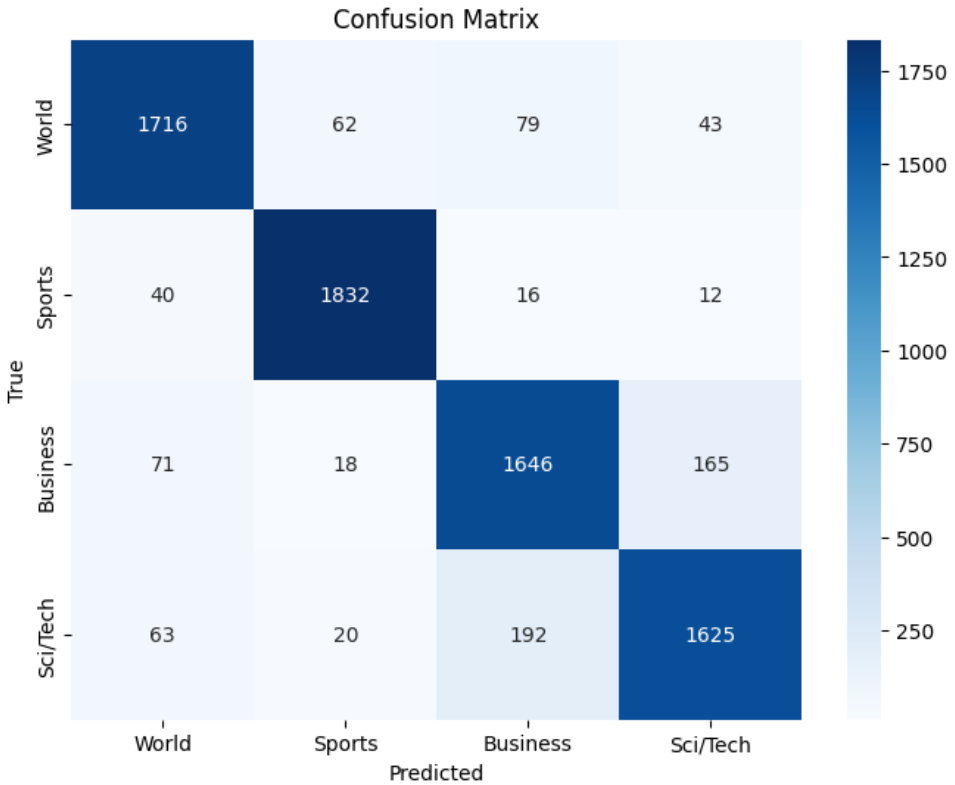
Hình 4.5: Confusion Matrix của MLP

```
                Classification Report:
                precision    recall  f1-score   support

         World       0.90      0.89      0.89      1900
        Sports       0.94      0.96      0.95      1900
      Business       0.83      0.84      0.84      1900
      Sci/Tech       0.86      0.85      0.86      1900

      accuracy                           0.89      7600
     macro avg       0.89      0.89      0.89      7600
  weighted avg       0.89      0.89      0.89      7600
```

Hình 4.6: Báo cáo phân loại của MLP

Nhận xét: Mô hình MLP đang mắc phải việc dự đoán sai với các bài báo khoa học/công nghệ, nhưng mô hình đang nhầm lẫn với tin tức kinh doanh do sự mập mờ giữa các ngữ cảnh ngôn ngữ tài chính và nội dung khoa học/công nghệ. Cải thiện có thể đạt được thông qua việc sử dụng dữ liệu đa dạng hơn và điều chỉnh mô hình để nhận biết rõ ràng hơn giữa các thể loại này.

4.4 Text CNN



Hình 4.7: Confusion Matrix của Text CNN

```
                Classification Report:
                precision    recall  f1-score   support

         World       0.91      0.90      0.91      1900
        Sports       0.95      0.96      0.96      1900
      Business       0.85      0.87      0.86      1900
      Sci/Tech       0.88      0.86      0.87      1900

      accuracy                           0.90      7600
     macro avg       0.90      0.90      0.90      7600
  weighted avg       0.90      0.90      0.90      7600
```

Hình 4.8: Báo cáo phân loại của Text CNN

Nhận xét: Text CNN đã đạt được kết quả phân loại tốt trong việc nhận biết tin tức từ bốn lĩnh vực khác nhau: thế giới, thể thao, kinh doanh, và khoa học/công nghệ. Mô hình thể hiện độ chính xác cao, đặc biệt là trong lĩnh vực thể thao, trong khi kinh doanh có vẻ là một thách thức hơn với độ chính xác thấp hơn. Có thể là do sự đa dạng và phức tạp của thông tin kinh doanh, với ngôn ngữ chuyên ngành và thông tin tài chính làm tăng khó khăn trong việc hiểu và phân loại. Điều này yêu cầu tối ưu hóa mô hình hoặc cân nhắc các chiến lược tăng cường dữ liệu và xử lý đặc trưng chuyên sâu cho lĩnh vực này để cải thiện độ chính xác.

Chương5

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Nhận xét

Đồ án của chúng tôi đã chứng minh được khả năng phân loại tiêu đề và mô tả bài báo vào các lĩnh vực tin tức như Thể thao, Thế giới, Kinh doanh và Khoa học/Công nghệ với hiệu suất cao. Sự kết hợp giữa các kĩ thuật Bag of Words và Word2Vec cùng với mô hình học máy đã đạt được kết quả ấn tượng trong việc phân loại và phân biệt các lĩnh vực tin tức.

Công việc áp dụng Bag of Words và Word2Vec đã giúp chúng tôi xây dựng các biểu diễn từ vựng hiệu quả, từ đó cải thiện khả năng nhận diện và phân loại. Mô hình Naive Bayes, Logistic Regression và MLP đã chứng minh tính hiệu quả của chúng trong việc xử lý bài toán phân loại lĩnh vực tin tức.

Công việc của chúng tôi cung cấp một cơ sở vững chắc để mở rộng và ứng dụng trong các lĩnh vực khác nhau, không chỉ trong việc phân loại tin tức mà còn trong việc xử lý văn bản và phân loại thông tin trong nhiều lĩnh vực ứng dụng khác.

5.2 Hướng phát triển

Mô hình vẫn gặp phải một số trường hợp phân loại không chính xác. Để cải thiện, chúng tôi đang xem xét việc mở rộng tập dữ liệu hoặc cải thiện quá trình thu thập dữ liệu. Điều này sẽ mang lại cơ sở dữ liệu phong phú hơn, giúp mô hình học máy nhận diện và sử dụng được nhiều đặc trưng hơn, từ đó trở nên linh hoạt và chính xác hơn trong việc phân loại.

Ngoài ra, chúng tôi đang tiếp tục tối ưu hóa các mô hình hiện có và thử nghiệm với các mô hình mới để cải thiện hiệu suất và giảm độ phức tạp của mô hình. Đặc biệt, chúng tôi đang tập trung vào việc khám phá và áp dụng các kỹ thuật mới như Transformer, BERT và các mô hình học sâu khác để nâng cao hiệu suất phân loại.

Hơn nữa, chúng tôi đặt mục tiêu áp dụng kết quả đồ án vào các ứng dụng thực tế, ví dụ như hệ thống tự động phân loại tin tức trên các trang web hoặc ứng dụng di động. Điều này sẽ giúp cải thiện trải nghiệm người dùng và mở ra nhiều cơ hội ứng dụng rộng rãi cho công trình nghiên cứu của chúng tôi.