**A picture containing picture frame, rectangle, screenshot, frame

Description automatically generated TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGÀNH KỸ THUẬT DỮ LIỆU**

**­\_\_\_🙢🕮🙠\_\_\_**



Ảnh có chứa văn bản, mẫu họa

Mô tả được tạo tự động

**KHÓA LUẬN TỐT NGHIỆP**

**PHÂN LOẠI VĂN BẢN DÙNG MÔ HÌNH BERT**

|  |  |
| --- | --- |
| **GVHD:** | **TS. Nguyễn Thành Sơn** |
| **SVTH1:** | **20133040 – Nguyễn Thị Diệu Hiền** |
| **SVTH2:** | **20133033 – Bùi Tấn Đạt** |

*TP. Hồ Chí Minh, tháng 07 năm 2024*

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGÀNH KỸ THUẬT DỮ LIỆU**

 **­\_\_\_🙢🕮🙠\_\_\_**

Ảnh có chứa văn bản, mẫu họa

Mô tả được tạo tự động

**KHÓA LUẬN TỐT NGHIỆP**

**PHÂN LOẠI VĂN BẢN DÙNG MÔ HÌNH BERT**

|  |  |
| --- | --- |
| **GVHD:** | **TS. Nguyễn Thành Sơn** |
| **SVTH1:** | **20133040 – Nguyễn Thị Diệu Hiền** |
| **SVTH2:** | **20133033 – Bùi Tấn Đạt** |

*TP. Hồ Chí Minh, tháng 07 năm 2024*

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA CNTT Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\* \*\*\*\*\*\*\***

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên 1: **Nguyễn Thị Diệu Hiền** MSSV: **20133040**

Họ và tên Sinh viên 2: **Bùi Tấn Đạt** MSSV: **20133033**

Chuyên ngành: **Kỹ thuật Dữ liệu**

Đề tài: **Phân loại văn bản dùng mô hình BERT**

Họ và tên Giáo viên hướng dẫn: **TS. Nguyễn Thành Sơn**

**NHẬN XÉT**

1. Về nội dung đề tài và khối lượng thực hiện:

1. Ưu điểm:

1. Khuyết điểm:

1. Đề nghị cho bảo vệ hay không?
2. Đánh giá loại:
3. Điểm:

*TP. Hồ Chí Minh, tháng 07 năm 2024*

Giáo viên hướng dẫn

*(Ký & ghi rõ họ tên)*

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA CNTT Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\* \*\*\*\*\*\*\***

**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Họ và tên Sinh viên 1: **Nguyễn Thị Diệu Hiền** MSSV: **20133040**

Họ và tên Sinh viên 2: **Bùi Tấn Đạt** MSSV: **20133033**

Chuyên ngành: **Kỹ thuật Dữ liệu**

Đề tài: **Phân loại văn bản dùng mô hình BERT**

Họ và tên Giáo viên phản biện: **ThS. Nguyễn Văn Thành**

**NHẬN XÉT**

1. Về nội dung đề tài và khối lượng thực hiện:

1. Ưu điểm:

1. Khuyết điểm:

1. Đề nghị cho bảo vệ hay không?
2. Đánh giá loại:
3. Điểm:

*TP. Hồ Chí Minh, tháng 07 năm 2024*

Giáo viên phản biện

*(Ký & ghi rõ họ tên)*

**LỜI CẢM ƠN**

Trước hết, nhóm chúng em xin gửi lời cảm ơn sâu sắc đến Khoa Công nghệ Thông tin, Trường Đại học Sư phạm Kỹ thuật TP.HCM, đã tạo mọi điều kiện thuận lợi để nhóm chúng em có thể thực hiện và hoàn thành đề tài này.

Chúng em cũng xin bày tỏ lòng biết ơn đặc biệt đến Thầy Nguyễn Thành Sơn, người đã trực tiếp hỗ trợ và hướng dẫn nhóm trong suốt quá trình thực hiện đề tài. Chúng em cảm ơn Thầy vì những lời khuyên quý báu từ kinh nghiệm thực tiễn, những định hướng cụ thể, sự giải đáp kịp thời các thắc mắc và những góp ý chỉnh sửa giúp nhóm hoàn thành đề tài đúng tiến độ.

Mặc dù đã nỗ lực hết sức để hoàn thành đề tài và báo cáo khóa luận tốt nghiệp này, chúng em nhận thức rằng vẫn còn nhiều hạn chế về kiến thức, kỹ thuật cũng như kinh nghiệm với những kiến thức khổng lồ xoay quanh đề tài này, chắc hẳn sẽ không thể nào tránh khỏi những thiếu xót không đáng có. Vì vậy, nhóm chúng em rất mong nhận được những ý kiến đóng góp từ Quý Thầy Cô để có thể cải thiện và phát triển hơn nữa trong các nghiên cứu sau này.

Cuối cùng, chúng em xin kính chúc Quý Thầy Cô luôn dồi dào sức khỏe và gặt hái nhiều thành công trong sự nghiệp giáo dục. Nhóm chúng em xin chân thành cảm ơn!

**ĐH SƯ PHẠM KỸ THUẬT TP.HCM CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**KHOA CNTT Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\* \*\*\*\*\*\*\***

**ĐỀ CƯƠNG KHÓA LUẬN TỐT NGHIỆP**

Họ và tên Sinh viên 1: **Nguyễn Thị Diệu Hiền** MSSV: **20133040**

Họ và tên Sinh viên 2: **Bùi Tấn Đạt** MSSV: **20133033**

Chuyên ngành: **Kỹ thuật Dữ liệu**

Đề tài: **Phân loại văn bản dùng mô hình BERT**

Họ và tên Giáo viên hướng dẫn: **TS. Nguyễn Thành Sơn**

**Nhiệm vụ của luận văn:**

1. Tìm hiểu về bài toán Phân loại văn bản.
2. Nghiên cứu lý thuyết, cách xây dựng các mô hình học sâu quan trọng mà nhóm muốn áp dụng trong đề tài là Bi-LSTM, PhoBERT và Longformer PhoBERT.
3. Tiến hành đánh giá mô hình bằng các trường hợp thực nghiệm.
4. Thiết kế giao diện bằng thư viện Streamlit nhằm thể hiện các chức năng của đề tài một cách trực quan.

**Đề cương viết luận văn:**

1. **Phần mở đầu**
2. Lý do chọn đề tài
3. Mục tiêu và nhiệm vụ nghiên cứu
4. Đối tượng và phạm vi nghiên cứu
5. **Phần nội dung**
6. Chương 1: Bài toán Phân loại văn bản
7. Chương 2: Các mô hình sử dụng
8. Mô hình Bi-LSTM
9. Transformer
10. Longformer
11. Mô hình BERT
12. Mô hình PhoBERT và Longformer PhoBERT
13. Chương 3: Thực nghiệm và đánh giá kết quả
14. Cài đặt mô hình
15. Trình tự thực hiện
16. So sánh kết quả của ba mô hình Bi-LSTM, PhoBERT và Longformer PhoBERT
17. **Phần kết luận**
18. Những kết quả đạt được
19. Hạn chế của đề tài
20. Phương hướng phát triển

**KẾ HOẠCH THỰC HIỆN**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Thời gian** | **Công việc** | **Ghi chú** |
| 1 | 20/01/2024 – 31/01/2024 | Tìm hiểu về đề tài, các bài toán và các mô hình liên quan |  |
| 2 | 01/02/2024 – 15/02/2024 | Nghỉ Tết Âm lịch |  |
| 3 | 16/02/2024 – 29/02/2024 | Tìm hiểu lý thuyết về các mô hình Bi-LSTM, BERT, PhoBERT và Longformer PhoBERT |  |
| 4 | 01/03/2024 – 31/03/2024 | * Tìm các tập dữ liệu bài báo tiếng Việt * Tiến hành code xử lý dữ liệu * Viết lý thuyết về các mô hình |  |
| 5 | 01/04/2024 – 30/04/2024 | * Tiếp tục viết và sửa code * Viết code để tạo giao diện bằng streamlit * Giới hạn độ lớn tập dữ liệu * Tiếp tục viết lý thuyết về các mô hình và các chủ đề liên quan đến Khóa luận. | Độ lớn tập dữ liệu không đủ tài nguyên để train |
| 6 | 01/05/2024 – 20/05/2024 | * Tiếp tục sửa code mô hình và test trên các tập dữ liệu với các độ lớn khác nhau * Tiếp tục viết lý thuyết và các bước cài đặt mô hình |  |
| 7 | 21/05/2024 – 04/06/2024 | * Hoàn thiện code và chốt tập dữ liệu cuối cùng * Hoàn thiện lý thuyết và làm file trình chiếu powerpoint |  |
| 8 | 05/06/2024 | Báo cáo trước Giáo viên hướng dẫn lần 1 |  |
| 9 | 06/06/2024 – 24/06/2024 | Hoàn thiện giao diện Streamlit và deloy lên Hugggingface |  |
| 10 | 25/06/2024 | Báo cáo trước Giáo viên hướng dẫn lần 2 |  |

**MỤC LỤC**

[MỞ ĐẦU 1](#_Toc11913)

[1. Lý do chọn đề tài 1](#_Toc25674)

[2. Mục tiêu và nhiệm vụ nghiên cứu 1](#_Toc5496)

[3. Đối tượng và phạm vi nghiên cứu 2](#_Toc10442)

[NỘI DUNG 3](#_Toc24499)

[CHƯƠNG 1: BÀI TOÁN PHÂN LOẠI VĂN BẢN 3](#_Toc30729)

[1.1. TỔNG QUAN VỀ BÀI TOÁN PHÂN LOẠI VĂN BẢN 3](#_Toc2249)

[1.2 BÀI TOÁN PHÂN LOẠI BÀI BÁO VÀ TIN TỨC TIẾNG VIỆT 6](#_Toc22130)

[1.2.1 Đặc điểm ngôn ngữ tiếng Việt 6](#_Toc1574)

[1.2.2 Thách thức trong phân loại bài báo, tin tức tiếng Việt 7](#_Toc9136)

[1.2.3 Phương pháp tiếp cận và mô hình ứng dụng cho bài toán 7](#_Toc8464)

[1.3 CÁC CÔNG TRÌNH ỨNG DỤNG LIÊN QUAN TỚI BÀI TOÁN PHÂN LOẠI VĂN BẢN TIẾNG VIỆT 8](#_Toc21289)

[1.3.1 PhoBERT: Pre-trained language models for Vietnamese 8](#_Toc31141)

[1.3.2 Vietnamese News Articles Classification Using Neural Networks 8](#_Toc23947)

[1.3.3 Universal language model fine-tuning for text classification 9](#_Toc17086)

[1.3.4 Improving Imbalanced Text Classification with Dynamic Curriculum Learning 10](#_Toc23461)

[CHƯƠNG 2: CÁC MÔ HÌNH SỬ DỤNG 11](#_Toc1500)

[2.1 MÔ HÌNH BI-LSTM 11](#_Toc32032)

[2.1.1 Mô hình RNN (Recurrent Neural Network) 11](#_Toc30604)

[2.1.2 Mô hình LSTM (Long Short-Term Memory) 14](#_Toc25038)

[2.1.3 Mô hình Bi-LSTM (Bidirectional Long Short-Term Memory) 16](#_Toc26600)

[2.2 TRANSFORMER 18](#_Toc21163)

[2.2.1 Mô hình Sequence-to-sequence trong RNN 19](#_Toc30749)

[2.2.2 Decoder và Encoder 20](#_Toc32716)

[2.3 LONGFORMER 27](#_Toc17690)

[2.3.1 Tổng quan về Longformer 27](#_Toc21194)

[2.3.2 Quy mô bậc hai trong Self-Attention 27](#_Toc30942)

[2.3.3 Cơ chế Attention trong Longformer 28](#_Toc8549)

[2.4 MÔ HÌNH BERT 31](#_Toc32504)

[2.4.1 Tổng quan về BERT 31](#_Toc10874)

[2.4.2 Tại sao lại cần BERT? 32](#_Toc15922)

[2.4.3 Một số khái niệm 32](#_Toc22406)

[2.4.4 Cách hoạt động của BERT 34](#_Toc2181)

[2.4.5 Các kiến trúc mô hình BERT 37](#_Toc20290)

[2.4.6 Mô hình BERT trong phân loại văn bản 38](#_Toc15308)

[2.5 MÔ HÌNH PHOBERT VÀ LONGFORMER PHOBERT 39](#_Toc2591)

[2.5.1 Sự ra đời của PhoBERT 39](#_Toc8387)

[2.5.2 Cấu trúc của PhoBERT 39](#_Toc24494)

[2.5.3 Longformer PhoBERT 40](#_Toc17243)

[CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ 41](#_Toc31796)

[3.1 MÔI TRƯỜNG VÀ DỮ LIỆU THỰC NGHIỆM 41](#_Toc5749)

[3.1.1 Môi trường cài đặt 41](#_Toc26623)

[3.1.2 Các thư viện python (đã sử dụng) 41](#_Toc1645)

[3.1.3 Dữ liệu thực nghiệm 47](#_Toc22066)

[3.2 TRÌNH TỰ THỰC HIỆN 50](#_Toc31017)

[3.2.1 Tiền xử lý dữ liệu 50](#_Toc24353)

[3.2.2 Trích xuất đặc trưng PhoBERT 51](#_Toc28557)

[3.2.3 Huấn luyện các mô hình 52](#_Toc10879)

[3.2.4 Đánh giá mô hình 57](#_Toc23414)

[3.3 SO SÁNH CÁC MÔ HÌNH 63](#_Toc28637)

[KẾT LUẬN 65](#_Toc5807)

[1. Những kết quả đạt được 65](#_Toc10324)

[2. Hạn chế của đề tài 65](#_Toc2691)

[3. Phương hướng phát triển 65](#_Toc25168)

[TÀI LIỆU THAM KHẢO 67](#_Toc21711)

[PHỤ LỤC 69](#_Toc28879)

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
| **Tên viết tắt** | **Tên đầy đủ** |
| BERT | Bidirectional Encoder Representations from Transformers |
| CNN | Convolutional Neural Network |
| RNN | Recurrent Neural Network |
| NLP | Natural Language Processing |
| Bi-LSTM | Bidirectional Long Short-Term Memory |
| LSTM | Long Short-Term Memory |
| GPT | Generative Pre-training Transformer |
| MLM | Masked Language Model |
| NSP | Next Sentence Prediction |

**DANH MỤC BẢNG**

[Bảng 3.1: Siêu tham số cho mô hình Bi-LSTM 46](#_Toc171232817)

[Bảng 3.2: Ví dụ về ma trận nhầm lẫn 50](#_Toc171232818)

[Bảng 3.3: Kết quả đánh giá trên từng nhãn mô hình Bi-LSTM 51](#_Toc171232819)

[Bảng 3.4: Kết quả đánh giá trên từng nhãn PhoBERT 53](#_Toc171232820)

[Bảng 3.5: Kết quả đánh giá trên từng nhãn Longformer PhoBERT 54](#_Toc171232821)

[Bảng 3.6: Kết quả đánh giá tổng thể các mô hình 55](#_Toc171232822)

**DANH MỤC HÌNH ẢNH**

[Hình 1.1: Input và Output của bài toán phân loại văn bản [2] 4](#_Toc171232552)

[Hình 1.2: Qui trình thực hiện bài toán phân loại văn bản [3] 5](#_Toc171232553)

[Hình 2.1: Mô hình RNN [9] 10](#_Toc171232554)

[Hình 2.2: Minh họa một bước thời gian trong RNN [10] 11](#_Toc171232555)

[Hình 2.3: Mô hình RNN vs LSTM [11] 13](#_Toc171232556)

[Hình 2.4: So sánh giữa RNN và LSTM [8] 13](#_Toc171232557)

[Hình 2.5: Mô hình Bi-LSTM [12] 15](#_Toc171232558)

[Hình 2.6: Sơ đồ kiến trúc Transformer [16] 17](#_Toc171232559)

[Hình 2.7: Biểu diễn từ dưới dạng vector [17] 19](#_Toc171232560)

[Hình 2.8: Mối quan hệ giữa các từ trong câu và attention [18] 20](#_Toc171232561)

[Hình 2.9: Layer normalization [19] 22](#_Toc171232562)

[Hình 2.10: Các vector và hoạt động chuẩn hóa lớp liên quan đến self attention [19] 23](#_Toc171232563)

[Hình 2.11: Cơ chế self-attention Longformer [20] 26](#_Toc171232564)

[Hình 2.12: Full Attention 27](#_Toc171232565)

[Hình 2.13: Sliding Window Attention 27](#_Toc171232566)

[Hình 2.14: Dilated Sliding Window Attention 28](#_Toc171232567)

[Hình 2.15: Global Sliding Window Attention 29](#_Toc171232568)

[Hình 2.16: Cách hoạt động của downstream task [22] 31](#_Toc171232569)

[Hình 2.17: Sơ đồ kiến trúc BERT cho nhiệm vụ MLM [21] 33](#_Toc171232570)

[Hình 2.18: Đầu vào trong nhiệm vụ NSP [21] 34](#_Toc171232571)

[Hình 2.19: Đầu ra trong nhiệm vụ NSP 34](#_Toc171232572)

[Hình 2.20: Kiến trúc của BERT [25] 35](#_Toc171232573)

[Hình 2.21: Pre-trained PhoBERT [26] 37](#_Toc171232574)

[Hình 3.1: Cấu hình máy dùng để train model 39](#_Toc171232575)

[Hình 3.2: Cấu hình dữ liệu ban đầu 40](#_Toc171232576)

[Hình 3.3: Tập dữ liệu train 41](#_Toc171232577)

[Hình 3.4: Tập dữ liệu test 41](#_Toc171232578)

[Hình 3.5: Tập dữ liệu validate 41](#_Toc171232579)

[Hình 3.6: Dữ liệu sau khi tiền xử lý 42](#_Toc171232580)

[Hình 3.7: Ví dụ trích xuất đặc trưng 43](#_Toc171232581)

[Hình 3.8: Xây dựng mô hình Bi-LSTM 44](#_Toc171232582)

[Hình 3.9: Kết quả tối ưu siêu tham số mô hình Bi-LSTM 46](#_Toc171232583)

[Hình 3.10: Biểu đồ quá trình tối ưu hoá siêu tham số mô hình Bi-LSTM 46](#_Toc171232584)

[Hình 3.11: Cài đặt fine-tune PhoBERT và Longformer PhoBERT 47](#_Toc171232585)

[Hình 3.12: Confusion matrix mô hình Bi-LSTM 51](#_Toc171232586)

[Hình 3.13: Confusion matrix PhoBERT 52](#_Toc171232587)

[Hình 3.14: Confusion matrix Longformer PhoBERT 54](#_Toc171232588)

# MỞ ĐẦU

**1. Lý do chọn đề tài**

Trong bất kỳ xã hội nào, con người luôn có nhu cầu giao tiếp và thể hiện bản thân, trong đó ngôn ngữ là phương tiện phổ biến nhất. Ngôn ngữ có thể biểu đạt qua lời nói, chữ viết hoặc các hình ảnh, sử dụng từ ngữ hoặc dấu hiệu để diễn tả ý tưởng. Với sự bùng nổ của Internet và các trang mạng xã hội, hàng ngày có một lượng lớn dữ liệu văn bản được tạo ra từ các nguồn như trang web tài liệu, sách báo, trang sản phẩm, email,...

Để máy tính có thể hiểu và xử lý được khối lượng dữ liệu khổng lồ này, xử lý ngôn ngữ tự nhiên (NLP) đã trở thành một lĩnh vực nghiên cứu quan trọng. NLP nghiên cứu sự tương tác giữa máy tính và ngôn ngữ tự nhiên của con người, sử dụng các kỹ thuật để xử lý và phân tích dữ liệu văn bản, chẳng hạn như mô hình ngôn ngữ và các mô hình dịch máy.

Một trong những nhiệm vụ quan trọng trong NLP là phân loại văn bản, đặc biệt là phân loại bài báo và tin tức tiếng Việt. Khả năng tự động phân loại các bài viết theo chủ đề không chỉ giúp giảm tải công việc cho con người mà còn mở ra nhiều ứng dụng mới cho các lĩnh vực như truyền thông, giáo dục, kinh doanh và quản lý thông tin.

Gần đây, Google AI đã giới thiệu mô hình ngôn ngữ BERT, được coi là một bước đột phá lớn trong học máy vì khả năng ứng dụng của nó vào nhiều bài toán xử lý ngôn ngữ tự nhiên khác nhau với kết quả rất tốt. Tiếp theo đó, PhoBERT ra đời nhằm xây dựng mô hình ngôn ngữ BERT riêng cho tiếng Việt với kết quả tốt nhất cho nhiều bài toán xử lý ngôn ngữ tự nhiên tiếng Việt.

Nhận thấy tầm quan trọng và tiềm năng ứng dụng rộng lớn của việc phân loại bài báo tiếng Việt, nhóm chúng em đã quyết định chọn đề tài **“Phân loại văn bản tiếng Việt bằng mô hình BERT”** cho bài Khóa Luận Tốt Nghiệp của mình.

**2. Mục tiêu và nhiệm vụ nghiên cứu**

* Mục tiêu nghiên cứu: Phân loại văn bản tin tức theo chủ đề dựa trên các nhãn có sẵn. Để đạt được mục tiêu này, nhóm em nghiên cứu và năm vững lý thuyết về các mô hình học sâu tiên tiến trong phân loại văn bản, đặc biệt là BERT và các biến thể của nó như PhoBERT.
* Nhiệm vụ nghiên cứu:
* Tìm hiểu về bài toán phân loại văn bản và các đặc thù của ngôn ngữ tiếng Việt.
* Nghiên cứu lý thuyết và cách triển khai các mô hình quan trọng như BERT và PhoBERT.
* Thu thập và xử lý dữ liệu từ các nguồn tin tức tiếng Việt.
* Huấn luyện và đánh giá hiệu suất các mô hình.
* Thiết kế và phát triển hệ thống phân loại bài báo tiếng Việt với giao diện trực quan, dễ sử dụng.

**3. Đối tượng và phạm vi nghiên cứu**

* Đối tượng nghiên cứu: Các bài báo, tin tức tiếng Việt từ các nguồn trực tuyến như báo điện tử, trang tin tức và mạng xã hội.
* Phạm vi nghiên cứu:
* Phân loại các bài báo và tin tức tiếng Việt theo các chủ đề chính như chính trị, kinh tế, xã hội, thể thao, giải trí, v.v.
* Áp dụng mô hình BERT và các biến thể của nó để phân loại bài báo.
* Chỉ tập trung vào các bài báo tiếng Việt, không mở rộng sang các ngôn ngữ khác.
* Đánh giá mô hình phân loại trên các tập dữ liệu có quy mô và tính chất đa dạng để đảm bảo tính tổng quát và hiệu quả của mô hình.

**NỘI DUNG**

**CHƯƠNG 1: BÀI TOÁN PHÂN LOẠI VĂN BẢN**

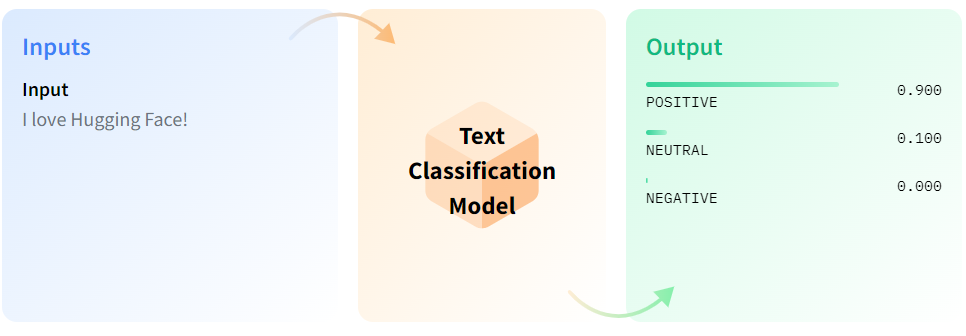
**1.1. TỔNG QUAN VỀ BÀI TOÁN PHÂN LOẠI VĂN BẢN**

Phân loại văn bản - định nghĩa có vẻ đơn giản nhưng lại ẩn chứa sức mạnh to lớn. Nó được xem là một lĩnh vực quan trọng trong xử lý ngôn ngữ tự nhiên (NLP), với mục tiêu gán nhãn các văn bản vào một hoặc nhiều nhóm phân loại đã định trước. Các ứng dụng thực tế của phân loại văn bản bao gồm phân tích cảm xúc (xác định cảm xúc tích cực hoặc tiêu cực trong đánh giá), phát hiện spam (như phát hiện email spam), và phân loại chủ đề (như các bài báo tin tức vào các chủ đề liên quan). Phân loại văn bản đóng vai trò quan trọng trong xử lý ngôn ngữ tự nhiên (NLP) bằng cách cho phép máy tính hiểu và tổ chức các lượng lớn văn bản không có cấu trúc. Điều này đơn giản hóa các nhiệm vụ như lọc nội dung, hệ thống gợi ý, và phân tích phản hồi từ khách hàng.

Các hình thức phân loại văn bản thường gặp bao gồm: [1]

* Phân tích cảm xúc (Sentiment Analysis): Xác định cảm xúc hoặc tình cảm được thể hiện trong một đoạn văn bản, thường phân loại là tích cực, tiêu cực hoặc trung lập. Được sử dụng để phân tích đánh giá sản phẩm, bài đăng trên mạng xã hội và phản hồi từ khách hàng.
* Phát hiện độc hại (Toxicity Detection): Liên quan đến phân tích cảm xúc, nhận diện các ngôn từ xúc phạm hoặc có hại trực tuyến. Giúp các quản trị viên của cộng đồng trực tuyến duy trì môi trường kỳ lịch trong các thảo luận trực tuyến, bình luận hoặc bài đăng trên mạng xã hội.
* Nhận diện ý định (Intent Recognition): Là một phần con của phân tích cảm xúc, được sử dụng để hiểu mục đích (hoặc ý định) đằng sau lời nhập văn bản của người dùng. Các trợ lý ảo và trợ lý ảo thường sử dụng nhận diện ý định để đáp ứng các truy vấn của người dùng.
* Phân loại nhị phân (Binary Classification): Phân loại văn bản thành một trong hai lớp hoặc danh mục. Một ví dụ phổ biến là phát hiện spam, phân loại các văn bản như email hoặc tin nhắn thành loại spam hoặc hợp lệ để tự động lọc bỏ nội dung không mong muốn và có thể gây hại.
* Phân loại đa lớp (Multiclass Classification): Phân loại văn bản thành ba lớp hoặc nhiều hơn. Điều này giúp dễ dàng tổ chức và truy xuất thông tin từ các nội dung như bài báo, bài đăng blog hoặc các nghiên cứu.
* Phân loại chủ đề (Topic Categorization): Liên quan đến phân loại đa lớp, nhóm các tài liệu hoặc bài báo vào các chủ đề hoặc đề tài được xác định trước. Ví dụ, các bài báo có thể được phân loại thành các chủ đề như chính trị, thể thao và giải trí.
* Xác định ngôn ngữ (Language Identification): Xác định ngôn ngữ mà một đoạn văn bản được viết. Điều này hữu ích trong các ngữ cảnh đa ngôn ngữ và các ứng dụng liên quan đến ngôn ngữ.
* Nhận dạng thực thể định danh (Named Entity Recognition): Tập trung vào việc nhận diện và phân loại các thực thể định danh trong văn bản, như tên của người, tổ chức, địa điểm và ngày tháng.
* Phân loại câu hỏi (Question Classification): Xử lý việc phân loại các câu hỏi dựa trên loại câu trả lời dự kiến, có ích cho các công cụ tìm kiếm và hệ thống trả lời câu hỏi.

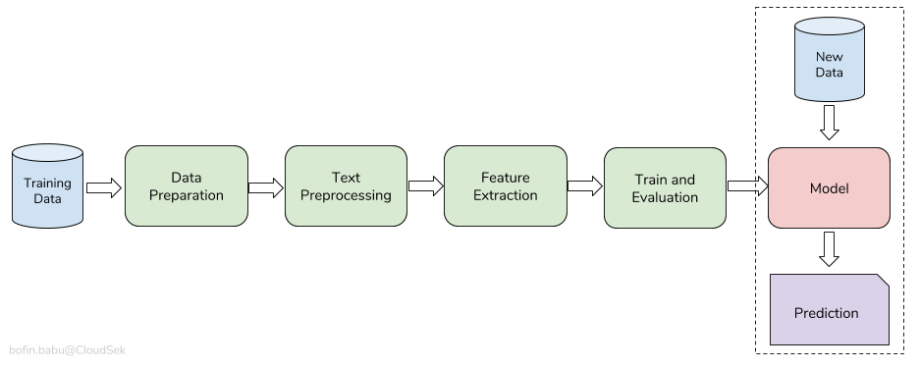
Bài toán phân loại văn bản có đầu vào và đầu ra như sau:



Hình 1.1: Input và Output của bài toán phân loại văn bản [2]

* Đầu vào (Input):
* Văn bản đầu vào:Là các đoạn văn bản, câu, hoặc tài liệu mà chúng ta muốn phân loại vào các lớp hay danh mục đã được xác định trước.
* Dữ liệu ngôn ngữ tự nhiên:Được biểu diễn bằng các từ, cụm từ, hoặc câu, có thể có các dạng khác nhau như tiếng Việt, tiếng Anh, hoặc các ngôn ngữ khác.
* Đầu ra (Output):
* Nhãn hoặc lớp đầu ra: Là các nhãn hoặc lớp mà mô hình phân loại dự đoán cho mỗi đoạn văn bản đầu vào. Ví dụ như chính trị, kinh tế, giải trí, thể thao, y tế, v.v. trong phân loại các bài báo.
* Xác suất phân loại: Một số mô hình có thể đưa ra xác suất cho mỗi lớp, giúp xác định độ chắc chắn của việc phân loại.

Qui trình thực hiện bài toán phân loại văn bản: [1]



Hình 1.2: Qui trình thực hiện bài toán phân loại văn bản [3]

* Thu thập dữ liệu: Thu thập một tập các tài liệu văn bản cùng với các nhãn tương ứng để sử dụng trong quá trình gán nhãn văn bản.
* Tiền xử lý dữ liệu: Chuẩn bị dữ liệu văn bản bằng cách loại bỏ các ký tự không cần thiết, chuyển đổi về chữ thường, và xử lý các ký tự đặc biệt như dấu câu.
* Tách từ (Tokenization): Phân chia văn bản thành các đơn vị nhỏ gọi là “tokens”, ví dụ như các từ. Quá trình này giúp tạo ra các đơn vị có thể tìm kiếm độc lập và hữu ích cho việc tìm kiếm vector và ngữ nghĩa, cung cấp kết quả dựa trên ý định người dùng.
* Trích xuất đặc trưng: Chuyển đổi văn bản thành các biểu diễn số học để mô hình học máy có thể hiểu. Các phương pháp phổ biến bao gồm đếm số lần xuất hiện của từ (gọi là Bag-of-Words) hoặc sử dụng nhúng từ để bắt lấy ý nghĩa của từ.
* Huấn luyện mô hình: Sử dụng dữ liệu đã được tiền xử lý để huấn luyện một mô hình học máy. Mô hình sẽ học các mẫu và mối liên hệ giữa các đặc trưng của văn bản và nhãn tương ứng. Quá trình này giúp mô hình hiểu các quy ước gán nhãn văn bản từ các ví dụ đã được gán nhãn trước đó.
* Đánh giá mô hình: Đánh giá hiệu suất của mô hình đã huấn luyện trong việc phân loại văn bản chưa được nhìn thấy.
* Điều chỉnh siêu tham số (Hyperparameter tuning): Điều chỉnh các thiết lập của mô hình dựa trên kết quả đánh giá để tối ưu hóa hiệu suất của nó.
* Triển khai mô hình (Model deployment): Quá trình sử dụng mô hình đã được huấn luyện và điều chỉnh để phân loại dữ liệu văn bản mới vào các danh mục phù hợp của chúng.

**1.2 BÀI TOÁN PHÂN LOẠI BÀI BÁO VÀ TIN TỨC TIẾNG VIỆT**

Phân loại bài báo và tin tức tiếng Việt là một nhánh cụ thể trong bài toán phân loại văn bản, với các đặc thù riêng về ngôn ngữ và ứng dụng thực tế. Trên Internet hiện nay, nguồn thông tin từ các bài báo và tin tức tiếng Việt ngày càng phong phú và đa dạng. Bài toán đặt ra vấn đề là làm thế nào để tự động phân loại các văn bản này vào các danh mục chủ đề như chính trị, kinh tế, xã hội, thể thao, giải trí, và nhiều lĩnh vực khác. Đây là một bài toán quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP), giúp máy tính hiểu và tổ chức các lượng lớn dữ liệu văn bản một cách tự động và hiệu quả.

**1.2.1 Đặc điểm ngôn ngữ tiếng Việt**

Tiếng Việt có những đặc điểm ngôn ngữ riêng biệt so với các ngôn ngữ khác:

* Tiếng Việt có hệ thống dấu thanh gồm 6 dấu làm tăng độ phức tạp trong việc xử lý ngôn ngữ
* Cấu trúc từ vựng trong Tiếng Việt thường là từ đơn âm hoặc ghép từ đơn âm, tạo ra nhiều cách hiểu khác nhau tuỳ vào ngữ cảnh. Ví dụ, từ “nhà sách” có thể hiểu là nơi bán sách hoặc nhà thuộc về sách, phụ thuộc vào ngữ cảnh sử dụng.
* Về ngữ pháp, Tiếng Việt có cấu trúc linh hoạt và có nhiều điẻm khác biệt so với các ngôn ngữ khác

**1.2.2 Thách thức trong phân loại bài báo, tin tức tiếng Việt**

Phân loại bài báo và tin tức tiếng Việt đặt ra nhiều thách thức. Đầu tiên, các nội dung này mang tính đa dạng về chủ đề và ngữ cảnh, từ chính trị, kinh tế, xã hội, đến thể thao, giải trí và nhiều lĩnh vực khác. Điều này yêu cầu mô hình phân loại phải đủ linh hoạt để nhận diện và phân biệt giữa các loại nội dung khác nhau.

Thứ hai, ngôn ngữ trong các bài viết thường sử dụng các biểu hiện ngôn ngữ tự nhiên, không chính thức, thậm chí là tiếng lóng, làm tăng độ phức tạp trong quá trình phân loại.

Bên cạnh đó, việc thu thập và gán nhãn dữ liệu tiếng Việt chất lượng để huấn luyện mô hình cũng là một thách thức lớn, đặc biệt là với số lượng dữ liệu lớn và độ phức tạp của ngôn ngữ tự nhiên.

**1.2.3 Phương pháp tiếp cận và mô hình ứng dụng cho bài toán**

Các mô hình học sâu như CNNs và RNNs, thường được áp dụng cho dữ liệu hình ảnh và dữ liệu tuần tự, hiện đang được sử dụng để phân loại văn bản [1]. Đặc biệt, PhoBERT và Longformer PhoBERT đã được phát triển để giải quyết các thách thức đặc thù của ngôn ngữ tiếng Việt. Những mô hình này có khả năng:

* Hiểu ngữ cảnh tiếng Việt: Nắm bắt được ngữ cảnh và mối quan hệ giữa các từ trong câu tiếng Việt.
* Học từ dữ liệu tiếng Việt: Học từ các tập dữ liệu tiếng Việt lớn, cải thiện độ chính xác và khả năng tổng quát hóa.
* Tự động hóa trích xuất đặc trưng: Tự động trích xuất các đặc trưng từ văn bản tiếng Việt, giảm thiểu sự can thiệp của con người.

Trong khóa luận này, nhóm sẽ tiếp cận vấn đề theo hai hướng chính:

* Sử dụng PhoBERT như một công cụ trích xuất đặc trưng: Đóng băng tất cả các lớp của PhoBERT và sử dụng nó như một bộ trích xuất đặc trưng. Sau đó, huấn luyện một hoặc nhiều bộ phân loại dựa trên các đặc trưng này với các nhãn đã biết. Phương pháp này sử dụng các trạng thái ẩn cuối cùng của PhoBERT để trích xuất đặc trưng. Các trích xuất đặc trưng này tạo ra một ma trận đặc trưng để xây dựng các tập huấn luyện, kiểm tra và xác thực.
* Fine-tuning PhoBERT: Tiến hành điều chỉnh lại các tham số của PhoBERT thông qua quá trình fine-tuning. Quá trình này cho phép điều chỉnh các trọng số của PhoBERT trên tập dữ liệu huấn luyện cụ thể, giúp cải thiện hiệu suất của mô hình cho bài toán phân loại văn bản tiếng Việt. [4]

**1.3 CÁC CÔNG TRÌNH ỨNG DỤNG LIÊN QUAN TỚI BÀI TOÁN PHÂN LOẠI VĂN BẢN TIẾNG VIỆT**

**1.3.1 PhoBERT: Pre-trained language models for Vietnamese**

Trong nghiên cứu này, Dat Quoc Nguyen từ VinAI Research (Việt Nam) và Anh Tuan Nguyen từ NVIDIA (Hoa Kỳ) đã giới thiệu mô hình ngôn ngữ PhoBERT, là mô hình tiền huấn luyện đầu tiên dành riêng cho tiếng Việt, bao gồm hai phiên bản PhoBERTbase và PhoBERTlarge.

Nghiên cứu của họ tập trung vào việc so sánh hiệu suất của PhoBERT so với mô hình đa ngôn ngữ hàng đầu hiện nay là XLM-R, cũng như đánh giá PhoBERT trên các nhiệm vụ như gán nhãn từ loại, phân tích cú pháp, nhận dạng thực thể tên và suy luận ngôn ngữ tự nhiên.

Kết quả cho thấy, PhoBERT không chỉ vượt trội hơn so với XLM-R trên các nhiệm vụ tiếng Việt mà còn đạt được kết quả state-of-the-art trong các lĩnh vực đánh giá. Ngoài ra, PhoBERT đã được công khai để phục vụ cho cộng đồng nghiên cứu và ứng dụng tiếng Việt trong tương lai. [5]

**1.3.2 Vietnamese News Articles Classification Using Neural Networks**

Tác giả của nghiên cứu này gồm có Nguyễn Phước Vinh và Hà Hoàng Kha từ Đại học Bách Khoa Thành phố Hồ Chí Minh và Đại học Quốc gia Thành phố Hồ Chí Minh.

Mục tiêu của họ là giới thiệu một bộ dữ liệu mới về các bài báo tin tức trực tuyến tiếng Việt được phân loại cho các nhiệm vụ đa nhãn. Đồng thời, họ cải tiến quy trình phân loại văn bản tiếng Việt bằng cách sử dụng cắt giảm chiều của các vector đặc trưng dựa trên tần suất thuật ngữ, kết hợp với đánh giá trọng số TF-IDF, thay vì áp dụng các thuật toán lựa chọn đặc trưng sau khi trích xuất vector đặc trưng có số chiều lớn.

Kết quả của nghiên cứu này là thành công trong xây dựng một bộ dữ liệu mẫu cho phân loại tin tức trực tuyến tiếng Việt và đạt được hiệu suất phân loại tốt hơn nhờ áp dụng mô hình mạng nơ-ron mạnh mẽ trong quá trình phân tích. [6]

**1.3.3 Universal language model fine-tuning for text classification**

Bài báo này được viết bởi Jeremy Howard từ fast.ai và University of San Francisco, và Sebastian Ruder từ Insight Centre, NUI Galway và Aylien Ltd., Dublin

 Mục tiêu chính của bài báo là đề xuất một phương pháp mới gọi là "Universal Language Model Fine-tuning" (ULMFiT) để thực hiện chuyển giao học tập (transfer learning) hiệu quả cho các tác vụ xử lý ngôn ngữ tự nhiên (NLP), tương tự như cách chuyển giao học tập được áp dụng rộng rãi trong thị giác máy tính.

Đã đề xuất được phương pháp ULMFiT và các kỹ thuật mới như "discriminative fine-tuning", "slanted triangular learning rates", và "gradual unfreezing" để giữ được kiến thức trước đây và tránh "catastrophic forgetting" trong quá trình fine-tuning. Phương pháp ULMFiT vượt trội so với các phương pháp tiên tiến hiện tại trên 6 tập dữ liệu phân loại văn bản, giảm lỗi từ 18-24% trên hầu hết các tập dữ liệu.Phương pháp ULMFiT cho phép chuyển giao học tập với rất ít dữ liệu nhãn, đạt được hiệu suất tương đương với việc huấn luyện từ đầu với 100 lần nhiều dữ liệu nhãn hơn. Ý nghĩa chính của bài báo là đề xuất một phương pháp chuyển giao học tập hiệu quả có thể áp dụng cho bất kỳ tác vụ NLP nào, tương tự như các phương pháp chuyển giao học tập đã được áp dụng rộng rãi trong thị giác máy tính. [30]

**1.3.4 Improving Imbalanced Text Classification with Dynamic Curriculum Learning**

Tác giả của công trình nghiên cứu này là Xulong Zhang, Jianzong Wang, Ning Cheng, Jing Xiao từ Ping An Technology (Shenzhen) Co.,[10]

Mục tiêu của họ là đề xuất một phương pháp học theo chương trình động (self-paced dynamic curriculum learning - SPDCL) để cải thiện hiệu suất phân loại văn bản không cân bằng. Phương pháp này sử dụng cả đặc điểm ngôn ngữ và khả năng của mô hình để đánh giá độ khó của mẫu (sample). Thay vì sử dụng phương pháp học theo chương trình tĩnh như các nghiên cứu trước đó, SPDCL có thể sắp xếp lại và lấy mẫu lại dữ liệu đào tạo theo tiêu chí độ khó với một nhịp độ thích ứng từ dễ đến khó.

Kết quả thực nghiệm trên nhiều tác vụ phân loại cho thấy hiệu quả của chiến lược SPDCL rất tốt, đặc biệt là với tập dữ liệu không cân bằng. [31]

# CHƯƠNG 2: CÁC MÔ HÌNH SỬ DỤNG

**2.1 MÔ HÌNH BI-LSTM**

**2.1.1 Mô hình RNN (Recurrent Neural Network)**

*2.1.1.1 Tổng quan về RNN*

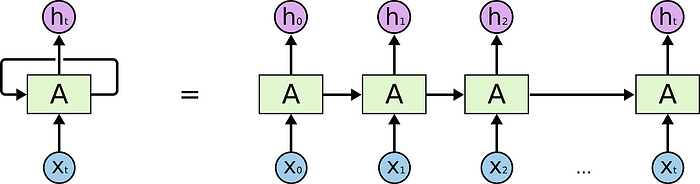
Mạng nơ-ron truy hồi (RNN) là một loại mạng nơ-ron cực kỳ quan trọng, được thiết kế chuyên biệt để xử lý thông tin dạng chuỗi. Khả năng ghi nhớ và sử dụng các mẫu dữ liệu trước đó giúp RNN dự đoán kết quả chính xác hơn. [7] [8]

Khác với các mạng neural thông thường, RNN xử lý dữ liệu theo trình tự thời gian. Điều này rất quan trọng khi dữ liệu có tính chất trình tự, vì thay đổi thứ tự dữ liệu có thể dẫn đến kết quả khác nhau. Ví dụ, hai câu “Bạn chưa đi làm” và “Bạn đi làm chưa” nếu tách ra thành từ và tạo bộ từ vựng [“Bạn”, “chưa”, “đi”, “làm”], sau đó mã hóa one-hot và đưa vào mạng neural thông thường, sẽ không có sự phân biệt nào giữa hai câu này, do việc đảo lộn thứ tự các từ làm sai lệch ý nghĩa của câu.

Để giải quyết vấn đề này, RNN xử lý dữ liệu theo trình tự, bằng cách đưa đầu vào một cách tuần tự, cho phép mạng neural hiểu và ghi nhớ thứ tự của các từ trong câu. Điều này giúp RNN phân biệt và xử lý chính xác hơn các câu có ý nghĩa khác nhau mặc dù sử dụng cùng một tập hợp từ vựng.

*2.1.1.2 Cách hoạt động của RNN*

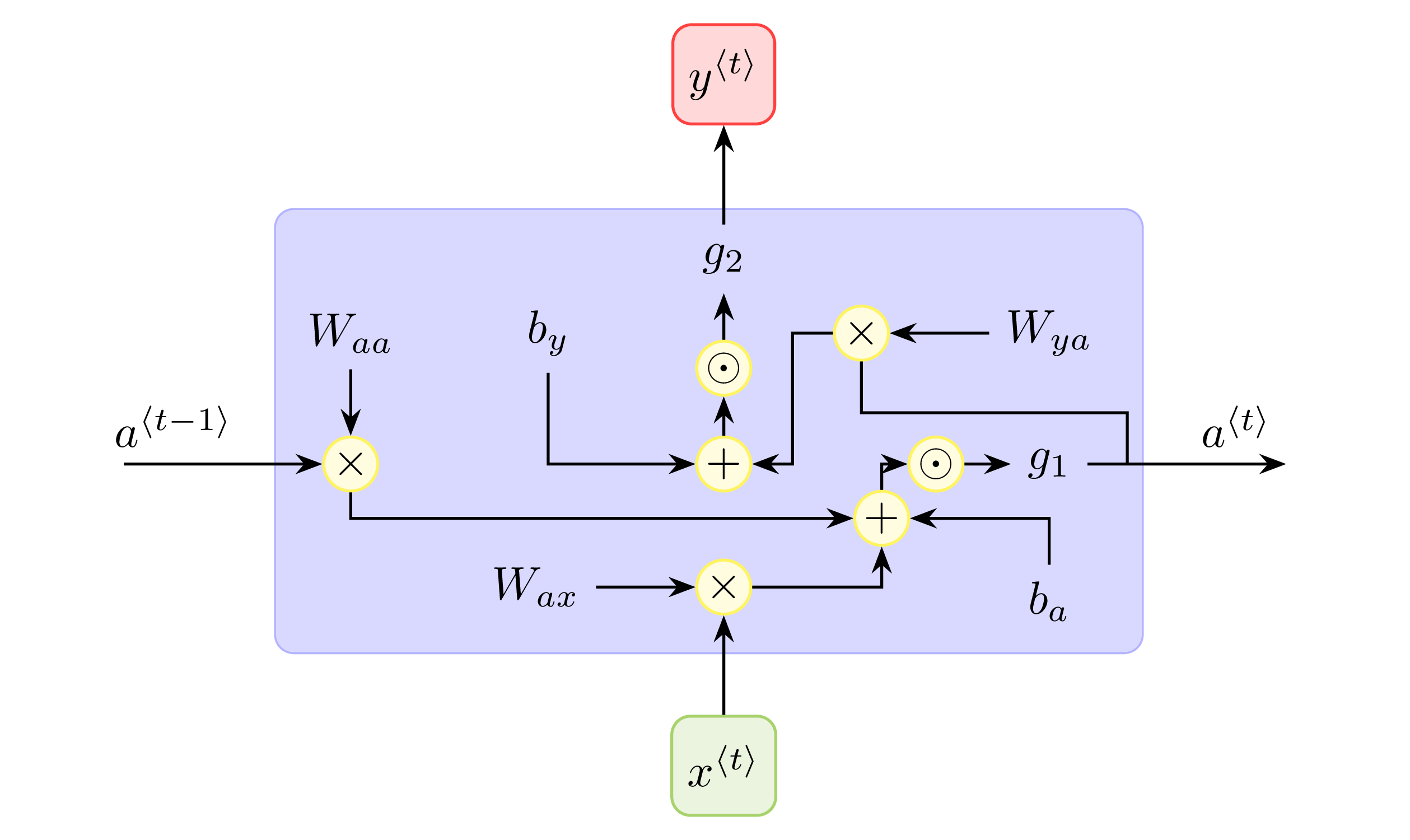
Dưới đây là hình ảnh thể hiện các hoạt động của RNN



Hình 2.1: Mô hình RNN [9]

Mỗi khối RNN sẽ nhận thông tin từ các khối trước đó cùng với dữ liệu đầu vào hiện tại. Các ở đây đại diện cho dữ liệu đầu vào theo từng bước thời gian (time step). biểu thị cho bước thời gian thứ , và là đầu ra của một bước. Ví dụ, sẽ là vector đại diện cho từ thứ hai trong câu văn bản. [8]

Hình ảnh dưới đây minh họa rõ ràng điều gì thực sự diễn ra trong một bước thời gian.



Hình 2.2: Minh họa một bước thời gian trong RNN [10]

Ta có biểu thức như sau:

=​ (​∗ ​ +∗ +​) [8]

Trong đó:

* ​: Là hidden state tại thời điểm .
* ​: Là hàm kích hoạt được áp dụng cho đầu ra của bộ tổ hợp tuyến tính. Thông thường, ​ có thể là hàm tanh, sigmoid, hoặc ReLU, được sử dụng để đưa ra hidden state có giá trị nằm trong một khoảng xác định và giúp mạng học được các mối quan hệ phi tuyến tính.
* ​: Là ma trận trọng số ánh xạ từ hidden state trước đó đến hidden state hiện tại ​. Đây là ma trận quan trọng để mạng có thể học và lưu trữ thông tin từ các bước thời gian trước.
* : Là ma trận trọng số ánh xạ từ input tại thời điểm hiện tại đến hidden state . Đây là cách mà mạng sử dụng thông tin từ input để cập nhật hidden state.
* ​: Là vector độ lệch (bias) được thêm vào sau khi tính tổng tuyến tính ​∗ ​ +∗ +​. Vector này giúp điều chỉnh và tạo sự linh hoạt cho mạng nơ-ron trong quá trình học và dự đoán.

Cách hoạt động của biểu thức này là cập nhật hidden state của mạng RNN dựa trên thông tin từ hidden state trước đó ​ và đầu vào hiện tại ​, qua đó giúp mạng RNN có khả năng xử lý và học dữ liệu tuần tự một cách liên tục và có tính liên kết.

Output của mỗi bước thời gian ​ trong mạng nơ-ron truy hồi (RNN) có hai phần chính. Ta có, là tổng hợp thông tin từ các trạng thái trước để truyền tiếp trong chuỗi mạng. Đồng thời, ​ là kết quả đầu ra của bước thời gian hiện tại, thường được áp dụng hàm softmax để tính toán xác suất của các lớp đầu ra.

*​= (​*∗ *​*+ *​)*

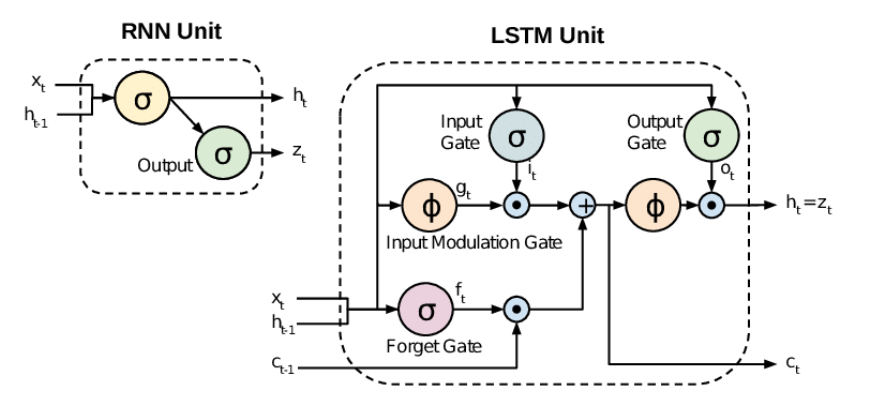
Trong quá trình huấn luyện, chúng ta cần tối ưu hóa ba tham số là ​, ​ và . Để làm điều này, chúng ta cần tính đạo hàm của hàm mất mát L theo từng tham số này, tức là ​, và . Khi tính toán đạo hàm của hàm mất mát đối với các tham số ​ trong mạng RNN, chúng ta phải sử dụng chuỗi các hidden state ​, ​, … từ các bước trước. Điều này dẫn đến hiện tượng gọi là vấn đề **gradient vanishing** hoặc **gradient exploding**. [8]

**Gradient vanishing** xảy ra khi gradient giảm dần đáng kể khi lan truyền ngược qua các bước time step, đặc biệt là khi các giá trị của gradient nhỏ hơn 1. Điều này có nghĩa là các tham số của mô hình không được cập nhật một cách hiệu quả và mô hình không học được mối quan hệ dài hạn trong dữ liệu chuỗi.

**Gradient exploding**, ngược lại, xảy ra khi gradient tăng quá nhanh khi lan truyền ngược, dẫn đến sự không ổn định trong quá trình học mô hình.

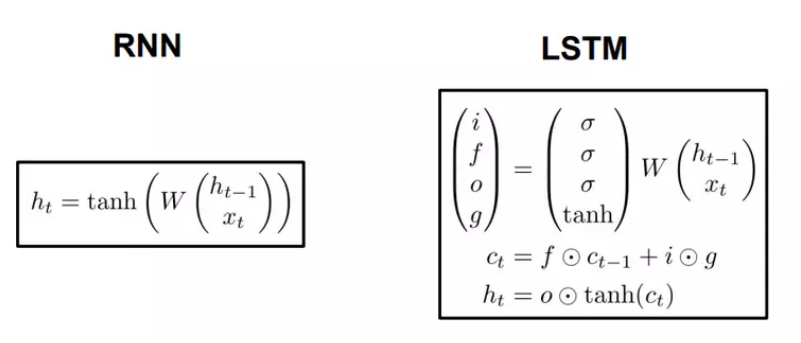
Có thể thấy RNN thuần không có cơ chế để loại bỏ thông tin không cần thiết trong quá trình xử lý dữ liệu. Để giải quyết vấn đề này, ta cần áp dụng một kiến trúc tiến bộ hơn, có khả năng lưu trữ thông tin dài hạn hơn và duy trì gradient ổn định hơn trong quá trình huấn luyện, đó là LSTM (Long Short-Term Memory).

**2.1.2 Mô hình LSTM (Long Short-Term Memory)**



Hình 2.3: Mô hình RNN vs LSTM [11]

Về cơ bản, ý tưởng của RNN và LSTM không khác nhau nhiều. Chúng ta chỉ thêm một số tính toán như dưới đây:



Hình 2.4: So sánh giữa RNN và LSTM [8]

Đầu tiên, mạng LSTM bao gồm ba cổng chính: cổng vào (input gate) , cổng quên (forget gate) , và cổng ra (output gate) . Các cổng này có công thức rất tương tự nhau và chỉ khác nhau ở ma trận tham số. Ma trận này quyết định vai trò và chức năng của từng cổng. Hàm sigmoid σ được sử dụng để quyết định mức độ thông tin nào sẽ được truyền qua các cổng này: nếu đầu ra của hàm sigmoid là 0, toàn bộ vector thông tin sẽ bị loại bỏ, và nếu là 1, hầu hết thông tin sẽ được giữ lại. [8]

Cụ thể:

* **Cổng vào (Input gate)** : Quyết định lượng thông tin từ đầu vào sẽ ảnh hưởng đến trạng thái mới . Cổng này điều chỉnh mức độ thông tin sẽ được lưu lại.
* **Cổng quên (Forget gate)** : Quyết định phần nào của thông tin từ trạng thái trước sẽ bị loại bỏ.
* **Cổng ra (Output gate)** : Điều chỉnh lượng thông tin từ trạng thái hiện tại sẽ được truyền đến đầu ra và sang trạng thái tiếp theo .

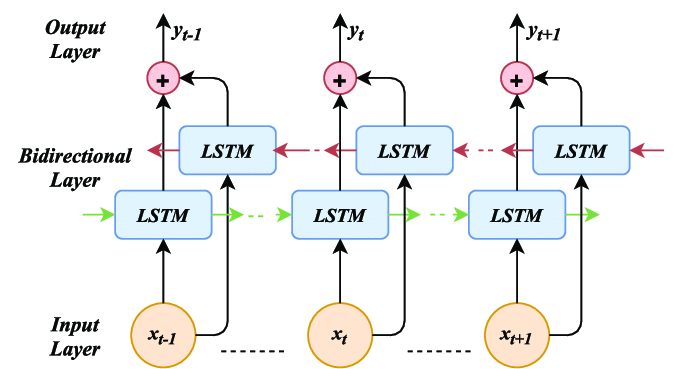
Tiếp theo, trạng thái ẩn mới được tính dựa trên đầu vào hiện tại và trạng thái ẩn trước . Công thức tính tương tự như cổng vào, nhưng sử dụng hàm tanh thay vì sigmoid. Trạng thái ẩn này giúp cập nhật và duy trì trạng thái bộ nhớ mới.

Cuối cùng, là bộ nhớ dài hạn của LSTM, được tính bằng cách tổng hợp bộ nhớ trước đó đã được lọc qua cổng quên , cộng với trạng thái ẩn đã được lọc bởi cổng vào . Bộ nhớ này chứa thông tin quan trọng và sẽ được sử dụng khi cần thiết để giảm thiểu vấn đề vanishing gradient.

RNN truyền thống là một trường hợp đặc biệt của LSTM khi giá trị đầu ra của cổng vào là 1 và giá trị đầu ra của cổng quên là 0, cho phép nó duy trì trạng thái ngắn hạn.

LSTM đã cải tiến đáng kể so với RNN bằng cách sử dụng cơ chế cổng để giải quyết vấn đề vanishing gradient, tuy nhiên vẫn có hạn chế trong việc xử lý ngữ cảnh toàn cục do tính chất unidirectional. Để giải quyết hạn chế này và cải thiện khả năng hiểu ngữ cảnh toàn cục, Bi-LSTM (Bidirectional Long Short-Term Memory) ra đời với khả năng tính toán từ cả hai hướng của chuỗi, giúp tăng tính linh hoạt và hiệu quả trong nhiều ứng dụng thực tế.

**2.1.3 Mô hình Bi-LSTM (Bidirectional Long Short-Term Memory)**



Hình 2.5: Mô hình Bi-LSTM [12]

Bi-LSTM (Bidirectional Long Short-Term Memory) là một loại mạng nơ-ron tái phát (RNN) được thiết kế để xử lý dữ liệu tuần tự theo cả hai hướng, trước và sau. Nó kết hợp sức mạnh của LSTM với khả năng xử lý hai chiều, cho phép mô hình nắm bắt được cả ngữ cảnh trước và sau của chuỗi đầu vào. [12]

Để hiểu về Bi-LSTM, chúng ta cùng phân tích các thành phần và chức năng của nó:

* LSTM (Long Short-Term Memory): Đây là một loại mạng nơ-ron có khả năng lưu giữ thông tin lâu dài và bỏ qua thông tin không cần thiết. Điều này giúp nó hiểu và xử lý các mối quan hệ dài hạn trong dữ liệu.
* Xử lý hai chiều (Bidirectional Processing): Khác với RNN truyền thống chỉ xử lý dữ liệu theo một hướng (trước hoặc sau), Bi-LSTM xử lý chuỗi đầu vào cùng lúc theo cả hai hướng. Nó bao gồm hai lớp LSTM: một lớp xử lý chuỗi theo hướng chuyển tiếp và một lớp khác xử lý theo hướng lùi. Mỗi lớp duy trì các trạng thái ẩn và ô nhớ riêng.
* Chuyển tiếp (Forward Pass): Trong quá trình chuyển tiếp, chuỗi đầu vào được đưa vào lớp LSTM theo hướng chuyển tiếp từ bước thời gian đầu tiên đến cuối. Tại mỗi bước thời gian, LSTM chuyển tiếp tính toán trạng thái ẩn của mình và cập nhật ô nhớ dựa trên đầu vào hiện tại, trạng thái ẩn trước đó và ô nhớ trước đó.
* Lùi (Backward Pass): Đồng thời, chuỗi đầu vào cũng được đưa vào lớp LSTM theo hướng ngược lại, từ bước thời gian cuối cùng đến đầu tiên. Tương tự như chuyển tiếp, LSTM lùi tính toán trạng thái ẩn của mình và cập nhật ô nhớ dựa trên đầu vào hiện tại, trạng thái ẩn trước đó và ô nhớ trước đó.
* Kết hợp trạng thái từ cả hai hướng: Sau khi hoàn thành chuyển tiếp và lùi, các trạng thái ẩn từ cả hai lớp LSTM được kết hợp tại mỗi bước thời gian. Phương pháp kết hợp này có thể đơn giản như ghép các trạng thái ẩn hoặc áp dụng một phép biến đổi khác. [12]

Kiến trúc Bi-LSTM có nhiều lợi ích trong các vấn đề thực tế, đặc biệt là trong xử lý ngôn ngữ tự nhiên (NLP). Điểm chính là mỗi thành phần của một chuỗi đầu vào đều có thông tin từ cả quá khứ và tương lai. Nhờ vậy, Bi-LSTM có thể tạo ra đầu ra có ý nghĩa hơn, đặc biệt trong việc xây dựng các mô hình ngôn ngữ, vì các từ trong một đoạn văn thường có liên quan cả với từ trước và từ sau. [13]

Ví dụ, trong câu *“Paris là …”*, từ *“Paris”* có thể là nói về tên một ai đó hoặc một thành phố. LSTM truyền thống sẽ không thể hiểu được nghĩa của *“Paris”* vì nó không có thông tin về ngữ cảnh phía sau.

Ngược lại, trong hai câu sau đây:

*“Paris là một thành phố xinh đẹp.”* và

*“Paris là một kỹ sư AI.”*

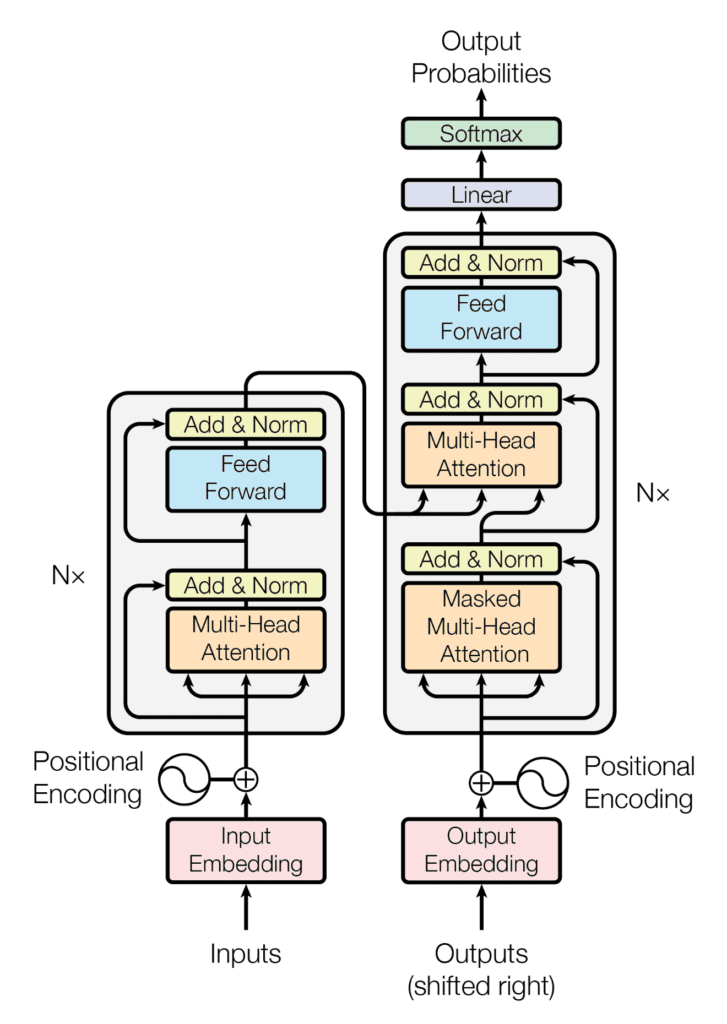
Bi-LSTM có thể làm tốt việc phân biệt *“Paris”* là một kỹ sư hay *“Paris”* là một thành phố, bằng cách sử dụng thông tin từ ngữ cảnh phía sau.

Vì vậy, có thể thấy rõ rằng mô hình Bi-LSTM có lợi trong nhiều nhiệm vụ NLP như phân loại câu, dịch thuật, và nhận diện thực thể. Ngoài ra, nó còn được ứng dụng trong nhận dạng giọng nói, dự đoán cấu trúc protein, nhận dạng chữ viết tay, và các lĩnh vực tương tự. [13]

Tuy nhiên, Bi-LSTM có thể tốn nhiều thời gian hơn để huấn luyện so với các mô hình RNN một chiều hay LSTM. Điều này do BiLSTM cần phải xử lý dữ liệu theo cả hai hướng (chuyển tiếp và lùi) đồng thời, dẫn đến việc tăng thời gian huấn luyện và phức tạp hóa mô hình. Ngoài ra, sự phức tạp về tính toán cũng là một vấn đề khiến cho Bi-LSTM không phù hợp với các ứng dụng yêu cầu tốc độ xử lý nhanh và tiết kiệm tài nguyên tính toán. [13] [14]

**2.2 TRANSFORMER**

Trước khi Google công bố bài báo về Transformers (Attention Is All You Need), hầu hết các tác vụ xử lý ngôn ngữ tự nhiên, đặc biệt là dịch máy, thường sử dụng kiến trúc Recurrent Neural Networks (RNNs). Tuy nhiên, phương pháp này có điểm yếu là khó bắt được sự phụ thuộc xa giữa các từ trong câu và tốc độ huấn luyện chậm do phải xử lý input theo thứ tự tuần tự. Transformers ra đời để giải quyết hai vấn đề này, và các biến thể như BERT, GPT-2 đã mang đến những tiến bộ đáng kể cho các nhiệm vụ liên quan đến xử lý ngôn ngữ tự nhiên. [15]



Hình 2.6: Sơ đồ kiến trúc Transformer [16]

**2.2.1 Mô hình Sequence-to-sequence trong RNN**

Mô hình Sequence-to-Sequence nhận đầu vào là một chuỗi và trả về đầu ra là một chuỗi khác, ví dụ như trong bài toán Q&A, nơi đầu vào là câu hỏi *“How are you?”* và đầu ra là câu trả lời *“I am good”.* Phương pháp truyền thống sử dụng RNNs cho cả phần mã hóa (encoder) và phần giải mã (decoder). Tuy nhiên, RNNs gặp hai vấn đề chính. Đầu tiên, là tốc độ huấn luyện chậm, yêu cầu sử dụng Truncated Backpropagation để có thể huấn luyện được. Mặc dù đã áp dụng biện pháp này, tốc độ huấn luyện vẫn chậm do không thể tận dụng tính toán song song trên GPU.

Thứ hai là RNNs không xử lý tốt với các câu dài do vấn đề Gradient Vanishing/Exploding. Với số lượng đơn vị (units) lớn, gradient giảm dần ở các đơn vị cuối do hiệu ứng đạo hàm chuỗi, dẫn đến mất thông tin hoặc sự phụ thuộc xa giữa các đơn vị. LSTM đã giải quyết được vấn đề Gradient Vanishing nhưng nó lại phức tạp hơn RNN rất nhiều và thời gian train cũng chậm hơn đáng kể.

Như vậy có cách nào tận dụng khả năng tính toán song song của GPU để tăng tốc độ huấn luyện cho các mô hình ngôn ngữ, đồng thời khắc phục vấn đề xử lý các câu dài không? Transformers là giải pháp cho cả hai vấn đề này.

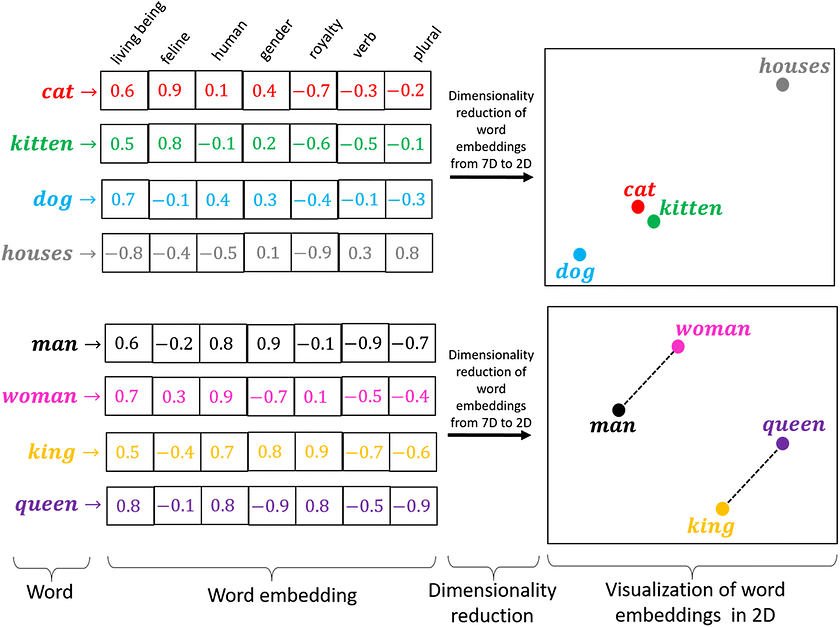
**2.2.2 Decoder và Encoder**

*2.2.2.1 Input Embedding*

Máy tính không thể hiểu trực tiếp câu chữ, mà chỉ có khả năng xử lý các số liệu, vectors, và ma trận. Do đó, cần biểu diễn câu chữ dưới dạng vector, được gọi là input embedding. Việc này đảm bảo rằng các từ có nghĩa gần nhau sẽ có vector tương đồng. Hiện nay, có nhiều phương pháp embedding từ đã được tiền huấn luyện như GloVe, Fasttext, và gensim Word2Vec để lựa chọn.

Ví dụ với từ *“vua”* và *“hoàng đế”*, cả hai từ này có ý nghĩa tương đồng nên vector biểu diễn của chúng trong không gian vector sẽ gần nhau. Ngược lại, từ *“vua”* và *“mèo”* có ý nghĩa khác biệt, do đó vector biểu diễn của chúng sẽ cách xa nhau trong không gian vector:

* *“vua”*: [0.2, 0.4, 0.5, ...]
* *“hoàng đế”*: [0.22, 0.41, 0.48, ...]
* *“mèo”*: [0.75, 0.88, 0.65, ...]

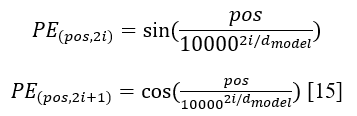


Hình 2.7: Biểu diễn từ dưới dạng vector [17]

*2.2.2.2 Positional Encoding*

Word embeddings phần nào giúp biểu diễn ngữ nghĩa của một từ, tuy nhiên cùng một từ ở các vị trí khác nhau trong câu có thể mang ý nghĩa khác nhau. Do đó, Transformers sử dụng thêm một thành phần gọi là Positional Encoding để cung cấp thông tin về vị trí của một từ trong câu. [15]

Công thức Positional Encoding được biểu diễn như sau:



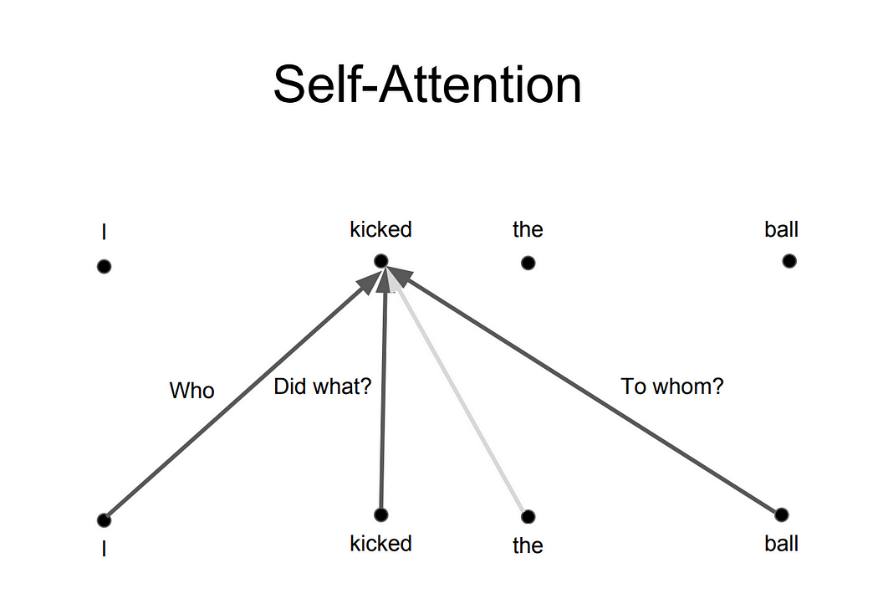
Trong đó, là vị trí của từ trong câu, là giá trị của phần tử thứ trong embeddings có độ dài ​. Sau khi tính toán Positional Encoding, chúng ta cộng vector với vector embedding của từ đó.

Giả sử câu *“Tôi yêu học máy”* với các từ *“Tôi”*, *“yêu”*, *“học”*, và *“máy”* lần lượt ở các vị trí 1, 2, 3, và 4. Vector embedding của từ “Tôi” có thể là [0.1, 0.3, 0.5, ...]. Sau khi áp dụng Positional Encoding, vector của từ “Tôi” tại vị trí 1 sẽ được điều chỉnh để bao gồm thông tin về vị trí, tạo ra một vector mới, chẳng hạn [0.15, 0.35, 0.55, ...].

Như vậy, Positional Encoding giúp mô hình Transformers không chỉ hiểu ngữ nghĩa của từ mà còn nắm bắt được vị trí của từ trong câu, từ đó cải thiện hiệu quả xử lý ngôn ngữ tự nhiên.

*2.2.2.3 Self-Attention*

Self-Attention là cơ chế cho phép Transformers “hiểu” được mối liên hệ giữa các từ trong một câu. Ví dụ, trong câu *“I kicked the ball”,* từ *“kicked”* có mối liên hệ như thế nào với các từ khác? Rõ ràng, nó liên quan mật thiết đến từ *“I”* (chủ ngữ), từ *“kicked”* là chính nó nên sẽ luôn có “liên kết mạnh”, và từ *“ball”* (tân ngữ). Trong khi đó, từ *“the”* là mạo từ nên sự liên kết với từ *“kicked”* gần như không có. Vậy Self-Attention trích xuất những mối liên hệ này như thế nào?



Hình 2.8: Mối quan hệ giữa các từ trong câu và attention [18]

Trong kiến trúc tổng thể của Transformer, các module Multi-head Attention (bản chất là Self-Attention) nhận đầu vào là 3 vectors: Query (Q), Key (K) và Value (V). Từ 3 vectors này, ta sẽ tính vector attention Z cho một từ theo công thức sau:



Công thức này được thực hiện như sau. Đầu tiên, để có được 3 vectors Q, K, V, input embeddings được nhân với 3 ma trận trọng số tương ứng (được tune trong quá trình huấn luyện) , và .

Lúc này, vector K đóng vai trò như một khóa đại diện cho từ, và Q sẽ truy vấn đến các vector K của các từ trong câu bằng cách nhân chập với những vector này. Phép nhân chập này nhằm tính toán độ liên quan giữa các từ với nhau, theo đó, hai từ liên quan sẽ có “Score” lớn và ngược lại.

Bước thứ hai là bước “Scale”, đơn giản chỉ là chia “Score” cho căn bậc hai của số chiều của Q/K/V. Điều này giúp cho giá trị “Score” không phụ thuộc vào độ dài của vector Q/K/V.

Bước thứ ba là áp dụng hàm softmax lên các kết quả để đạt được một phân bố xác suất trên các từ.

Bước thứ tư, ta nhân phân bố xác suất đó với vector V để loại bỏ những từ không cần thiết (xác suất nhỏ) và giữ lại những từ quan trọng (xác suất lớn).

Cuối cùng, các vector V (đã được nhân với softmax output) sẽ được cộng lại với nhau, tạo ra vector attention Z cho một từ. Lặp lại quá trình này cho tất cả các từ trong câu, ta sẽ có được ma trận attention cho cả câu.

*2.2.2.4 Multi-head Attention*

Vấn đề của Self-attention là attention của một từ sẽ luôn “chú ý” vào chính nó. Điều này rất hợp lý vì rõ ràng “nó” phải liên quan đến “nó” nhiều nhất. Ví dụ, trong câu *“Anh ấy ăn một quả táo”*, từ “ăn” sẽ có attention cao nhất đối với chính nó. [15]

Tuy nhiên, chúng ta không mong muốn điều này. Điều mà chúng ta thực sự cần là sự tương tác giữa các từ khác nhau trong câu. Để giải quyết vấn đề này, các nhà nghiên cứu đã giới thiệu một phiên bản nâng cấp hơn của Self-attention, gọi là Multi-head attention. Ý tưởng rất đơn giản: thay vì chỉ sử dụng một Self-attention (1 head), chúng ta sử dụng nhiều Attention khác nhau (multi-head). Mỗi Attention có thể chú ý đến một phần khác nhau trong câu. [15]

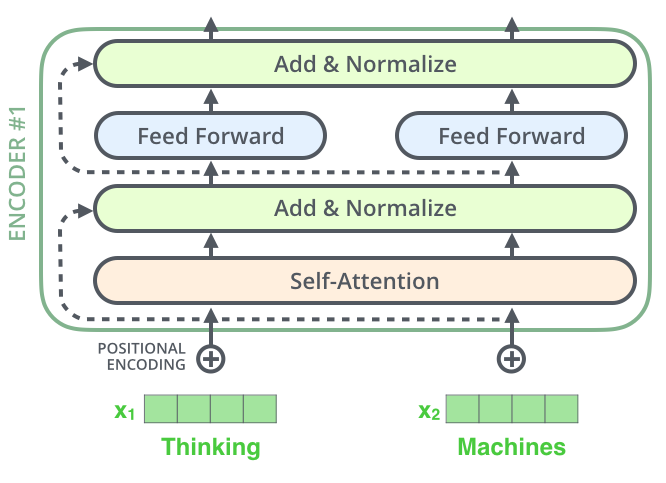
Vì mỗi “head” sẽ cho ra một ma trận attention riêng, nên chúng ta phải kết hợp (concat) các ma trận này và nhân với ma trận trọng số ​ để tạo ra một ma trận attention duy nhất (weighted sum). Và tất nhiên, ma trận trọng số này cũng được điều chỉnh (tune) trong quá trình huấn luyện. [15]

Ví dụ, trong câu *“Chị ấy mua một quyển sách mới”*, một head có thể chú ý đến mối quan hệ giữa *“mua”* và *“quyển sách”*, trong khi một head khác lại chú ý đến mối quan hệ giữa *“chị ấy”* và *“mới”.* Khi kết hợp các ma trận attention này, chúng ta sẽ có một cái nhìn tổng quan hơn về sự tương tác giữa các từ trong câu.

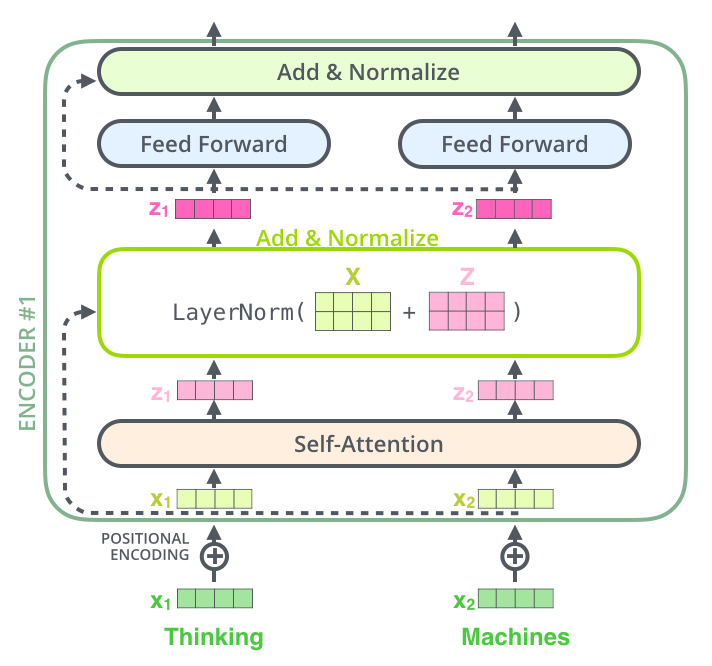
Multi-head attention giúp mô hình nắm bắt được nhiều khía cạnh khác nhau của ngữ cảnh, giúp tăng cường khả năng hiểu biết của mô hình đối với ngữ nghĩa của câu. Điều này làm cho mô hình trở nên mạnh mẽ và hiệu quả hơn trong việc xử lý các tác vụ ngôn ngữ tự nhiên.

*2.2.2.5 Residuals*

Một chi tiết trong kiến trúc của bộ mã hóa mà chúng ta cần đề cập trước khi tiến xa hơn, đó là mỗi sub-layer (self-attention, ffnn) trong mỗi bộ mã hóa đều có một kết nối dư được bao quanh và sau đó là một bước chuẩn hóa lớp (layer-normalization). [19]



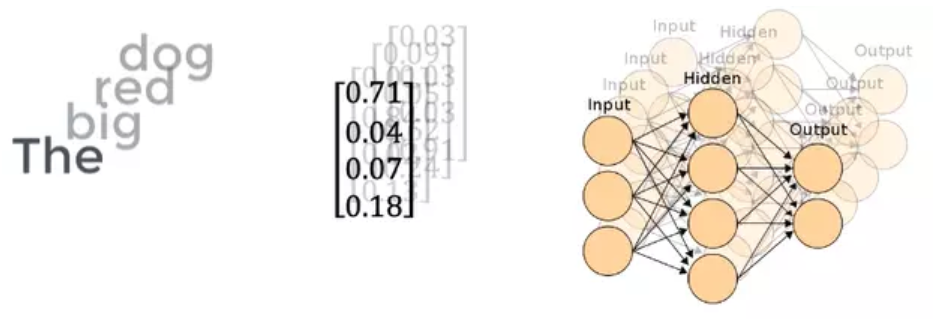
Hình 2.9: Layer normalization [19]



Hình 2.10: Các vector và hoạt động chuẩn hóa lớp liên quan đến self attention [19]

*2.2.2.6 Feed Forward*

Sau khi được chuẩn hóa, các vector được truyền qua một mạng fully connected trước khi đi vào Decoder. Vì các vector này độc lập với nhau, do đó ta có thể tận dụng tính toán song song cho toàn bộ câu. [15]



Hình 2.11: Mạng fully connected trước khi đi vào decoder [15]

*2.2.2.7 Masked Multi-head Attention*

Trong trường hợp muốn Transformers thực hiện bài toán dịch từ tiếng Anh sang tiếng Pháp, nhiệm vụ của Decoder là giải mã thông tin từ Encoder và tạo ra từng từ tiếng Pháp dựa trên từ đã được dịch trước đó. Do đó, nếu ta áp dụng Multi-head attention cho toàn bộ câu giống như ở Encoder, Decoder sẽ có thể “nhìn thấy” từ tiếp theo mà nó cần dịch. Để ngăn chặn điều này, khi Decoder đang dịch từ thứ iii, phần sau của câu tiếng Pháp sẽ được “che đi” (masked), và Decoder chỉ có thể “nhìn thấy” phần của câu mà nó đã dịch trước đó. [15]

*2.2.2.8 Quá trình Decode*

Quá trình giải mã (decode) cơ bản tương tự như quá trình mã hóa (encode) trong Transformers, nhưng có một số khác biệt quan trọng. Đầu tiên, input của Decoder là câu tiếng Pháp bị masked. Sau khi masked input được đưa qua sub-layer #1 của Decoder, nó sẽ không nhân với ba ma trận trọng số để tạo ra Q, K, V như trong multi-head attention của Encoder, mà chỉ nhân với một ma trận trọng số. Các ma trận K và V vẫn được lấy từ Encoder, còn Q được tạo ra từ masked multi-head attention và được đưa vào sub-layer #2 và #3 tương tự như trong Encoder. [15]

Cuối cùng, các vector kết quả từ các sub-layer này sẽ được đưa qua một lớp Linear (một mạng Fully Connected) và sau đó thông qua hàm Softmax để tính toán xác suất của từ tiếp theo.

Giả sử chúng ta có câu tiếng Anh *“I love reading books”* và muốn dịch sang tiếng Việt. Quá trình giải mã (decode) trong Transformers sẽ diễn ra như sau:

* **Masked Input**: Input của Decoder là câu tiếng Việt đã bị mask. Ví dụ, khi chúng ta đang dịch từ *“I love”*, từ *“love”* sẽ là từ đang được xử lý, và các từ *“I”* đã dịch trước đó sẽ bị mask để không ảnh hưởng đến quá trình dịch tiếp theo.
* **Sub-layer #1 - Masked Multi-head Attention**: Input sau khi mask được đưa qua sub-layer này. Ở đây, không có việc tạo ra các ma trận Q, K, V như trong multi-head attention của Encoder. Thay vào đó, chỉ sử dụng ma trận trọng số để tính toán.
* **Sub-layer #2 và #3 - Multi-head Attention và Feed Forward Network (FFN)**: Các ma trận K và V vẫn được lấy từ Encoder, và Q được tạo ra từ masked multi-head attention. Các vector kết quả từ sub-layer này được đưa qua một lớp Linear để tính toán xác suất của từ tiếp theo trong câu tiếng Việt.

**2.3 LONGFORMER**

**2.3.1 Tổng quan về Longformer**

Longformers là các mạng nơ-ron được thiết kế đặc biệt để xử lý và hiểu các chuỗi văn bản hoặc dữ liệu dài. Chúng có khả năng xử lý các chuỗi và tài liệu rất dài với hàng ngàn từ mà không gặp phải các thách thức tính toán như Transformer. [20]

Mặc dù các mô hình dựa trên Transformer có khả năng hiểu và xử lý ngôn ngữ tốt, nhưng chúng gặp khó khăn khi phải xử lý các đoạn văn bản dài. Nguyên nhân là do chúng sử dụng *Self-Attention*. Khi số lượng từ tăng lên, phép toán này trở nên phức tạp và chậm hơn nhiều. Điều này giống như việc phải xem xét mỗi từ và xác định mối quan hệ của nó với mọi từ khác, việc này tốn rất nhiều thời gian và tài nguyên tính toán.

Để giải quyết vấn đề này, các nhà nghiên cứu đã phát triển một biến thể của Transformer gọi là *Longformer*. Longformer được thiết kế để hoạt động hiệu quả với các đoạn văn bản cực kỳ dài bằng cách thay đổi cách thức chú ý đến các từ.

Thay vì cố gắng xem xét mọi từ trong toàn bộ văn bản cùng một lúc, Longformer sử dụng sự kết hợp của hai loại attention. Đầu tiên, nó tập trung chú ý vào các từ lân cận, giống như đọc một đoạn văn hay một trang sách một lần, giúp tăng tốc độ xử lý. Sau đó, nó cũng xem xét bức tranh tổng thể, cố gắng hiểu cách các phần khác nhau của văn bản liên kết với nhau. Sự kết hợp này giữa chú ý cận cảnh và chú ý tổng thể giúp Longformer hoạt động hiệu quả với các tài liệu rất dài, như những tài liệu có hàng ngàn từ hoặc nhiều hơn.

Ví dụ, khi sử dụng Longformer để dịch một bài báo dài từ tiếng Anh sang tiếng Việt. Thay vì phải xem xét toàn bộ bài báo một cách chi tiết, Longformer sẽ tập trung vào từng đoạn văn nhỏ trước, xác định ngữ cảnh và ý nghĩa của chúng. Sau đó, nó sẽ liên kết các đoạn văn này lại để hiểu tổng thể nội dung bài báo. Việc này giúp Longformer xử lý bài báo dài một cách hiệu quả và chính xác hơn, không gặp phải các vấn đề về tính toán và tốc độ như các mô hình Transformer truyền thống.

**2.3.2 Quy mô bậc hai trong Self-Attention**

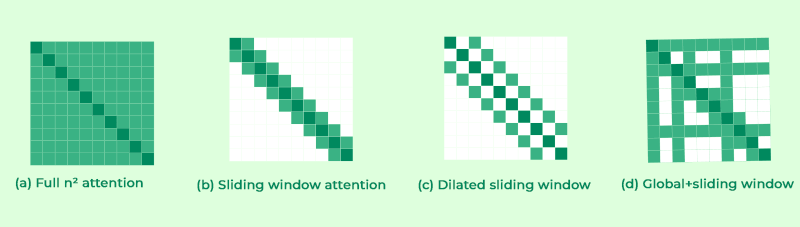
Để tính toán vector attention của một từ, chúng ta cần thực hiện phép nhân giữa các vector và , sau đó áp dụng và nhân kết quả của với vector . Đối với một từ như *“I”* trong câu *“I love you”*, phép nhân này cần được thực hiện với mỗi từ khác trong câu. Vì vậy, một từ sẽ có phép toán.

Việc tính toán vector attention cho mỗi từ trong câu sẽ yêu cầu thực hiện phép toán cho mỗi từ. Vì có từ trong câu, nên cần phải thực hiện lần tính toán này. Do đó, chúng ta cần thực hiện phép toán cho mỗi câu có độ dài , điều này làm cho quy mô phép toán tăng theo bậc hai.

Sự tăng quy mô bậc hai của các phép toán trong transformer theo kích thước đầu vào làm cho nó trở nên không hiệu quả khi xử lý các câu hoặc tài liệu dài. Điều này không chỉ tiêu tốn nhiều bộ nhớ mà còn làm chậm quá trình xử lý. Mô hình BERT chuẩn chỉ có thể xử lý tối đa 512 token. Bất kỳ tài liệu dài nào có hơn 512 từ đều phải bị cắt ngắn hoặc chia nhỏ, dẫn đến mất thông tin hoặc lỗi lan truyền.

Đây chính là lúc mô hình longformer được giới thiệu. Nó giúp tăng quy mô tuyến tính với kích thước đầu vào, cho phép xử lý lượng token gấp 4 lần so với transformer thông thường. Điều này giúp longformer có thể xử lý các đoạn văn bản dài hơn một cách hiệu quả mà không gặp phải các vấn đề về bộ nhớ và tốc độ như trước. [20]

**2.3.3 Cơ chế Attention trong Longformer**



Hình 2.11: Cơ chế self-attention Longformer [20]

*2.3.3.1 Full Attention*

Đây là cơ chế attention toàn phần cơ bản trong kiến trúc transformer như đã thảo luận ở trên. Tại đây, tất cả các ô đều màu xanh lá cây, cho thấy rằng đối với mỗi từ, tất cả các từ khác đều được sử dụng để tính toán cơ chế attention. Do đó, số lượng phép toán là (theo tỷ lệ bậc hai). Ví dụ, attention toàn phần có thể được biểu diễn trực quan như sau:



Hình 2.12: Full Attention

Trong ví dụ này, mỗi từ (Tôi, thích, học hỏi, tại, OpenAI) đều được tính toán cơ chế attention với tất cả các từ khác trong câu. Mỗi ô xanh lá cây biểu thị rằng cơ chế attention được tính toán giữa các từ tương ứng.

*2.3.3.2 Sliding Window Attention*

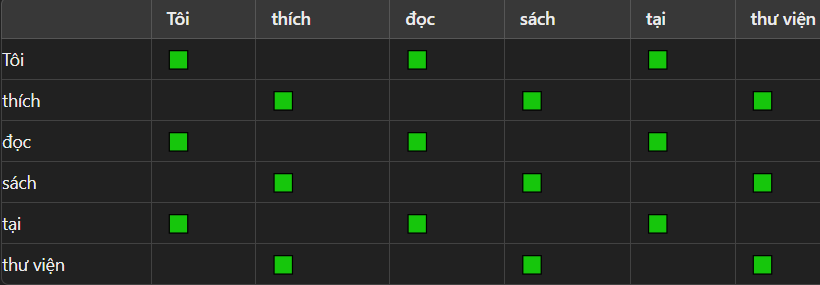
Trong cơ chế attention dạng cửa sổ trượt (sliding window), chúng ta không tính toán attention của mỗi từ đối với tất cả các từ khác. Thay vào đó, chúng ta chọn một cửa sổ có độ dài và tính toán cơ chế attention trong phạm vi độ dài của cửa sổ này. Ví dụ, giả sử chúng ta có một cửa sổ có độ dài là 2. Vậy trong câu ***“****Tôi thích đọc sách tại thư viện****”***, đối với từ ***“****đọc****”*** (từ thứ ba), chúng ta sẽ tính toán vector attention chỉ với các từ ***“****thích****”*** và ***“****sách****”****.* Độ phức tạp tính toán của cửa sổ attention này là, tức là tỷ lệ thuận với độ dài chuỗi đầu vào . Cửa sổ trượt có thể được biểu diễn trực quan như sau:



Hình 2.13: Sliding Window Attention

*2.3.3.3 Dilated Sliding Window Attention*

Cơ chế attention dạng cửa sổ trượt giãn cách (Dilated Sliding Window Attention) còn được gọi là attention dạng cửa sổ rời rạc (sparse window attention). Trong cơ chế này, chúng ta bỏ qua một số từ nhất định, tạo ra các khoảng trống trong các mẫu kết nối. Nó tương tự như cửa sổ trượt, nhưng ở đây chúng ta chỉ chú ý đến các từ có khoảng cách trong một cửa sổ có độ dài (được chỉ định bởi các ô trắng giữa các ô xanh). Điều này cho phép tăng khả năng nhận thức mà không làm tăng yêu cầu về bộ nhớ. [20]



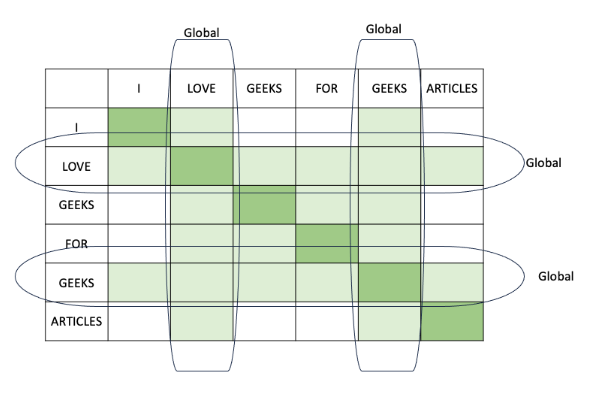
Hình 2.14: Dilated Sliding Window Attention

*2.3.3.4 Global Sliding Window Attention*

Global Sliding Window Attention là một sự điều chỉnh của cơ chế cửa sổ trượt trong đó cho phép một số từ hoặc token truy cập vào tất cả các token khác để tính toán vector attention (được chỉ định bởi các đường ngang và dọc màu xanh lá cây). Cơ chế tự chú ý của transformer hoạt động trên cả ngữ cảnh “cục bộ” và “toàn cầu”. Trong kiến trúc Longformer, hầu hết các token chú ý “cục bộ” đến nhau trong một kích thước cửa sổ nhất định. Các token xem xét các token trước và sau đó trong ngữ cảnh cục bộ này. Một vài token được chọn có khả năng chú ý “toàn cầu” đến tất cả các token khác trong chuỗi. Những token chú ý toàn cầu này có khả năng xem xét thông tin từ toàn bộ chuỗi, khác với việc giới hạn trong một kích thước cửa sổ cụ thể.

Điều quan trọng cần lưu ý là trong thiết kế của Longformer, mỗi token tham gia cục bộ không chỉ xem xét các token trong cửa sổ của nó mà còn chú ý đến tất cả các token tham gia toàn cầu. Điều này đảm bảo rằng chú ý toàn cầu là đối xứng.

Global Sliding Window Attention đạt được một sự cân bằng trung gian, cung cấp một sự điều chỉnh giữa hiệu quả tính toán và khả năng mô hình hóa. Điều này làm cho nó phù hợp cho các nhiệm vụ yêu cầu sự cân bằng tinh tế giữa quản lý tài nguyên tính toán và việc thu thập thông tin ngữ cảnh trong một cửa sổ ngữ cảnh hạn chế. [20]



Hình 2.15: Global Sliding Window Attention

**2.4 MÔ HÌNH BERT**

**2.4.1 Tổng quan về BERT**

BERT (Bidirectional Encoder Representations from Transformers) là một nghiên cứu mới được công bố bởi các nhà nghiên cứu tại Google AI Language. Nghiên cứu này đã gây sốt trong cộng đồng Học máy bằng cách đạt được những kết quả hàng đầu trong nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên (NLP), bao gồm Trả lời câu hỏi (SQuAD v1.1), Suy luận Ngôn ngữ Tự nhiên (MNLI), và nhiều nhiệm vụ khác.

Điểm đột phá chính của BERT là áp dụng huấn luyện hai chiều của mô hình Transformer, một mô hình attention phổ biến, cho mô hình hóa ngôn ngữ. Điều này khác với những nỗ lực trước đó chỉ nhìn vào một chuỗi văn bản từ trái sang phải hoặc kết hợp huấn luyện từ trái sang phải và từ phải sang trái. Kết quả của nghiên cứu cho thấy rằng một mô hình ngôn ngữ được huấn luyện hai chiều có thể hiểu sâu hơn về ngữ cảnh và cấu trúc ngôn ngữ so với các mô hình chỉ huấn luyện một chiều. [21]

**2.4.2 Tại sao lại cần BERT?**

Một trong những thách thức lớn nhất của NLP là vấn đề dữ liệu. Trên internet có rất nhiều dữ liệu, nhưng chúng không đồng nhất. Mỗi phần lại chỉ được sử dụng cho một mục đích cụ thể. Do đó, khi giải quyết một bài toán cụ thể, chúng ta phải lựa chọn một bộ dữ liệu phù hợp và thường chỉ có ít dữ liệu. Ví dụ, trong mô hình *OpenAI GPT*, kiến trúc left-to-right được sử dụng, có nghĩa là các từ chỉ phụ thuộc vào các từ đứng trước đó.

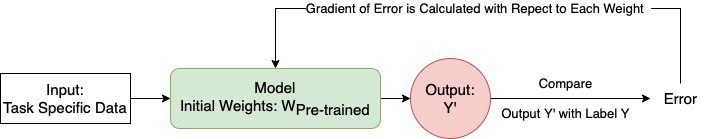
Tuy nhiên, mô hình Deep Learning lại cần một lượng dữ liệu rất lớn - hàng triệu mẫu - để có thể đạt được kết quả tốt. Vì vậy, một vấn đề quan trọng được đặt ra là làm thế nào để tận dụng được nguồn dữ liệu khổng lồ có sẵn để giải quyết bài toán của mình. Đó là lý do cho sự xuất hiện của một kỹ thuật mới *Transfer Learning*. Với *Transfer Learning*, các mô hình được “tiền huấn luyện” (pre-training) trên dữ liệu lớn trên internet, và sau đó có thể được “tinh chỉnh” (fine-tune) cho các bài toán cụ thể.

Nhờ vào kỹ thuật này, kết quả cho các bài toán đã được cải thiện đáng kể, không chỉ trong xử lý ngôn ngữ tự nhiên mà còn trong nhiều lĩnh vực khác như *Computer Vision*,... BERT là một ví dụ nổi bật trong *Transfer Learning* cho xử lý ngôn ngữ tự nhiên, nổi bật không chỉ vì kết quả ấn tượng trong nhiều bài toán khác nhau mà còn bởi sự miễn phí của nó, cho phép ai cũng có thể sử dụng BERT cho bài toán của mình.

**2.4.3 Một số khái niệm**

*2.4.3.1 Nhiệm vụ phía sau (Downstream task)*

Downstream task là quá trình áp dụng mô hình đã được huấn luyện trước đó vào một vấn đề mới. Nó dựa vào kết quả của một nhiệm vụ hoặc quá trình trước đó làm đầu vào. Bằng cách tinh chỉnh (fine-tune) mô hình sẵn có trên các nhiệm vụ cụ thể, chúng ta có thể cải thiện hiệu suất của mô hình và làm cho nó hữu ích hơn cho các ứng dụng thực tế. Điều này giúp chúng ta không phải xây dựng các mô hình từ đầu mà tận dụng và cải thiện các mô hình đã có để giải quyết các vấn đề mới.



Hình 2.16: Cách hoạt động của downstream task [22]

Trong quá trình fine-tune, chúng ta sử dụng các tham số của mô hình đã được huấn luyện trước làm giá trị ban đầu. Sau đó, chúng ta điều chỉnh các tham số dựa trên đạo hàm của hàm lỗi để tối ưu hóa hiệu suất của mô hình trên nhiệm vụ mới. Sau khi mô hình đã được điều chỉnh, chúng ta có thể sử dụng nó để đưa ra dự đoán hoặc thực hiện các nhiệm vụ khác liên quan đến downstream task. [22]

Ví dụ, đối với các mô hình ngôn ngữ lớn như GPT và BERT, chúng ta đầu tiên huấn luyện mô hình trên lượng lớn dữ liệu văn bản tổng quát để học ngôn ngữ. Sau đó, chúng ta điều chỉnh lại mô hình để thực hiện các nhiệm vụ cụ thể như phân tích cảm xúc, nhận diện thực thể (NER - Name Entity Recognition), phân loại văn bản, trả lời câu hỏi, tóm tắt văn bản, v.v

*2.4.3.2 Ngữ cảnh (Contextual) và vai trò trong NLP*

Ngôn ngữ cơ bản là âm thanh con người phát ra để diễn đạt suy nghĩ. Trong giao tiếp, từng từ không tồn tại độc lập mà thường kết hợp với những từ khác để tạo thành câu hoàn chỉnh. Việc này giúp hiệu quả biểu thị nội dung và truyền đạt ý nghĩa lớn hơn so với việc xử lý từng từ riêng lẻ. [23]

Ví dụ, từ *“đồng”* trong hai câu sau mang hai ý nghĩa khác nhau:

* Câu A: Đơn vị tiền tệ của Việt Nam là *“đồng”.*
* Câu B: Vợ *“đồng”* ý với ý kiến của chồng là tăng thêm mỗi tháng 500k tiền tiêu vặt.

Trong trường hợp này, từ “đồng” cần phải có hai biểu diễn từ khác nhau để phản ánh ý nghĩa của nó trong mỗi ngữ cảnh khác nhau. Các thuật toán không có ngữ cảnh như WORD2VEC và FASTTEXT không thể đáp ứng được sự đa dạng về nghĩa của từ trong NLP. [23]

**2.4.4 Cách hoạt động của BERT**

BERT được thiết kế để tạo ra một mô hình ngôn ngữ, do đó chỉ cơ chế bộ mã hóa (encoder) được sử dụng. Chuỗi các token được đưa vào bộ mã hóa Transformer. Những token này được nhúng thành các vector và sau đó được xử lý trong mạng nơ-ron. Đầu ra là một chuỗi các vector, mỗi vector tương ứng với một token đầu vào, cung cấp các biểu diễn ngữ cảnh.

Trong quá trình huấn luyện mô hình ngôn ngữ, việc định nghĩa mục tiêu dự đoán là một thách thức. Nhiều mô hình dự đoán từ tiếp theo trong chuỗi, là một hướng tiếp cận hướng đạo lộn (directional) và có thể hạn chế việc học ngữ cảnh. BERT giải quyết vấn đề này với hai chiến lược huấn luyện: [24]

* Masked Language Model (Dự đoán từ thiếu trong câu)
* Next Sentence Prediction (Dự đoán câu tiếp theo)

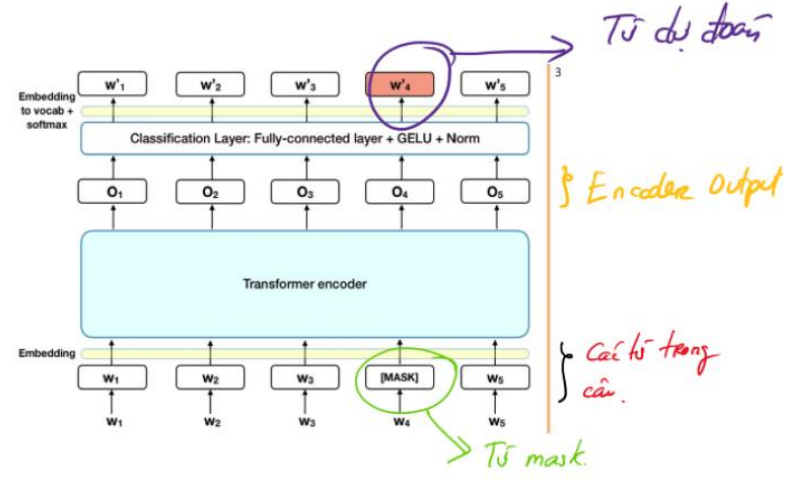
*2.4.4.1 Masked Language Model (MLM)*

Đối với kỹ thuật MLM, trước khi BERT học từ các câu, nó sẽ che giấu một số từ (khoảng 15%) và thay thế chúng bằng một ký hiệu đặc biệt, chẳng hạn như [MASK].

Sau đó, BERT sẽ cố gắng dự đoán chính xác các từ bị che giấu này dựa trên ngữ cảnh của các từ xung quanh. Ta có thể hình dung đây như một trò chơi đoán từ, khi một số từ bị thiếu và BERT cố gắng điền vào những từ đó.

Để làm điều này, BERT thêm một lớp đặc biệt lên đầu của nó để thực hiện các dự đoán. Nó sẽ đánh giá xem dự đoán của mình về từ bị che giấu có gần đúng với từ thực sự hay không. Quá trình này giúp BERT hiểu được ý nghĩa và ngữ cảnh của từng từ trong câu một cách sâu sắc hơn.

Về mặt kỹ thuật, BERT sử dụng một lớp phân loại để dự đoán các từ bị che giấu. Các dự đoán này sau đó được biến đổi thành xác suất, giúp cho BERT có thể đưa ra dự đoán chính xác hơn về từng từ. Mô hình chỉ quan tâm đến việc dự đoán các từ bị che giấu, và điều này giúp nó học được ngữ cảnh một cách hiệu quả hơn, mặc dù quá trình học có thể diễn ra chậm hơn so với mô hình chỉ hướng đạo. [24]



Hình 2.17: Sơ đồ kiến trúc BERT cho nhiệm vụ MLM [21]

Giả sử chúng ta có câu: *“I want to buy a car.”*

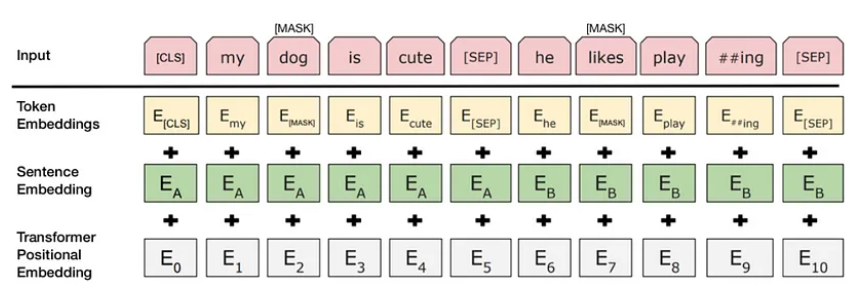
Trong quá trình huấn luyện, BERT sẽ ngẫu nhiên che giấu một vài từ trong câu. Câu trên có từ *“buy”* bị che giấu. BERT sẽ cố gắng dự đoán từ bị che giấu này dựa trên ngữ cảnh của các từ còn lại trong câu.

* Câu ban đầu: *“I want to buy a car.”*
* Sau khi che giấu: *“I want to [MASK] a car.”*

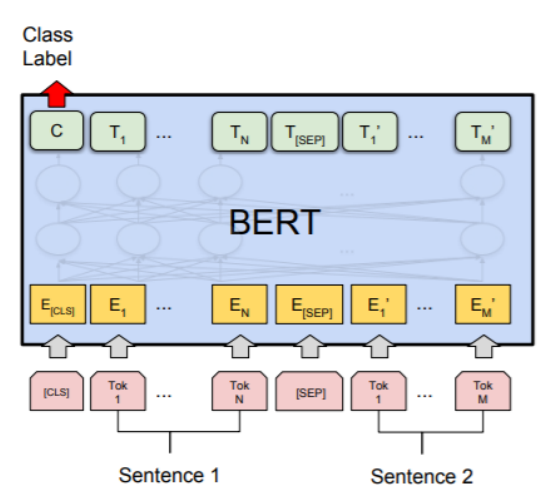
BERT sẽ nhìn vào *“I”*, *“want”*, *“to”*, *“a”*, *“car”* để dự đoán từ bị che giấu. Trong trường hợp này, nếu BERT dự đoán từ *“buy”* cho từ bị che giấu, thì đó là một dự đoán chính xác và mô hình sẽ được cập nhật dựa trên mức độ chính xác của dự đoán này.

*2.4.4.2 Next Sentence Prediction (NSP)*

* Trong quá trình huấn luyện BERT, mô hình nhận đầu vào là các cặp câu và học để dự đoán liệu câu thứ hai có phải là câu tiếp theo trong tài liệu gốc so với câu đầu tiên hay không. Quá trình này được thực hiện như sau:
* Mỗi cặp câu được ghép lại và mã đặc biệt được thêm vào: [CLS] ở đầu câu thứ nhất và [SEP] ở cuối mỗi câu. Ví dụ: *“Hôm nay em đi học”* và *“Học ở trường rất vui”* sẽ được ghép thành ***[CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP].***
* Mỗi từ trong câu được gắn thêm một nhúng câu (Sentence Embedding) để chỉ ra từ thuộc câu thứ nhất hay câu thứ hai. Ví dụ, nếu từ thuộc câu thứ nhất thì sẽ có nhúng câu là “0”, và nếu thuộc câu thứ hai thì sẽ có nhúng câu là “1”.
* Mỗi từ cũng được thêm vào một véc tơ mã hóa vị trí (Positional Encoding) để xác định vị trí của từ trong chuỗi.
* Sau đó, chuỗi được đưa vào mạng BERT để huấn luyện.
* Đầu ra từ mạng sẽ là véc tơ tại mã [CLS], được biến đổi thành một véc tơ có hai phần tử [c1 c2].
* Cuối cùng, áp dụng hàm softmax lên véc tơ này để tính toán xác suất của hai lớp: “Đi sau” và “Không đi sau”. Argmax của véc tơ này sẽ xác định xem câu thứ hai có đi sau câu thứ nhất hay không.



Hình 2.18: Đầu vào trong nhiệm vụ NSP [21]



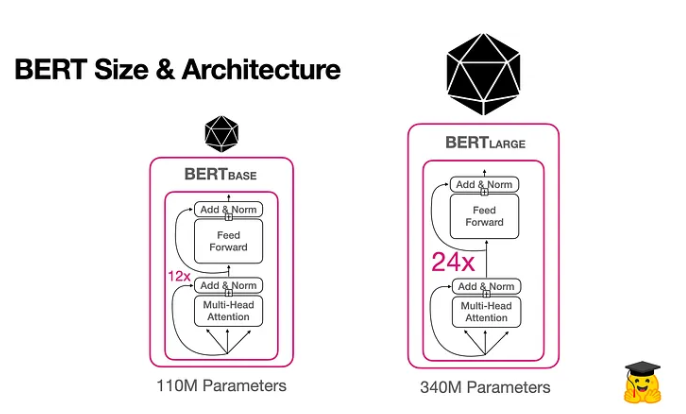
Hình 2.19: Đầu ra trong nhiệm vụ NSP

Thông tin đầu vào trước khi được đưa vào mô hình huấn luyện BERT bao gồm ba thành phần chính:

* Nhúng từ (Token Embeddings): Đây là những véc tơ biểu diễn cho từng từ trong câu, được khởi tạo từ mô hình huấn luyện trước. Mỗi từ trong câu được biểu diễn bằng một véc tơ đặc biệt, giúp mô hình hiểu ý nghĩa của từ đó.
* Nhúng loại câu (Segment Embeddings): Đây là hai véc tơ và , tương ứng với câu thứ nhất và câu thứ hai. Điều này giúp mô hình phân biệt và hiểu rõ từ thuộc câu nào trong cặp câu đang xử lý.
* Nhúng vị trí (Position Embedding): Đây là các véc tơ , , ..., thể hiện vị trí của từng từ trong câu. Đây cũng là một phần quan trọng trong kiến trúc Transformer, giúp mô hình nhận biết được thứ tự và vị trí của các từ trong câu.

Các véc tơ này được tổng hợp lại để tạo thành đầu vào cho mô hình, giúp BERT hiểu và xử lý các mối quan hệ ngữ nghĩa giữa các từ và câu trong văn bản.

**2.4.5 Các kiến trúc mô hình BERT**



Hình 2.20: Kiến trúc của BERT [25]

BERT có kiến trúc là một bộ mã hóa transformer đa tầng, hoạt động theo hướng đa chiều, tương tự như mô hình transformer. Kiến trúc transformer gồm một mạng mã hóa-giải mã sử dụng tự chú ý ở phần mã hóa và chú ý ở phần giải mã.

sử dụng 12 tầng trong bộ mã hóa, trong khi có 24 tầng. Đây là nhiều hơn so với mô hình Transformer ban đầu mô tả (6 tầng mã hóa). Cả và đều có các mạng lan truyền tiến lớn hơn (lần lượt là 768 và 1024 đơn vị ẩn) và nhiều đầu chú ý hơn (12 và 16 đầu chú ý). Điều này khác biệt so với mô hình Transformer gốc, chỉ có 512 đơn vị ẩn và 8 đầu chú ý. Số lượng tham số của là 110 triệu, trong khi là 340 triệu. [24]

**2.4.6 Mô hình BERT trong phân loại văn bản**

BERT có thể được sử dụng cho các nhiệm vụ phân loại như phân tích cảm xúc, nơi mà mục tiêu là phân loại văn bản vào các danh mục khác nhau (tích cực/ tiêu cực/ trung lập). Để áp dụng BERT vào nhiệm vụ này, chúng ta thêm một lớp phân loại lên đầu đầu ra của Transformer cho token [CLS].

Token [CLS] đại diện cho thông tin tổng hợp từ toàn bộ chuỗi đầu vào. Biểu diễn này có thể được sử dụng làm đầu vào cho một lớp phân loại để dự đoán cho nhiệm vụ cụ thể. [24]

**2.5 MÔ HÌNH PHOBERT VÀ LONGFORMER PHOBERT**

**2.5.1 Sự ra đời của PhoBERT**

BERT là một nghiên cứu mới của Google AI, đã đem lại đột phá đáng kể trong lĩnh vực xử lý ngôn ngữ tự nhiên. Nó đã nổi lên như là một trong những mô hình tiên tiến nhất trong nhiều lĩnh vực, bao gồm Question Answering và Sentiment Analysis.

Tuy nhiên, việc huấn luyện mô hình BERT cho Tiếng Việt không đơn giản. Mặc dù Google đã có mô hình huấn luyện trước đa ngôn ngữ (pre-trained multilingual), bao gồm cả Tiếng Việt, nhưng hiệu quả vẫn chưa được chứng minh tốt nhất.

BERT vẫn tiếp tục được áp dụng rộng rãi trong các bài toán NLP với nhiều phiên bản cải tiến như RoBERTa, ALBERT, DistilBERT, v.v. Nó đã trở thành công cụ ưa chuộng trên nhiều nền tảng cạnh tranh như Kaggle, AIVIVN và được trình bày tại nhiều hội nghị quan trọng.

PhoBERT, một biến thể huấn luyện trước dành riêng cho Tiếng Việt, đã đạt được kết quả ấn tượng trong nhiều bài toán xử lý ngôn ngữ. Tên gọi “Pho” được lấy từ món ăn phổ biến của Việt Nam, thể hiện sự gắn kết với ngôn ngữ và văn hóa địa phương.

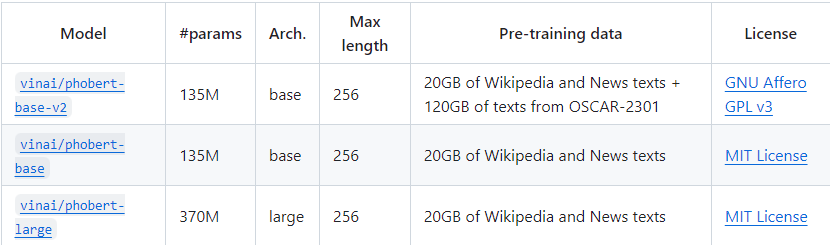
PhoBERT dễ dàng sử dụng, tích hợp được vào các thư viện như FAIRSeq của Facebook hay Transformers của Hugging Face, từ đó tăng cường sự phổ biến và ứng dụng của BERT cho cả Tiếng Việt lẫn Tiếng Anh.

**2.5.2 Cấu trúc của PhoBERT**

PhoBERT là một mô hình huấn luyện trước dành riêng cho tiếng Việt, được xây dựng trên nền tảng BERT và có hai phiên bản chính là PhoBERTbase với 12 khối transformers và PhoBERTlarge với 24 khối transformers, tương tự như BERT.

Mô hình PhoBERT đã được huấn luyện trên khoảng 20GB dữ liệu, bao gồm khoảng 1GB nguồn dữ liệu Wikipedia tiếng Việt và 19GB còn lại từ nguồn tin tức tiếng Việt. Đây là một lượng dữ liệu đáng kể, đủ để huấn luyện một mô hình như BERT.

PhoBERT sử dụng RDRSegmenter của VnCoreNLP để tách từ cho dữ liệu đầu vào trước khi áp dụng mã hóa BPE (Byte Pair Encoding). Mô hình này chỉ tập trung vào nhiệm vụ Mô hình ngôn ngữ đánh dấu trong quá trình huấn luyện và không bao gồm nhiệm vụ dự đoán câu tiếp theo. [26]



Hình 2.21: Pre-trained PhoBERT [26]

**2.5.3 Longformer PhoBERT**

Mô hình Longformer PhoBERT-base với độ dài đầu vào tối đa là 4096 token của tác giả **bluenguyen** [27]**, s**ử dụng phiên bản Transformers 4.25.1, mô hình được điều chỉnh từ phiên bản gốc **vinai/phobert-base (**với độ dài đầu vào tối đa là 256 token) để tích hợp các tính năng của Longformer. Quá trình này bao gồm chuyển đổi mô hình PhoBERT sử dụng mã nguồn của tác giả, sau đó tiến hành huấn luyện tiếp theo cho Marked Language Model (MLM) trong 5000 bước. Việc huấn luyện sử dụng kích thước batch là 64 trên tập dữ liệu ***[Binhvq News Corpus](https://github.com/binhvq/news-corpus)***, nhằm giúp mô hình học cách xử lý hiệu quả các tình huống văn bản dài với cơ chế sliding window attention mới.

# CHƯƠNG 3: THỰC NGHIỆM VÀ ĐÁNH GIÁ KẾT QUẢ

**3.1 MÔI TRƯỜNG VÀ DỮ LIỆU THỰC NGHIỆM**

**3.1.1 Môi trường cài đặt**

* Sử dụng Jupyter Notebook trên Google Colab với Python phiên bản 3.10.12
* Sử dụng linux server do khoa cấp với GPU Tesla T4, CPU Intel Xeon Silver 4216 (64) @ 3.200GHz để train mô hình.

A screen shot of a computer

Description automatically generated

Hình 3.1: Cấu hình máy dùng để train model

**3.1.2 Các thư viện python (đã sử dụng)**

1. *PySpark*

Thư viện dùng này sẽ có thể biết đến là Apache Spark kết hợp với ngôn ngữ lập trình Python, nó có hiệu quả rất tốt trong việc xử lý, phân tích khối lượng dữ liệu cực kỳ lớn. PySParkcung cấp giao diện cho Spark thông qua API của Python, để người dùng Python khai thác sức mạnh của Spark mà không cần phải thông qua các ngôn ngữ lập trình khác. [32]

* **Chức năng và ưu điểm chính:** PySpark sở hữu kho tàng tính năng giúp ta có thể chế ngự mọi thách thức về dữ liệu:[33]
* Sức mạnh phân tán: Phân tán dữ liệu và tính toán trên nhiều máy tính, mang đến tốc độ xử lý bứt phá.Khả năng mở rộng vô hạn, thích ứng mọi quy mô dữ liệu.
* Cấu trúc dữ liệu linh hoạt: RDD (Resilient Distributed Datasets) - lưu trữ và thao tác dữ liệu lớn một cách an toàn, hiệu quả. DataFrames - Cấu trúc bảng dễ sử dụng, tối ưu hóa tự động cho truy vấn nhanh chóng.
* Phân tích dữ liệu chuyên sâu: Spark SQLTruy vấn dữ liệu bằng ngôn ngữ SQL quen thuộc, tích hợp dễ dàng.MLlib (Machine Learning Library) - Xây dựng và triển khai mô hình học máy tiên tiến.GraphX - Phân tích đồ thị quy mô lớn, giải mã các mối liên hệ tiềm ẩn.
* Xử lý dữ liệu theo thời gian thực: Spark Streaming - Phân tích dữ liệu liên tục, cập nhật thông tin tức thì. Structured Streamin - Khả năng chịu lỗi và mở rộng cao cho dữ liệu streaming.
* Kết nối đa dạng: Hỗ trợ HDFS, Cassandra, HBase, Hive và nhiều hệ thống lưu trữ khác. Dễ dàng nhập và xuất dữ liệu, kết nối với các ứng dụng và công cụ khác.
* Tối ưu hóa hiệu suất: Catalyst Optimizer và Tungsten execution engine giúp tăng tốc độ xử lý dữ liệu.

1. *Tensorflow*

TensorFlow là một thư viện mã nguồn mở mạnh mẽ được phát triển bởi Google. Được thiết kế để hỗ trợ xây dựng các mô hình học sâu (deep learning) và học máy (machine learning). Nó cho phép xây dựng và huấn luyện các mô hình học máy từ đơn giản đến phức tạp trên nhiều loại thiết bị từ máy tính cá nhân đến cụm máy chủ và thiết bị di động. Nó có sử dụng biểu đồ tính toán (computational graph) để biểu diễn các phép toán dưới dạng nút và cạnh, giúp tối ưu hóa việc tính toán song song và phân tán.[34]

**Chức năng và ưu điểm chính:** [34]

* Xây dựng mô hình: TensorFlow cung cấp các công cụ để thiết kế và xây dựng các kiến trúc mô hình học máy phức tạp, từ các lớp cơ bản như fully-connected, convolutional, recurrent cho đến các mô hình tiên tiến như Transformers và Generative Adversarial Networks.
* Huấn luyện mô hình: Thư viện này hỗ trợ các thuật toán huấn luyện hiệu quả như gradient descent, Adam, RMSProp, cùng với các tính năng như checkpoint, early stopping để theo dõi và kiểm soát quá trình huấn luyện.
* Triển khai mô hình: TensorFlow cho phép triển khai các mô hình học máy lên nhiều nền tảng khác nhau, từ máy tính cá nhân đến cụm máy chủ lớn, cả thiết bị di động và edge devices. Nó tối ưu hóa hiệu suất tính toán thông qua sự hỗ trợ cho các thiết bị gia tốc phần cứng như GPU và TPU.
* Quản lý dữ liệu: Thư viện cung cấp các API linh hoạt để nạp, tiền xử lý và quản lý dữ liệu đầu vào cho mô hình học máy, hỗ trợ nhiều định dạng dữ liệu phổ biến.
* Kiểm soát và theo dõi: TensorFlow cung cấp các công cụ như TensorBoard để theo dõi, can thiệp và trực quan hóa quá trình huấn luyện và triển khai mô hình, bao gồm tracking metric, visualizing computational graphs, và phân tích hiệu suất.
* Hệ sinh thái và cộng đồng: Thư viện này có một cộng đồng lớn và sôi nổi, được tích hợp với nhiều công cụ, frameworks và thư viện khác để mở rộng khả năng ứng dụng.

1. *Pytorch*

Là thư viện có mã nguồn mở, được phát triển bởi đội ngũ facebookThư viện này là một trong số những thư viện hỗ trợ deeplearning như Keras hay tensoflow. Mục tiêu chính là thay thế kiến trúc của numpy để tính toán được trên GPU và Deep learning platform cung cấp các xử lý tốc độ và linh hoạt.[35]

**Chức năng và ưu điểm chính:** [35]

* Tensor Computing: PyTorch sử dụng tensors, một cấu trúc dữ liệu đa chiều tương tự như NumPy arrays, để biểu diễn và thao tác dữ liệu. Tensors hỗ trợ các phép toán đại số tuyến tính như cộng, nhân, phép chia, v.v. và có thể chạy trên GPU để tăng tốc độ tính toán.
* Automatic Differentiation: PyTorch có một hệ thống tự động đạo hàm (autograd) để tính các đạo hàm cần thiết cho các thuật toán tối ưu hóa, như gradient descent, trong quá trình huấn luyện mô hình machine learning.
* Neural Networks: PyTorch cung cấp một API linh hoạt và mạnh mẽ để xây dựng, huấn luyện và triển khai các mô hình neural network. Nó hỗ trợ nhiều kiến trúc phổ biến như convolutional networks, recurrent networks, và transformer networks.
* Distributed Training: PyTorch có hỗ trợ tích hợp sẵn cho đào tạo phân phối trên nhiều GPU và máy tính, giúp tăng tốc quá trình huấn luyện các mô hình lớn.
* Mobile và Embedded Deployment: Với PyTorch Mobile và TorchScript, có thể triển khai các mô hình PyTorch lên các thiết bị di động và nhúng, mở rộng khả năng ứng dụng của các mô hình machine learning.
* Eager Execution: Ngược lại với đồ thị tính toán tĩnh của TensorFlow, PyTorch sử dụng eager execution, cho phép thực hiện các tính toán theo từng dòng code Python, giúp code dễ hiểu và debug hơn.

1. *Underthesea*

Underthesea là một thư viện mã nguồn mở dành cho xử lý ngôn ngữ tự nhiên (NLP) cho tiếng Việt và ra mắt vào năm 2017, được phát triển bởi cộng đồng các nhà nghiên cứu và các kỹ sư về máy học tại Việt Nam. Với các mục tiêu như là: [36]

* Cung cấp các công cụ và tính năng NLP cơ bản, như phân tích từ loại, nhận dạng thực thể, phân tích cảm xúc, v.v. cho các ứng dụng tiếng Việt.
* Xây dựng cơ sở dữ liệu và mô hình NLP miễn phí, chất lượng cao, phục vụ cho cộng đồng nghiên cứu và phát triển ứng dụng NLP tiếng Việt.
* Thúc đẩy và hỗ trợ các nghiên cứu và ứng dụng NLP tiếng Việt trong cộng đồng.

**Chức năng và ưu điểm chính:** [36]

* Phân tích từ vựng: Underthesea cung cấp các công cụ để phân tích và trích xuất các thành phần ngữ nghĩa của văn bản tiếng Việt, như phân tích từ loại, nhận dạng thực thể, và phân tích cú pháp.
* Phân đoạn văn bản: Thư viện này có thể phân chia văn bản thành các đoạn, câu, và từ một cách chính xác, giúp chuẩn bị dữ liệu cho các tác vụ NLP khác.
* Gắn thẻ từ loại: Underthesea có thể gán các thẻ từ loại (như danh từ, động từ, tính từ, ...) cho từng từ trong văn bản, rất hữu ích cho các tác vụ như phân tích ngữ nghĩa.
* Nhận dạng thực thể: Thư viện này có khả năng nhận dạng các thực thể như tên người, tên địa danh, tổ chức, v.v. trong văn bản tiếng Việt.
* Phân tích cảm xúc: Underthesea cung cấp các công cụ để phân tích cảm xúc (tích cực, tiêu cực) trong văn bản tiếng Việt.
* Bộ dữ liệu và mô hình sẵn có: Underthesea đi kèm với một số bộ dữ liệu và mô hình NLP đã được huấn luyện sẵn, giúp người dùng nhachóng triển khai các ứng dụng NLP cho tiếng Việt.

1. *PhoNLP*

PhoNLP là một thư viện NLP (Xử lý ngôn ngữ tự nhiên) mạnh mẽ và linh hoạt, được phát triển đặc biệt để giải quyết các bài toán xử lý ngôn ngữ tự nhiên là tiếng Việt, và không chỉ hữu ích cho các ứng dụng cho tiếng Việt mà còn có thể áp dụng cho các ngôn ngữ khác. Nó được xây dựng trên nền tảng của Underthesea, mở rộng và bổ sung thêm nhiều tính năng tiên tiến. [37]

**Chức năng và ưu điểm chính:** PhoNLP tập trung vào các chức năng cốt lõi của xử lý ngôn ngữ tự nhiên tiếng Việt, đáp ứng nhu cầu của các ứng dụng thực tế [37]

* Phân tích từ loại (Part-of-Speech Tagging): PhoNLP có thể phân loại từng từ trong một văn bản thành các loại từ như danh từ, động từ, tính từ, ... với độ chính xác rất cao.
* Nhận dạng thực thể (Named Entity Recognition): Thư viện này có thể nhận dạng và trích xuất các thực thể quan trọng như tên người, tổ chức, địa điểm, ... từ văn bản tiếng Việt.
* Phân tích cú pháp (Syntactic Analysis): PhoNLP có thể phân tích cấu trúc ngữ pháp của câu, xác định các thành phần cú pháp như chủ ngữ, vị ngữ, ... rất hữu ích cho các ứng dụng xử lý ngôn ngữ.
* Phân đoạn văn bản (Text Segmentation): Tính năng này giúp chia một văn bản thành các đoạn nhỏ dựa trên các dấu câu, từ khóa, ... để thuận tiện cho các bước xử lý tiếp theo.
* Gắn nhãn chuỗi (Sequence Labeling): PhoNLP có thể gắn nhãn cho từng từ trong câu, rất hữu ích cho các tác vụ như nhận dạng thực thể, phân tích cảm xúc, ...

1. *Streamlit*

Streamlit là một thư viện mã nguồn mở của Python mạnh mẽ giúp người dùng có thể tạo ra các ứng dụng web tương tác và trực quan hóa dữ liệu và mường tượng hóa các bước thực hiện của các bài toán về AI một cách nhanh chóng và dễ dàng bằng Python. Không giống như các framework web truyền thống yêu cầu kiến thức sâu về HTML, CSS, và JavaScript, Streamlit cho phép ta xây dựng các giao diện người dùng bằng cách sử dụng hầu hết là code Python thuần túy, giúp tiết kiệm thời gian và công sức đáng kể. [38]

**Chức năng và ưu điểm chính:**  [38]

* Tạo giao diện người dùng (UI): Streamlit cung cấp một loạt các widgets như text input, buttons, checkboxes, sliders, selectboxes, file uploader, v.v. để tạo ra giao diện người dùng.Ta có thể sử dụng các hàm của Streamlit như st.text\_input(), st.button(), st.slider() để tạo ra các widget này trong ứng dụng web.
* Hiển thị dữ liệu: Streamlit hỗ trợ hiển thị nhiều loại dữ liệu như text, số, dataframe, plots, images, videos, v.v. Ta có thể sử dụng các hàm như st.write(), st.table(), st.line\_chart(), st.image() để hiển thị các loại dữ liệu này.
* Tương tác với người dùng: Streamlit cho phép người dùng tương tác với ứng dụng thông qua các widget như buttons, sliders, dropdowns, v.v. Giá trị của các widget này có thể được sử dụng để cập nhật hiển thị dữ liệu hoặc thực hiện các tính toán.
* Quản lý trạng thái: Streamlit cung cấp cơ chế quản lý trạng thái, cho phép lưu trữ và truy xuất dữ liệu giữa các lần chạy ứng dụng. Điều này cho phép taxây dựng các ứng dụng tương tác, với dữ liệu được lưu trữ và cập nhật liên tục.
* Triển khai ứng dụng: Streamlit ứng dụng có thể dễ dàng được triển khai trên các nền tảng như máy tính cá nhân, máy chủ, Heroku, AWS, Azure, v.v. Streamlit cung cấp các công cụ và hướng dẫn để giúp ta triển khai ứng dụng web một cách nhanh chóng và dễ dàng.
* Tích hợp với các thư viện khác: Streamlit có thể dễ dàng tích hợp với các thư viện Python khác như Pandas, Matplotlib, Scikit-learn, TensorFlow, v.v. Điều này cho phép ta xây dựng các ứng dụng web kết hợp với các tính năng và công cụ phân tích dữ liệu, học máy, v.v.

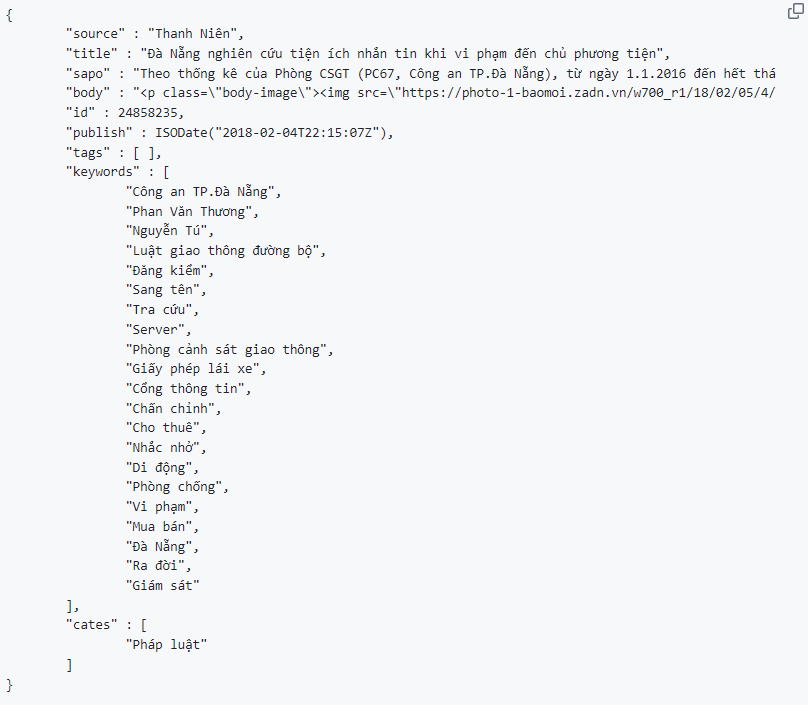
**3.1.3 Dữ liệu thực nghiệm**

Tập dữ liệu được sử dụng bao gồm các bài báo tin tức tiếng Việt được thu thập từ nhiều cổng thông tin tin tức trực tuyến khác nhau tại Việt Nam. Ban đầu, dữ liệu được lấy từ một bản sao lưu MongoDB chứa hơn 20 triệu bài báo. Từ tập dữ liệu lớn này, nhóm của chúng tôi đã rút trích khoảng 162,000 bài báo được phân loại vào 13 danh mục khác nhau.

Tập dữ liệu cung cấp một cái nhìn toàn diện về các sự kiện và chủ đề được các báo điện tử Việt Nam đưa tin, làm cho nó phù hợp để huấn luyện và đánh giá các mô hình xử lý ngôn ngữ tự nhiên dành cho nội dung truyền thông tiếng Việt.

Link Tập Dữ Liệu: [Vietnamese News Corpus on GitHub](https://github.com/binhvq/news-corpus" \t "_new)

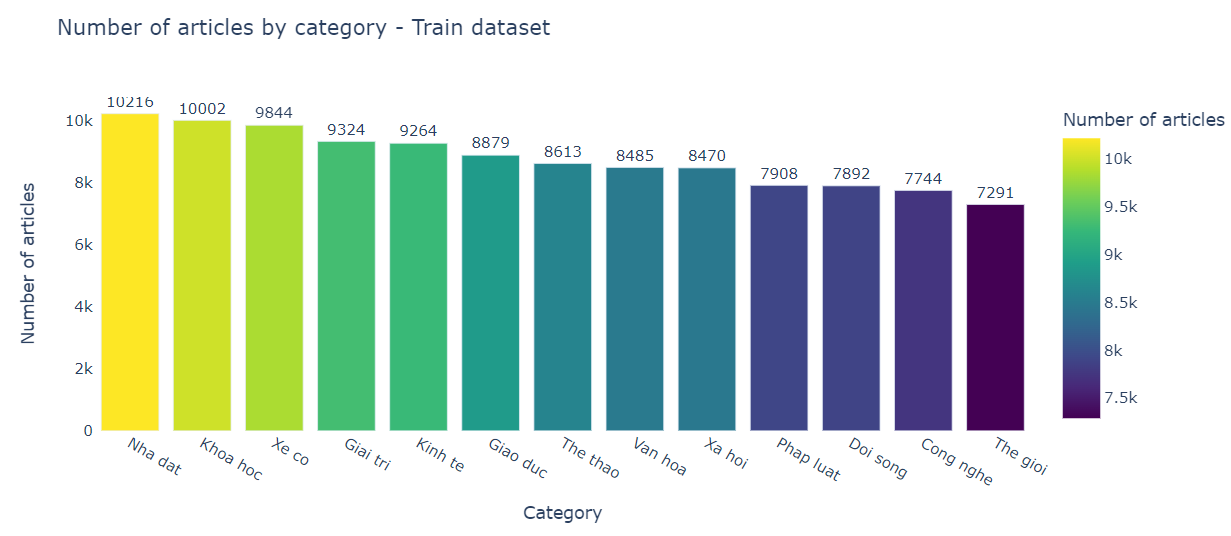
Dưới đây là một ví dụ về cấu trúc dữ liệu ban đầu:



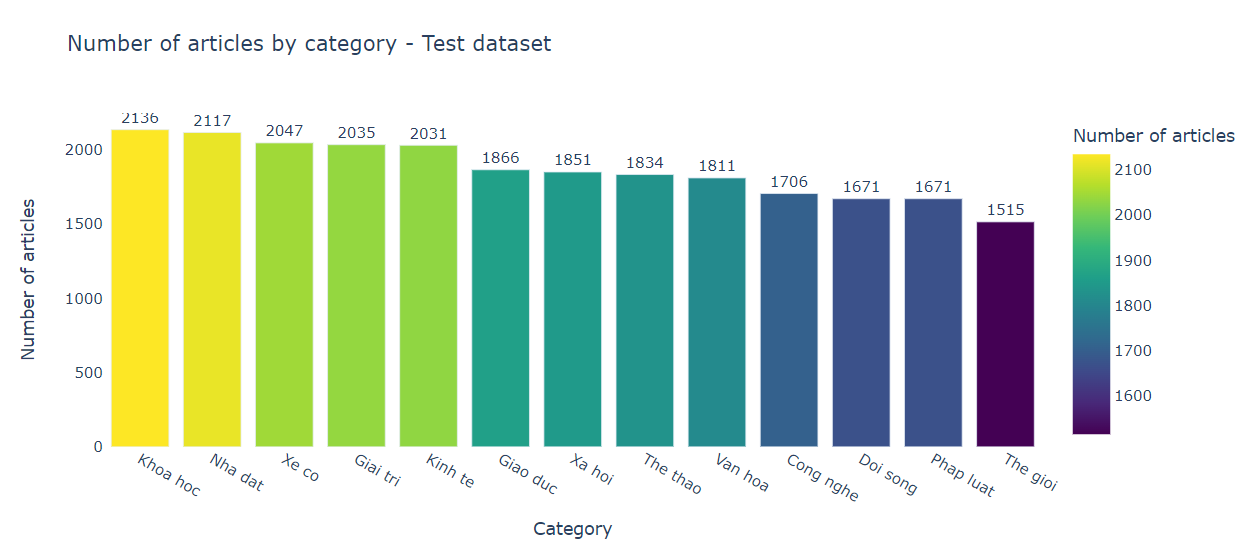
Hình 3.2: Cấu hình dữ liệu ban đầu

Để chuẩn bị tập dữ liệu cho các nhiệm vụ mô hình hóa, chúng đã thực hiện các bước xử lý sau:

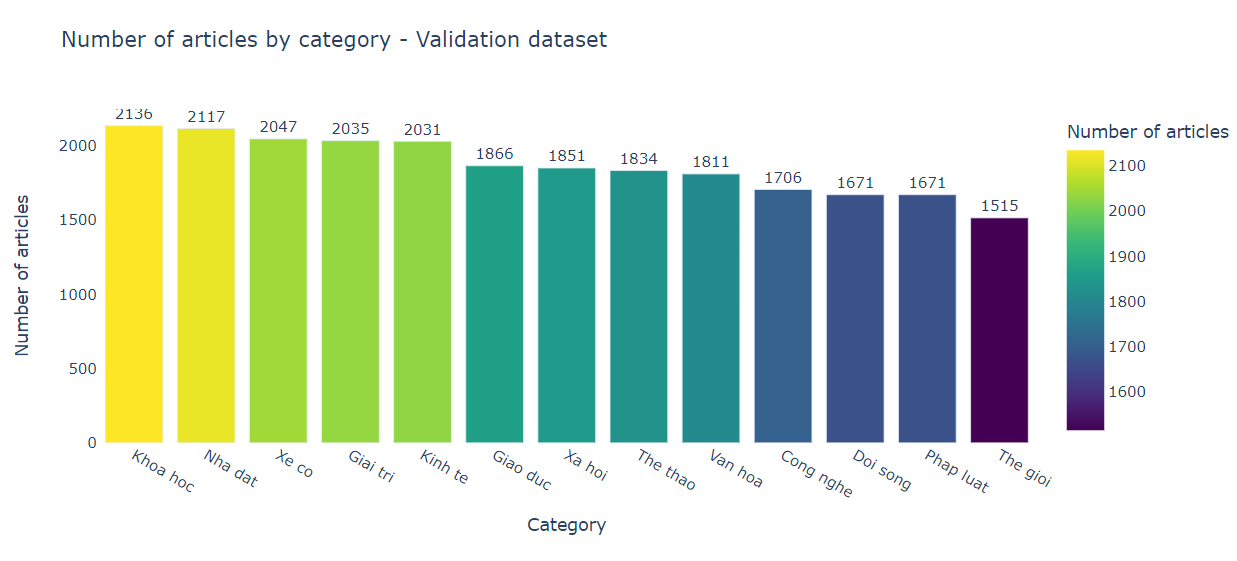
* Các bài báo được phân tích để trích xuất các thành phần chính như title (tiêu đề), sapo (tóm tắt), body (nội dung chính) và keywords (từ khoá).
* Mỗi bài báo được gán ít nhất một trong 13 danh mục đã được định nghĩa từ trước, giúp cho việc phân loại dễ dàng hơn.
* Tập dữ liệu được chia thành các tập huấn luyện (70%), kiểm tra (15%), và đánh giá (15%) để đảm bảo đánh giá mô hình hiệu quả trên nhiều tập dữ liệu khác nhau.



Hình 3.3: Tập dữ liệu train



Hình 3.4: Tập dữ liệu test



Hình 3.5: Tập dữ liệu validate

**3.2 TRÌNH TỰ THỰC HIỆN**

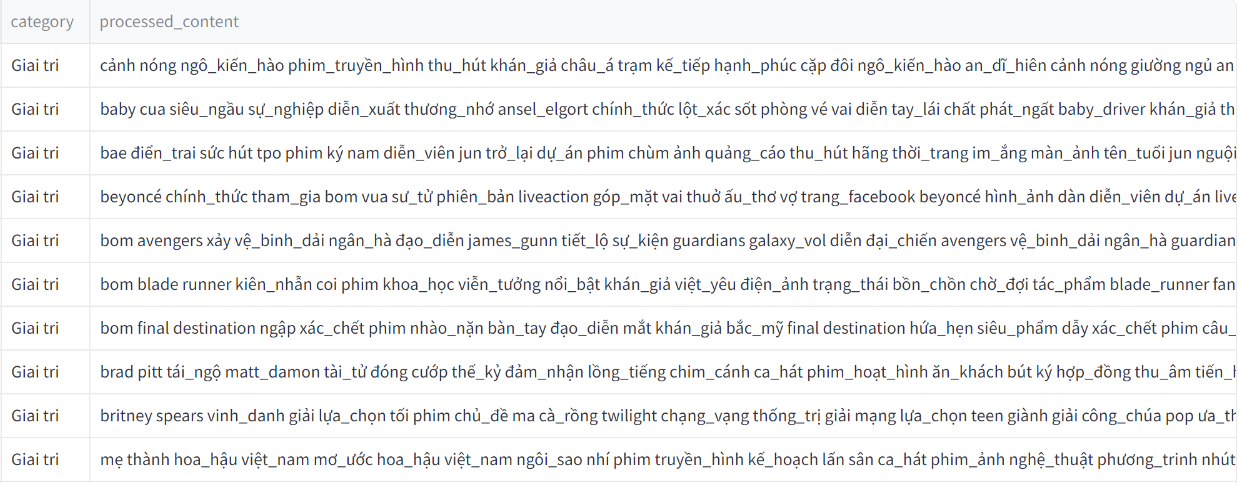
**3.2.1 Tiền xử lý dữ liệu**

Sau khi trích xuất, dữ liệu được chia thành hai phần chính là nội dung (content) và danh mục (category). Đầu tiên, chuẩn hóa tất cả các từ trong nội dung về dạng chữ thường để làm cho quá trình xử lý sau này nhất quán hơn và giảm thiểu sự khác biệt về dạng viết. Đồng thời, áp dụng các kỹ thuật biểu thức chính quy để loại bỏ các thành phần không cần thiết như thẻ HTML, đường dẫn URL, địa chỉ email, số liệu, vv. Quá trình này giúp làm sạch nội dung bài báo.

Tiếp theo, sử dụng thư viện Underthesea để phân đoạn từ trong câu. Sau đó, sử dụng thư viện PhoNLP để gắn nhãn từ loại cho mỗi từ trong câu, giúp xác định chính xác loại từ trong ngữ cảnh câu và loại bỏ các thành phần không quan trọng.

Lý do lựa chọn sử dụng PhoNLP cho nhiệm vụ gán nhãn từ loại thay vì Underthesea vì đây là một mô hình học đa nhiệm dựa trên BERT, được thiết kế để thực hiện ba nhiệm vụ chính là gán nhãn từ loại (POS tagging), nhận dạng thực thể (NER) và phân tích cú pháp phụ thuộc (dependency parsing). Các thử nghiệm trên các tập dữ liệu tiếng Việt đã cho thấy rằng PhoNLP mang lại kết quả vượt trội, đạt chuẩn hiện đại nhất so với các phương pháp học đơn nhiệm khác.

Dữ liệu sau khi tiền xử lý như sau:



Hình 3.6: Dữ liệu sau khi tiền xử lý

**3.2.2 Trích xuất đặc trưng PhoBERT**

Bước tiếp theo là trích xuất đặc trưng từ PhoBERT để chuẩn bị dữ liệu đầu vào cho mô hình BiLSTM. Để làm điều này, sử dụng PhoBERT như một công cụ trích xuất đặc trưng (feature extractor). Quá trình này bao gồm đưa các câu từ các bài báo qua PhoBERT để lấy các biểu diễn vector của từng từ. Các vector này sau đó được sử dụng như là đầu vào cho mô hình BiLSTM.

Các bước trích xuất đặc trưng như sau:

* Tokenization sử dụng BERT Tokenizer của PhoBERT: Trước khi đưa câu vào mô hình, câu được phân đoạn thành các token sử dụng BERT Tokenizer của PhoBERT và thêm hai token đặc biệt là [CLS] ở đầu câu và [SEP] ở cuối câu.
* Chèn câu đã tokenized vào mô hình với attention mask: Attention mask giúp mô hình tập trung vào các từ trong câu và bỏ qua các từ dư thừa do thêm padding. Các từ được thêm vào sẽ có giá trị attention mask là 0.
* Lấy vector đầu ra và sử dụng vector đầu tiên (ở vị trí token đặc biệt [CLS]): Sau khi đưa câu qua mô hình PhoBERT, ta lấy vector đầu ra và sử dụng vector đầu tiên (tại vị trí token đặc biệt [CLS]) như là đặc trưng cho câu đó để huấn luyện hoặc dự đoán.

Dưới đây là ví dụ cho câu “Hôm nay tôi đi học”

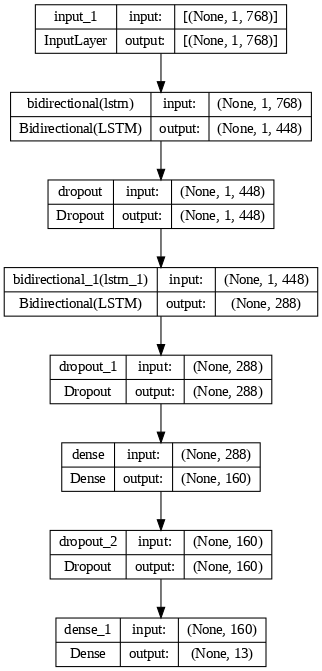


Hình 3.7: Ví dụ trích xuất đặc trưng

**3.2.3 Huấn luyện các mô hình**

*3.2.3.1 Mô hình Bi-LSTM*

Mô hình Bi-LSTM được xây dựng như sau:



Hình 3.8: Xây dựng mô hình Bi-LSTM

Sử dụng API Sequential của Keras để thiết kế mô hình. Mô hình bắt đầu bằng lớp input với kích thước dữ liệu huấn luyện đã xác định trước. Lớp đầu tiên là Bidirectional LSTM để xử lý chuỗi đầu vào theo cả hai hướng, kết hợp thông tin từ quá khứ và tương lai. Sau đó áp dụng Dropout để loại bỏ ngẫu nhiên một phần các đơn vị, ngăn chặn overfitting. Lớp Bidirectional LSTM thứ hai tiếp tục xử lý chuỗi kết quả từ lớp trước. Sau đó, thêm một lớp Dense với hàm kích hoạt ReLU để học các mẫu phức tạp trong dữ liệu. Cuối cùng, áp dụng lớp Dense cuối cùng với hàm kích hoạt softmax để dự đoán xác suất các lớp đầu ra. Mô hình sử dụng tối ưu hóa Adam và được biên soạn với hàm mất mát cross-entropy để đánh giá hiệu suất trong quá trình huấn luyện.

Sử dụng thư viện Optuna để tối ưu hoá siêu tham số cho mô hình. Optuna là một thư viện tối ưu hóa siêu tham số được sử dụng để cải thiện hiệu suất của mô hình thông qua các thuật toán tìm kiếm tiên tiến và tự động hóa quy trình tối ưu hóa. Việc lựa chọn Optuna vì tính hiệu quả và khả năng tự động hóa, giúp giảm thiểu công việc thủ công và tăng cường hiệu suất của mô hình.

Trong quá trình nghiên cứu, sử dụng TPESampler để thực hiện tìm kiếm siêu tham số hiệu quả và thích ứng. Đồng thời, áp dụng HyperbandPruner để sớm dừng các thử nghiệm không triển vọng, từ đó tiết kiệm tài nguyên và tăng tốc quá trình tối ưu hóa.

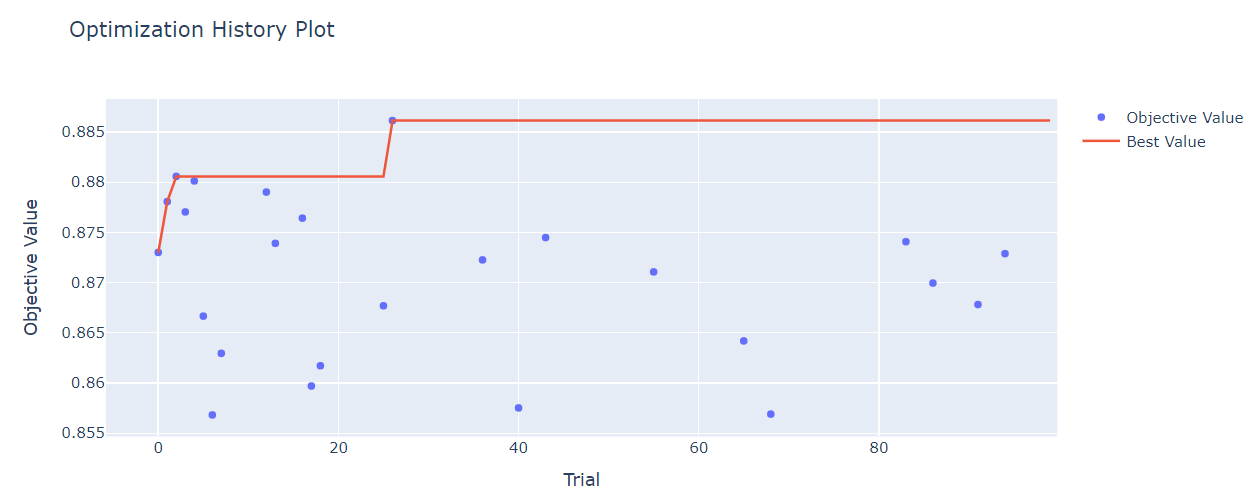
Trong Optuna, các thuật ngữ quan trọng bao gồm:

* Pruner Trials: Các thử nghiệm bị cắt giảm bởi Optuna để giảm thiểu việc tiêu tốn tài nguyên. Thao tác này loại bỏ những thử nghiệm không có khả năng cải thiện hoặc mất quá nhiều thời gian để đạt kết quả tốt.
* Complete Trials: Các thử nghiệm đã hoàn thành thành công và cung cấp kết quả hợp lệ. Chúng được sử dụng để đánh giá và lựa chọn siêu tham số tối ưu dựa trên mục tiêu tối ưu hóa.
* Failed Trials: Các thử nghiệm gặp lỗi hoặc không hoàn thành do vấn đề kỹ thuật hoặc cấu hình không phù hợp. Chúng không đóng góp kết quả hợp lệ cho quá trình tối ưu hóa.

Dưới đây là kết quả sau 100 lần thử nghiệm và siêu tham số tốt nhất của mô hình



Hình 3.9: Kết quả tối ưu siêu tham số mô hình Bi-LSTM

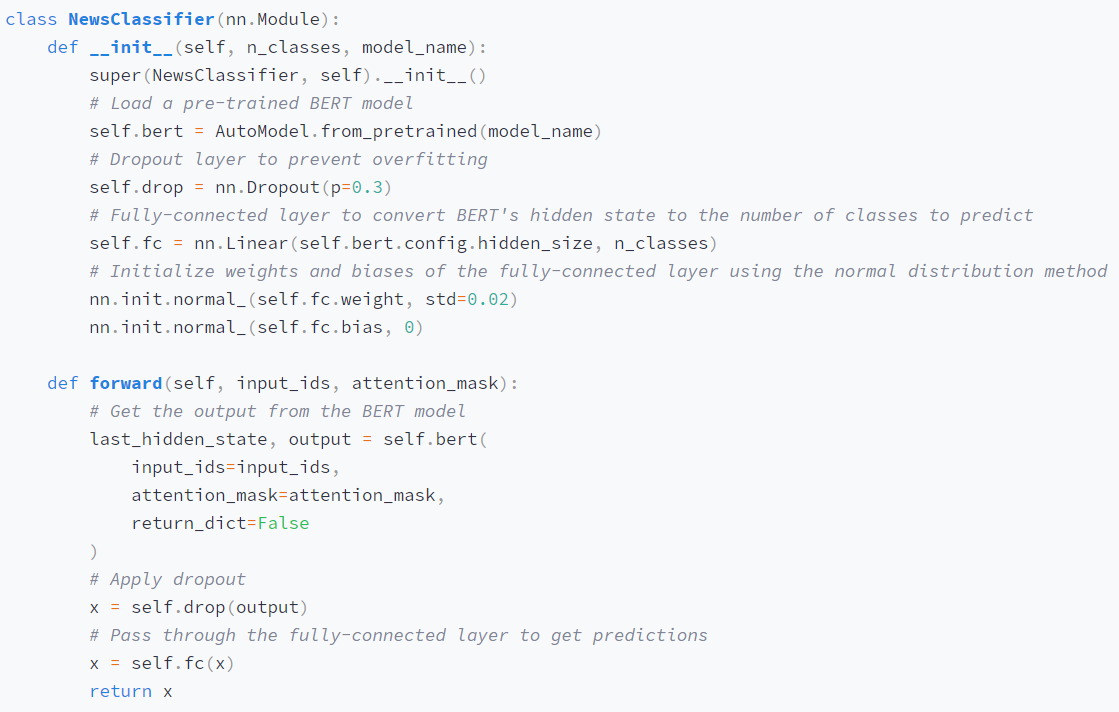


Hình 3.10: Biểu đồ quá trình tối ưu hoá siêu tham số mô hình Bi-LSTM

Bảng 3.1: Siêu tham số cho mô hình Bi-LSTM

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| lstm\_units\_1 | lstm\_units\_2 | dense\_units | dropout\_rate | learning\_rate | epochs | batch\_size |
| 224 | 144 | 160 | 0.4 | 0.000679344 | 30 | 192 |

*3.2.3.2 Fine-tune PhoBERT và Longformer PhoBERT*



Hình 3.11: Cài đặt fine-tune PhoBERT và Longformer PhoBERT

* **Khởi tạo Mô hình BERT**: Sử dụng AutoModel từ thư viện Transformers để tải mô hình BERT đã được huấn luyện trước đó (pre-trained model).
* **Lớp Dropout:** Thêm một lớp Dropout với xác suất loại bỏ đơn vị đầu vào là 0.3. Dropout giúp ngăn ngừa overfitting bằng cách ngẫu nhiên loại bỏ một số đơn vị trong quá trình huấn luyện, giúp mô hình tổng quát hóa tốt hơn.
* **Lớp Fully Connected (FC)**: Thêm một lớp Fully Connected để chuyển đổi các biểu diễn ngữ nghĩa từ BERT thành số lượng lớp cần thiết cho nhiệm vụ phân loại (số lượng lớp = n\_classes). Lớp này có vai trò quan trọng trong việc dự đoán nhãn của văn bản dựa trên các đặc trưng đã học được.
* **Khởi tạo Trọng số và Bias**: Khởi tạo trọng số của lớp Fully Connected theo phân phối chuẩn với độ lệch chuẩn là 0.02. Khởi tạo bias của lớp này bằng 0. Việc khởi tạo này giúp mô hình bắt đầu học từ các giá trị khởi tạo hợp lý.
* **Hàm forward:** Hàm forward định nghĩa quá trình truyền dữ liệu qua các lớp của mô hình để tạo ra dự đoán:
* Truyền dữ liệu qua BERT: Nhận đầu vào gồm input\_ids và attention\_mask, sau đó truyền qua mô hình BERT để lấy biểu diễn ngữ nghĩa của văn bản (last\_hidden\_state và output). Trong đó, output là biểu diễn của token [CLS], thường được sử dụng cho các tác vụ phân loại.
* Áp dụng Dropout: Áp dụng Dropout lên output của BERT để giảm overfitting.
* Truyền qua Lớp Fully Connected: Truyền dữ liệu qua lớp Fully Connected để chuyển đổi các đặc trưng ngữ nghĩa thành xác suất của các lớp. Đây là bước cuối cùng để tạo ra dự đoán nhãn của văn bản.
* Trả về Kết quả Dự đoán

Trong quá trình huấn luyện và đánh giá mô hình, sử dụng Stratified K-Fold Cross-Validation và kết hợp các fold. Stratified K-Fold Cross-Validation là một phương pháp quan trọng trong việc đánh giá và huấn luyện mô hình machine learning. Mục tiêu chính của phương pháp này là đảm bảo tính chính xác và khách quan trong quá trình đánh giá mô hình, đồng thời giúp mô hình tổng quát hóa tốt hơn khi áp dụng trên dữ liệu thực tế.

* **Cross-Validation**: Giúp đánh giá mô hình bằng cách chia dữ liệu thành các phần nhỏ (fold). Mô hình được huấn luyện trên các fold và đánh giá trên các fold còn lại. Điều này giúp giảm thiểu vấn đề overfitting và đánh giá khả năng tổng quát hóa của mô hình.
* **Stratified K-Fold**: Đảm bảo tỷ lệ mẫu của từng lớp (class) trong dữ liệu được giữ nguyên trong từng fold. Điều này quan trọng đặc biệt khi dữ liệu có sự mất cân bằng giữa các lớp, giúp đánh giá mô hình trở nên khách quan và chính xác hơn.

Khi sử dụng Stratified K-Fold Cross-Validation, thay vì chỉ chọn một mô hình từ một fold duy nhất, việc kết hợp thông tin từ các fold khác nhau để tạo ra một mô hình kết hợp (ensemble model) mang lại nhiều lợi ích. Mỗi fold chứa đựng những thông tin riêng biệt về tập dữ liệu, việc kết hợp chúng giúp mô hình có cái nhìn tổng quan và toàn diện hơn về dữ liệu. Ensemble model có xu hướng tổng quát hóa tốt hơn do được huấn luyện trên nhiều tập dữ liệu khác nhau, giúp giảm thiểu vấn đề overfitting. Việc kết hợp dự đoán từ nhiều mô hình con thường dẫn đến mô hình cuối cùng có độ chính xác và độ tin cậy cao hơn.

Các tham số lựa chọn cho fine-tune mô hình như sau:

{

    “k\_fold”: 5,

    “dropout\_probability”: 0.3,

    “hidden\_size”: 768,

    “learning\_rate”: 2e-05,

    “batch\_size”: 16,

    “num\_epochs”: 5,

    “gradient\_clipping”: 1.0,

    “type\_learning\_rate\_scheduler”: “linear\_schedule\_with\_warmup”,

    “num\_warmup\_steps”: 0,

    “loss\_function”: “CrossEntropyLoss”

}

**3.2.4 Đánh giá mô hình**

Các tiêu chí đánh giá mô hình như sau:

* Accuracy: Accuracy được tính bằng tỷ lệ số lượng dự đoán đúng (true positives và true negatives) trên tổng số mẫu.

Ví dụ: Giả sử mô hình phân loại 100 mẫu, trong đó dự đoán đúng 85 mẫu. Độ chính xác của mô hình là:

* Precision (Độ chính xác dương tính): Precision được tính bằng tỷ lệ số lượng true positives (mẫu dương tính thực sự được phân loại đúng) trên tổng số mẫu được phân loại là dương tính (positive predictions).

Ví dụ: Giả sử mô hình dự đoán có 60 mẫu là dương tính, trong đó 50 mẫu thực sự là dương tính. Precision của mô hình là:

* Recall (Độ phủ ): Recall được tính bằng tỷ lệ số lượng true positives trên tổng số mẫu thực sự là dương tính.

Ví dụ: Giả sử trong tổng số 100 mẫu dương tính, mô hình nhận diện được 70 mẫu. Recall của mô hình là:

* F1-score: F1-score là trung bình điều hòa của precision và recall, được tính theo công thức:

Với precision là 0.8333 và recall là 0.7, ta tính được F1-score là:

* Confusion Matrix (Ma trận nhầm lẫn):

Ví dụ: Một confusion matrix cho một bài toán phân loại nhị phân có thể nhìn như sau:

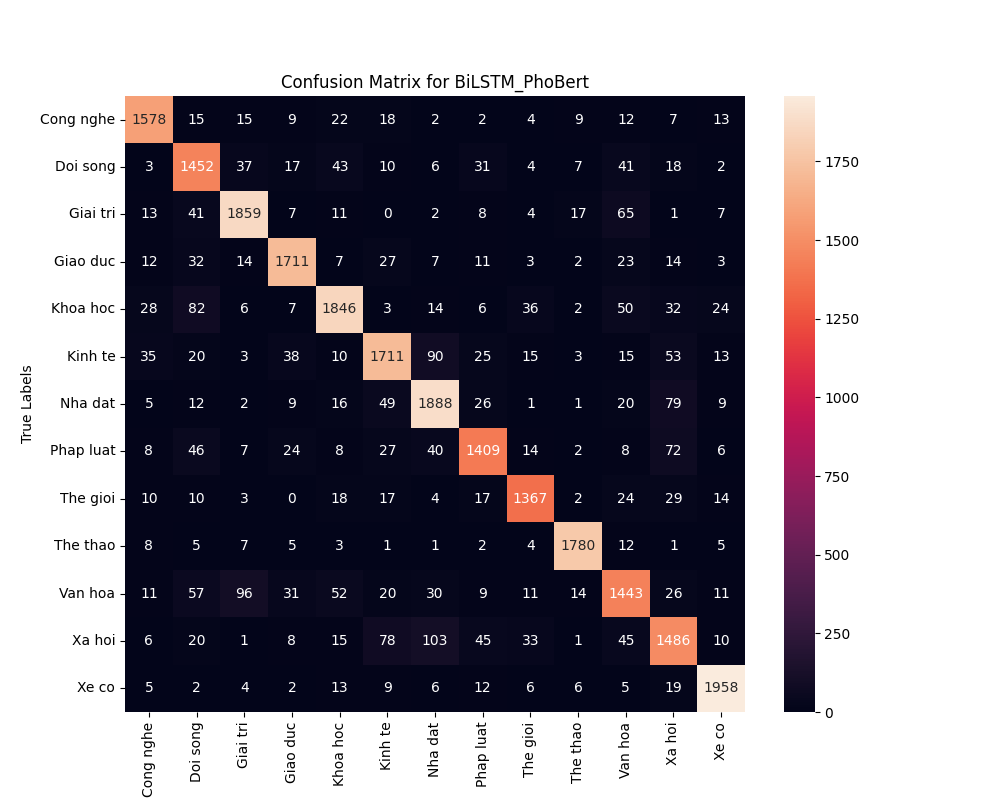
Bảng 3.2: Ví dụ về ma trận nhầm lẫn

|  |  |  |
| --- | --- | --- |
|  | Predicted Negative | Predicted Positive |
| Actual Negative | 50 | 10 |
| Actual Positive | 5 | 35 |

Trong ma trận này, có tổng cộng mẫu. True negatives là 50 (mẫu thực sự là negative và được dự đoán đúng là negative). False positives là 10 (mẫu thực sự là negative nhưng bị dự đoán là positive). False negatives là 5 (mẫu thực sự là positive nhưng bị dự đoán là negative). True positives là 35 (mẫu thực sự là positive và được dự đoán đúng là positive).

*3.2.4.1 Mô hình Bi-LSTM*

Dưới đây là kết quả đánh giá mô hình trên từng nhãn:



Hình 3.12: Confusion matrix mô hình Bi-LSTM

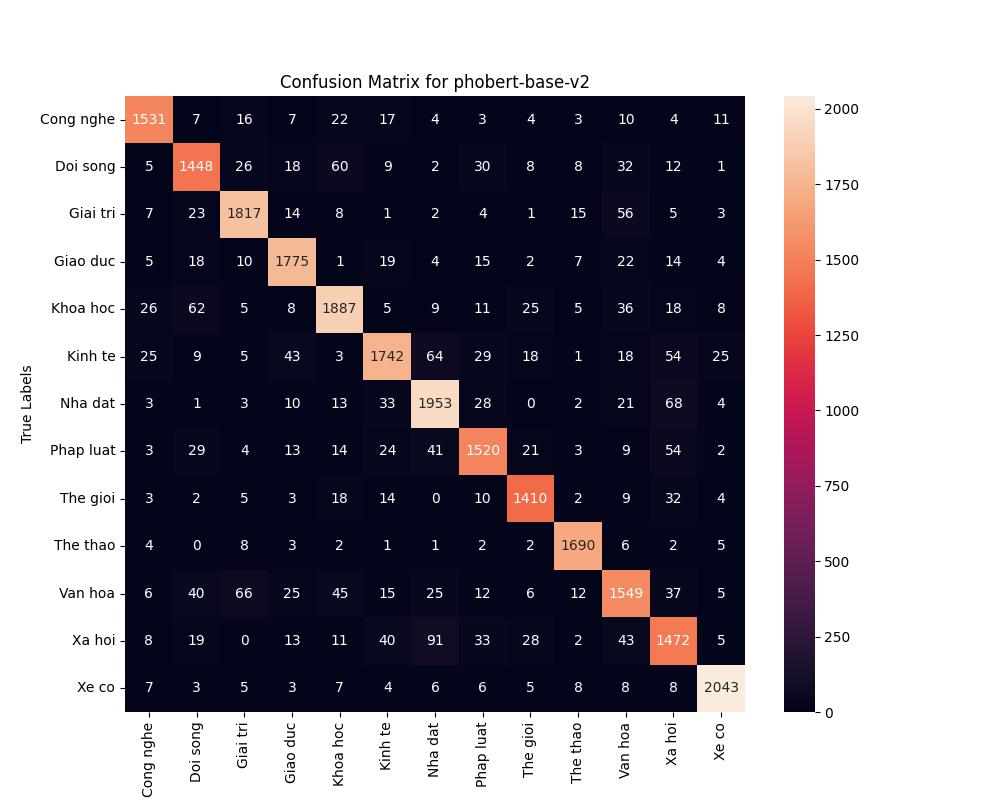
Bảng 3.3: Kết quả đánh giá trên từng nhãn mô hình Bi-LSTM

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1-score |
| Cong nghe | 0.91637631 | 0.92497069 | 0.92065344 |
| Doi song | 0.80936455 | 0.86894075 | 0.83809524 |
| Giai tri | 0.90506329 | 0.91351351 | 0.90926877 |
| Giao duc | 0.91595289 | 0.91693462 | 0.91644349 |
| Khoa hoc | 0.89437984 | 0.86423221 | 0.87904762 |
| Kinh te | 0.86852792 | 0.84244215 | 0.85528618 |
| Nha dat | 0.86092111 | 0.89182806 | 0.87610209 |
| Phap luat | 0.87897692 | 0.84320766 | 0.86072083 |
| The gioi | 0.91011984 | 0.90231023 | 0.90619821 |
| The thao | 0.96424702 | 0.97055616 | 0.9673913 |
| Van hoa | 0.81849121 | 0.79679735 | 0.8074986 |
| Xa hoi | 0.8089276 | 0.80280929 | 0.80585683 |
| Xe co | 0.94361446 | 0.95652174 | 0.95002426 |
| accuracy | 0.88460747 | 0.88460747 | 0.88460747 |
| macro avg | 0.88422792 | 0.88423573 | 0.88404514 |
| weighted avg | 0.88472792 | 0.88460747 | 0.88448596 |

Nhận xét:

* Hiệu suất chung của mô hình khá tốt, với độ chính xác (accuracy) lên tới 88.46%.
* Mô hình có hiệu suất rất tốt trên một số nhóm như “Công nghệ”, “Giải trí”, “Thể thao” và “Xe cộ”, với độ chính xác (precision), độ nhạy (recall) và F1-score cao, trên 90%.
* Tuy nhiên, vẫn có một số nhóm có hiệu suất không ấn tượng bằng, như “Đời sống”, “Văn hóa” và “Xã hội”, với F1-score dưới 85%.
* Phân tích ma trận nhầm lẫn cho thấy có một số nhầm lẫn đáng kể giữa các nhóm gần nhau như “Khoa học” và “Giáo dục”, “Kinh tế” và “Xã hội”. Điều này cho thấy mô hình vẫn gặp khó khăn trong việc phân biệt những nhóm dữ liệu tương tự.

*3.2.4.2 PhoBERT*



Hình 3.13: Confusion matrix PhoBERT

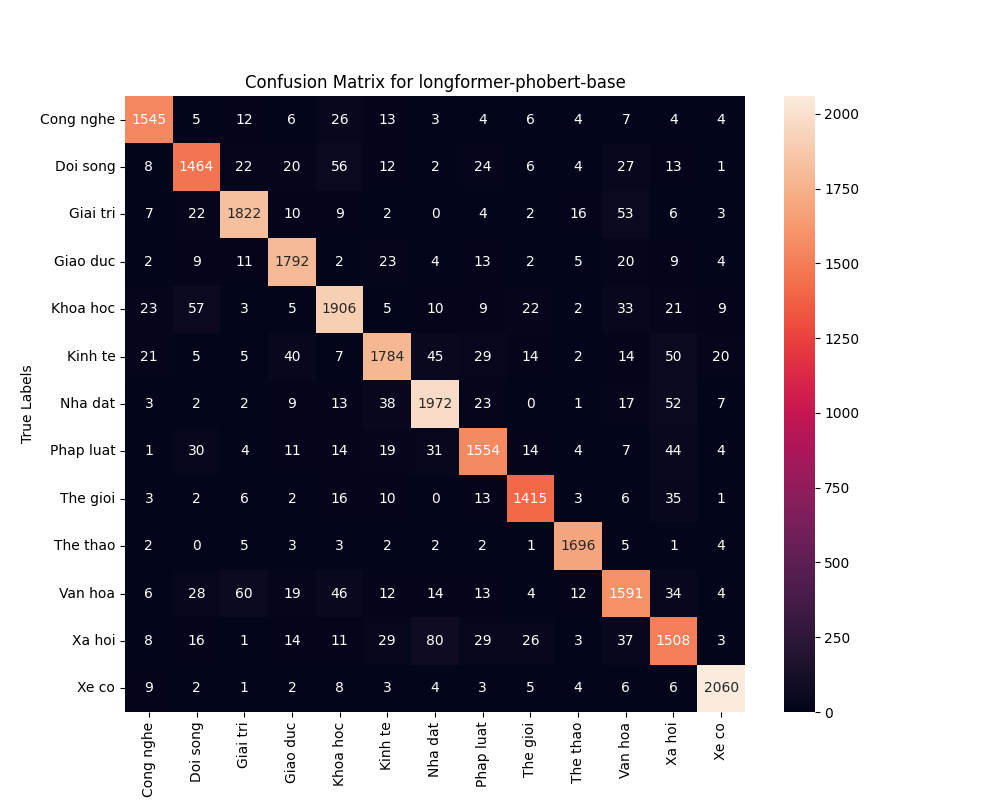
Bảng 3.4: Kết quả đánh giá trên từng nhãn PhoBERT

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1-score |
| Cong nghe | 0.93753827 | 0.93410616 | 0.93581907 |
| Doi song | 0.871764 | 0.87281495 | 0.87228916 |
| Giai tri | 0.92233503 | 0.92893661 | 0.92562404 |
| Giao duc | 0.91731266 | 0.93618143 | 0.926651 |
| Khoa hoc | 0.90243902 | 0.89643705 | 0.89942803 |
| Kinh te | 0.90540541 | 0.85559921 | 0.87979798 |
| Nha dat | 0.88692098 | 0.91304348 | 0.89979267 |
| Phap luat | 0.89254257 | 0.87507196 | 0.88372093 |
| The gioi | 0.92156863 | 0.93253968 | 0.9270217 |
| The thao | 0.96131968 | 0.97914253 | 0.97014925 |
| Van hoa | 0.85156679 | 0.84047748 | 0.8459858 |
| Xa hoi | 0.82696629 | 0.83399433 | 0.83046544 |
| Xe co | 0.96367925 | 0.96687175 | 0.96527286 |
| accuracy | 0.9051231 | 0.9051231 | 0.9051231 |
| macro avg | 0.90471989 | 0.90501666 | 0.90477061 |
| weighted avg | 0.90502531 | 0.9051231 | 0.90497015 |

Dựa vào ma trận nhầm lẫn và kết quả đánh giá trên từng nhãn, có thể rút ra các nhận xét sau:

* Mô hình đạt được độ chính xác (accuracy) cao, lên tới 90.51%, cho thấy hiệu suất chung của mô hình khá tốt.
* Có một số nhóm dữ liệu mà mô hình thể hiện hiệu suất rất cao, như “Công nghệ”, “Giải trí”, “Giáo dục”, “Thể thao” và “Xe cộ”, với độ chính xác (precision), độ nhạy (recall) và F1-score đều trên 90%.
* Tuy nhiên, vẫn có một số nhóm dữ liệu mà mô hình còn gặp một số khó khăn, như “Đời sống”, “Kinh tế”, “Văn hóa” và “Xã hội”, với F1-score dưới 88%.
* Phân tích ma trận nhầm lẫn cho thấy vẫn còn một số nhầm lẫn đáng kể giữa các nhóm dữ liệu tương tự

*3.2.4.3 Longformer PhoBERT*



Hình 3.14: Confusion matrix Longformer PhoBERT

Bảng 3.5: Kết quả đánh giá trên từng nhãn Longformer PhoBERT

|  |  |  |  |
| --- | --- | --- | --- |
|  | Precision | Recall | F1-score |
| Cong nghe | 0.94322344 | 0.94264796 | 0.94293561 |
| Doi song | 0.89159562 | 0.88245931 | 0.88700394 |
| Giai tri | 0.93244626 | 0.93149284 | 0.93196931 |
| Giao duc | 0.92705639 | 0.94514768 | 0.93601463 |
| Khoa hoc | 0.90033066 | 0.90546318 | 0.90288963 |
| Kinh te | 0.91393443 | 0.8762279 | 0.89468405 |
| Nha dat | 0.91001384 | 0.92192613 | 0.91593126 |
| Phap luat | 0.90348837 | 0.89464594 | 0.89904542 |
| The gioi | 0.93276203 | 0.93584656 | 0.93430175 |
| The thao | 0.96583144 | 0.98261877 | 0.97415279 |
| Van hoa | 0.87273725 | 0.86326641 | 0.867976 |
| Xa hoi | 0.84576556 | 0.85439093 | 0.85005637 |
| Xe co | 0.96986817 | 0.97491718 | 0.97238612 |
| accuracy | 0.91639725 | 0.91639725 | 0.91639725 |
| macro avg | 0.91608104 | 0.91623468 | 0.9161036 |
| weighted avg | 0.91629086 | 0.91639725 | 0.91628691 |

Nhận xét:

* Mô hình Long former-PhoBERT đạt được độ chính xác (accuracy) khá cao, lên tới 91.64%, cho thấy hiệu suất chung của mô hình tương đối tốt.
* Có một số lĩnh vực mà mô hình thể hiện hiệu suất rất cao, như "Công nghệ", "Giải trí", "Giáo dục", "Thể thao" và "Xe cộ", với độ chính xác (precision), độ nhạy (recall) và F1-score đều trên 93%.
* Tuy nhiên, vẫn còn một số lĩnh vực mà mô hình có độ chính xác thấp hơn, như "Đời sống", "Kinh tế", "Văn hóa" và "Xã hội", với F1-score dưới 88%.

**3.3 SO SÁNH CÁC MÔ HÌNH**

Dưới đây là kết quả huấn luyện tổng thể của ba mô hình

Bảng 3.6: Kết quả đánh giá tổng thể các mô hình

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Precision | Recall | F1-Score | Accuracy | Training Time |
| BiLSTM+PhoBERT | 88.47% | 88.46% | 88.44% | 88.46% | 192.62 seconds + 8057.01 seconds (feature extraction) |
| Longformer-PhoBERT | 91.63% | 91.64% | 91.63% | 91.64% | 275155.18 seconds |
| PhoBERT-base | 90.50% | 90.51% | 90.50% | 90.51% | 130717.66 seconds |

Dựa trên những kết quả đánh giá từ ba mô hình Bi-LSTM kết hợp với trích xuất đặc trưng từ PhoBERT, PhoBERT fine-tuning và Longformer PhoBERT, có thể đưa ra nhận xét và kết luận như sau:

* **Hiệu suất tổng thể:**
* *Longformer PhoBERT* đạt được hiệu suất tốt nhất với precision, recall và F1-score trên mỗi nhãn đều cao hơn so với hai mô hình còn lại. Đây là một mô hình được thiết kế để xử lý các văn bản có độ dài lớn hơn so với các transformer thông thường. Do đó, nó bảo toàn được thông tin context rộng hơn trong các văn bản dài, từ đó cải thiện khả năng phân loại trên các lớp dữ liệu khác nhau. Ngoài ra, nó có khả năng dự đoán chính xác hơn và khả năng tái tạo lại các trường hợp tích cực và thực tế tốt hơn.
* *PhoBERT base* mặc dù không đạt được hiệu suất cao nhất như Longformer PhoBERT, vẫn cho kết quả rất tốt và vượt qua Bi-LSTM ở hầu hết các nhãn.
* *Bi-LSTM* dù có tính linh hoạt trong việc xử lý chuỗi, nhưng hạn chế trong việc học các đặc trưng phức tạp và tổng quát hóa trên dữ liệu ngôn ngữ phức tạp như các bài báo nên hiệu suất không cao bằng hai mô hình trên.
* **Thời gian huấn luyện:**
* *Bi-LSTM*yêu cầu thời gian huấn luyện ngắn nhất, có thể phần nào do sự đơn giản của kiến trúc LSTM so với transformer và longformer.
* *PhoBERT base* và *Longformer PhoBERT* đều có thời gian huấn luyện lớn hơn rất nhiều so với Bi-LSTM, đặc biệt là Longformer PhoBERT có thời gian luyện lớn hơn đáng kể do độ dài bối cảnh lớn hơn và tính phức tạp của mô hình cũng như hạn chế về tài nguyên để training.
* **Đánh giá tổng quát**
* Việc sử dụng các mô hình transformer-based như PhoBERT và Longformer PhoBERT cho thấy khả năng mở rộng và cải thiện đáng kể so với mô hình truyền thống như BiLSTM trong phân loại văn bản tiếng Việt.
* Longformer PhoBERT đặc biệt có thể xử lý các văn bản dài hơn và bảo toàn thông tin context rộng hơn, dẫn đến hiệu suất cao hơn trên mọi nhãn.
* Kết quả này khuyến khích việc phát triển và cải tiến các mô hình Transformer dựa trên các kiến trúc sâu hơn như Longformer, nhằm cải thiện khả năng xử lý và hiệu suất phân loại trên các dữ liệu ngôn ngữ phức tạp và đa dạng.

**KẾT LUẬN**

**1. Những kết quả đạt được**

* Nghiên cứu và áp dụng thành công các mô hình tiên tiến như PhoBERT và Longformer PhoBERT để phân loại và xử lý văn bản tiếng Việt.
* Mô hình Longformer PhoBERT đạt được hiệu suất cao nhất với Precision, Recall, và F1-score cao trên từng nhãn chủ đề, cho thấy khả năng tổng quát hóa và ứng dụng rộng rãi trong thực tế.
* Thiết kế giao diện bằng Streamlit rõ ràng, dễ hiểu, trực quan hóa các bước thực hiện bài toán và dự đoán danh mục cho các bài báo.
* Kết quả nghiên cứu mở ra nhiều cơ hội ứng dụng trong truyền thông, giáo dục và kinh doanh, nâng cao khả năng tự động hóa và hiệu quả trong quản lý thông tin văn bản tiếng Việt.

**2. Hạn chế của đề tài**

* Thời gian huấn luyện dài với mô hình Longformer PhoBERT do yêu cầu xử lý dữ liệu lớn và phức tạp, tiêu tốn nhiều tài nguyên tính toán và có thể gây chi phí cao.
* Giới hạn chỉ tập trung vào phân loại với 13 nhãn khác nhau, giới hạn khả năng tổng quát hóa của mô hình khi áp dụng vào các bài toán phân loại với số lượng nhãn lớn hơn.
* Dữ liệu không đủ đa dạng và lượng lớn có thể làm giảm khả năng tổng quát hóa của mô hình, mặc dù đã sử dụng nhiều nguồn từ các bài báo và tin tức tiếng Việt.

**3. Phương hướng phát triển**

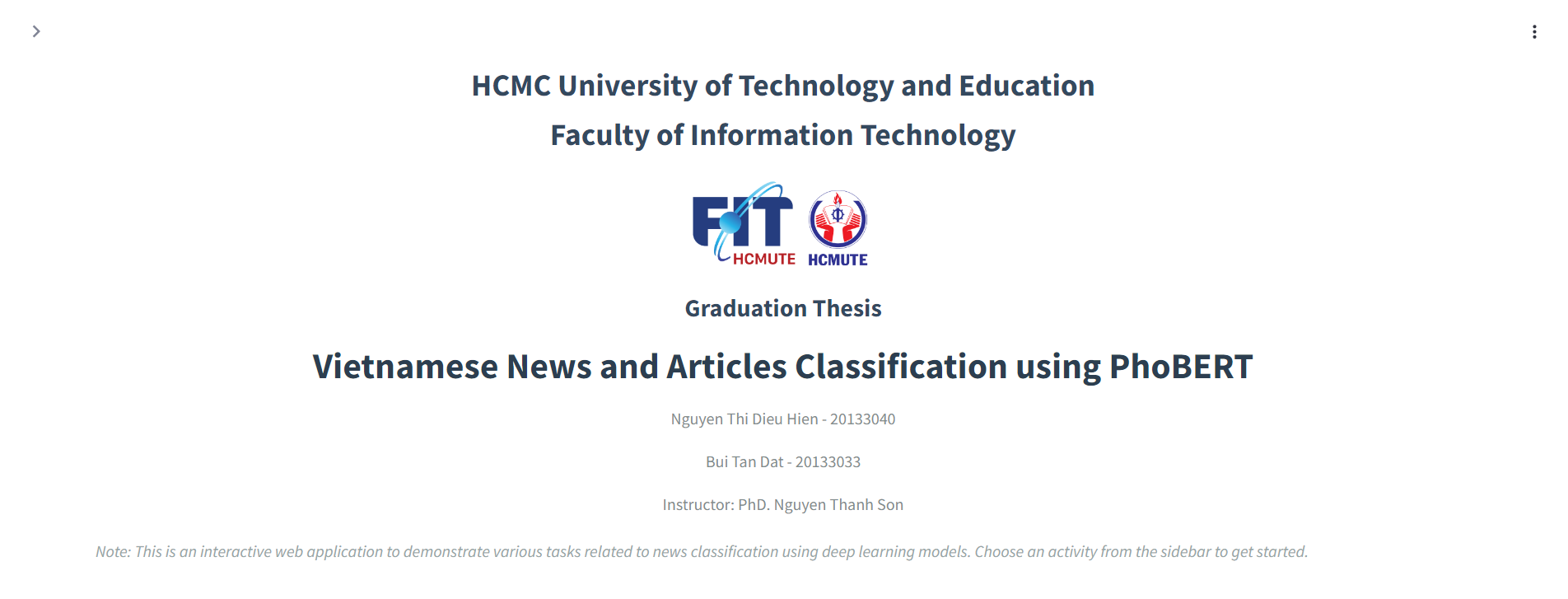
* Nghiên cứu và thu thập thêm dữ liệu phong phú hơn từ nhiều nguồn khác nhau, bao gồm cả các nhãn phân loại mới.
* Áp dụng các biến thể mới nhất của BERT và transformer-based models để cải thiện hiệu suất và tính ứng dụng của hệ thống phân loại
* Mở rộng áp dụng của các mô hình này vào các lĩnh vực khác như phân loại văn bản pháp luật, tóm tắt và tổng hợp nội dung, phân loại sản phẩm và dịch vụ, …

# TÀI LIỆU THAM KHẢO

|  |  |
| --- | --- |
| [1] | "What is Text Classification? | A Comprehensive Text Classification Guide," [Online]. Tại: https://www.elastic.co/what-is/text-classification. |
| [2] | "Text Classification," [Online]. Tại: https://huggingface.co/tasks/text-classification. |
| [3] | "Best Machine Learning Algorithm For Text Classification," [Online]. Tại: https://images.app.goo.gl/JGyVfiRCeVu15hfn8. |
| [4] | F. Ayık, "Mastering Text Classification with BERT: A Comprehensive Guide," 17 Dec 2023. [Online]. Tại: https://medium.com/@ayikfurkan1/mastering-text-classification-with-bert-a-comprehensive-guide-194ddb2aa2e5. |
| [5] | D. Q. N. a. A. T. Nguyen, "PhoBERT: Pre-trained language models for Vietnamese," p. 1, 2020. |
| [6] | N. P. V. a. H. H. Kha, "Vietnamese News Articles Classification Using," *Journal of Advances in Information Technolog,* vol. 12, p. 1, 2021. |
| [7] | A. malak, "RNN & LSTM & BI-LSTM & GRU," 28 Feb 2023. [Online]. Tại: https://medium.com/@antoniosmalak14/rnn-lstm-bi-lstm-gru-eb5869660691. |
| [8] | N. T. Huyen, "Recurrent Neural Network: Từ RNN đến LSTM," 24 June 2021. [Online]. Tại: https://viblo.asia/p/recurrent-neural-network-tu-rnn-den-lstm-gGJ597z1ZX2. |
| [9] | D. C. Ihenacho, "Recurrent Neural Network, LSTM and Bidirectional LSTM," 5 Jan 2023. [Online]. Tại: https://medium.com/@danihenacho95/recurrent-neural-network-lstm-and-bidirectional-lstm-110b72a3df9b. |
| [10] | "How do I draw 90 degree angle arrows and background," 6 May 2024. [Online]. Tại: https://tex.stackexchange.com/questions/718953/how-do-i-draw-90-degree-angle-arrows-and-background. [Accessed 26 July 2024]. |
| [11] | A. T. NLP, "WHAT IS THE MAIN DIFFERENCE BETWEEN RNN AND LSTM | NLP | RNN VS LSTM," 2 July 2021. [Online]. Tại: https://ashutoshtripathi.com/2021/07/02/what-is-the-main-difference-between-rnn-and-lstm-nlp-rnn-vs-lstm/. |
| [12] | Anishnama, "Understanding Bidirectional LSTM for Sequential Data Processing," 18 May 2023. [Online]. Tại: https://medium.com/@anishnama20/understanding-bidirectional-lstm-for-sequential-data-processing-b83d6283befc. |
| [13] | Y. Zhao, "Complete Guide to RNN, LSTM, and Bidirectional LSTM," 12 March 2023. [Online]. Tại: https://dagshub.com/blog/rnn-lstm-bidirectional-lstm/. |
| [14] | S. Nath, "Why is BiLSTM better than LSTM ?," 24 Apr 2023. [Online]. Tại: https://medium.com/@souro400.nath/why-is-bilstm-better-than-lstm-a7eb0090c1e4. |
| [15] | N. V. Anh, "Transformers - "Người máy biến hình" biến đổi thế giới NLP," 5 Jan 2020. [Online]. Tại: https://viblo.asia/p/transformers-nguoi-may-bien-hinh-bien-doi-the-gioi-nlp-924lJPOXKPM. |
| [16] | S. Cristina, "The Transformer Model," 6 Jan 2023. [Online]. Tại: https://machinelearningmastery.com/the-transformer-model/. |
| [17] | H. Gautam, "Word Embedding: Basics," 1 March 2020. [Online]. Tại: https://medium.com/@hari4om/word-embedding-d816f643140. |
| [18] | G. Giacaglia, "How Transformers Work," 11 March 2019. [Online]. Tại: https://towardsdatascience.com/transformers-141e32e69591. |
| [19] | J. Alammar, "The Illustrated Transformer," [Online]. Tại: https://jalammar.github.io/illustrated-transformer/. |
| [20] | rahulsm27, "Longformer in Deep Learning," 30 Nov 2023. [Online]. Tại: https://www.geeksforgeeks.org/longformer/. |
| [21] | R. Horev, "BERT Explained: State of the art language model for NLP," 11 Nov 2018. [Online]. Tại: https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270. |
| [22] | baeldung, "What Are Downstream Tasks?," 18 March 2024. [Online]. Tại: https://www.baeldung.com/cs/downstream-tasks. |
| [23] | N. T. Long, "TÌM HIỂU VỀ MÔ HÌNH NGÔN NGỮ PHOBERT," Hai Phong, 2021. |
| [24] | pawangfg, "Explanation of BERT Model – NLP," 10 Jan 2024. [Online]. Tại: https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/. |
| [25] | F. Ayık, "Mastering Text Classification with BERT: A Comprehensive Guide," 17 Dec 2023. [Online]. Tại: https://medium.com/@ayikfurkan1/mastering-text-classification-with-bert-a-comprehensive-guide-194ddb2aa2e5. |
| [26] | [Online]. Tại: https://github.com/VinAIResearch/PhoBERT. |
| [27] | bluenguyen. [Online]. Tại: https://huggingface.co/bluenguyen/longformer-phobert-base-4096. |
| [28] | G. Lokare, "Masked Language Modeling with Hugging Face Transformers: A Beginner’s Guide," 6 Feb 2023. [Online]. Tại: https://medium.com/@lokaregns/masked-language-modeling-with-hugging-face-transformers-a-beginners-guide-b36af74a7560. |
| [29] | "Recurrent Neural Network," [Online]. Tại: https://developer.nvidia.com/discover/recurrent-neural-network. |
| [30] | Howard, J. and Ruder, S. (2018) Universal language model fine-tuning for text classification, arXiv.org. Tại: https://arxiv.org/abs/1801.06146 |
| [31] | Zhang, Xulong et al. Improving imbalanced text classification with Dynamic Curriculum Learning. Tại: https://arxiv.org/pdf/2210.14724 |
| [32] | PySpark overview¶ PySpark Overview - PySpark master documentation. Tại: https://spark.apache.org/docs/latest/api/python/index.html |
| [33] | Harsoor, S. (2023) PySpark: Everything you need to know!, Medium. Tại: https://pub.aimind.so/pyspark-everything-you-need-to-know-24f87d12bfe1 . |
| [34] | Tensorflow là gì? Nguyên lý Hoạt động Của Tensorflow (no date) Stringee. Tại: https://stringee.com/vi/blog/post/Tensorflow-la-gi. |
| [35] | Pytorch là gì? Pytorch khác tensorflow như thế nào? (no date) gcs.vn. Tại: Https://gcs.vn/pytorch-la-gi/. |
| [36] | Under the Sea Documentation. Tại: https://underthesea.readthedocs.io/\_/downloads/en/develop/pdf/ |
| [37] | PHONLP: A joint multi-task learning model for Vietnamese part ... Tại: https://aclanthology.org/2021.naacl-demos.1.pdf |
| [38] | Huy, T.V. (2021) Introduction to Streamlit, Introduction to Streamlit – Tran Van Huy – Artificial Intellegence. Tại: https://huytranvan2010.github.io/Introduction-to-streamlit |

# PHỤ LỤC

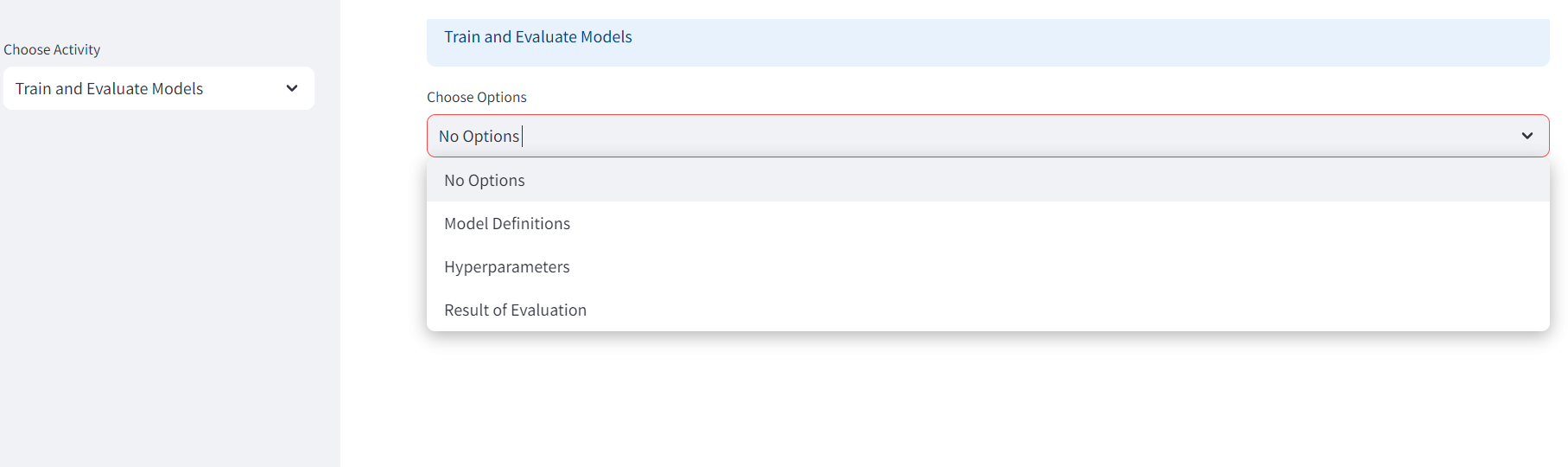
**HƯỚNG DẪN SỬ DỤNG PHẦN MỀM PHÂN LOẠI TIN TỨC**



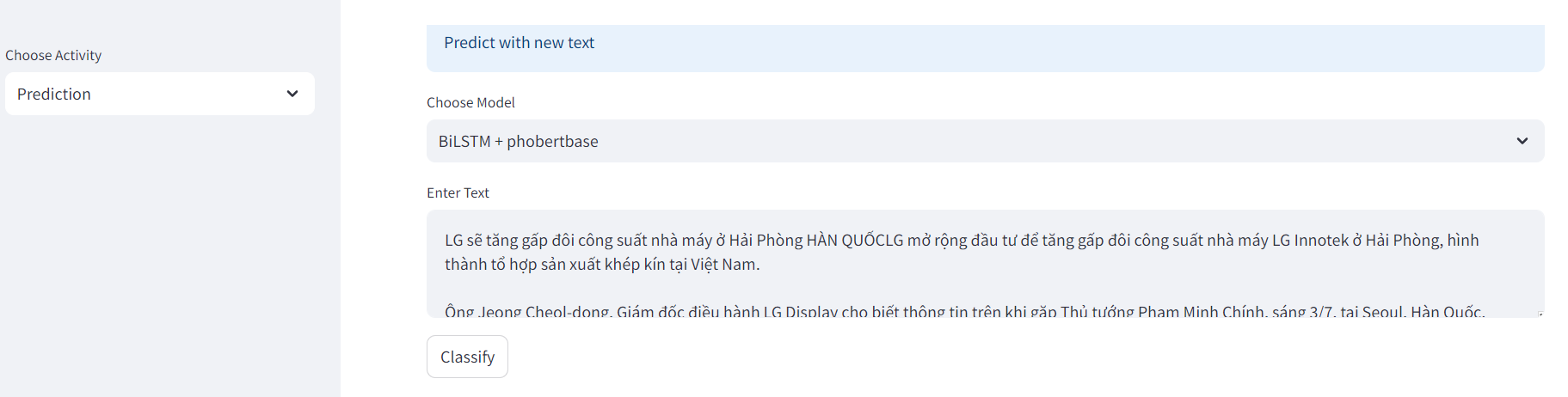
*Trang chủ của phần mềm*



Bấm vào đây để chọn task muốn thực hiện

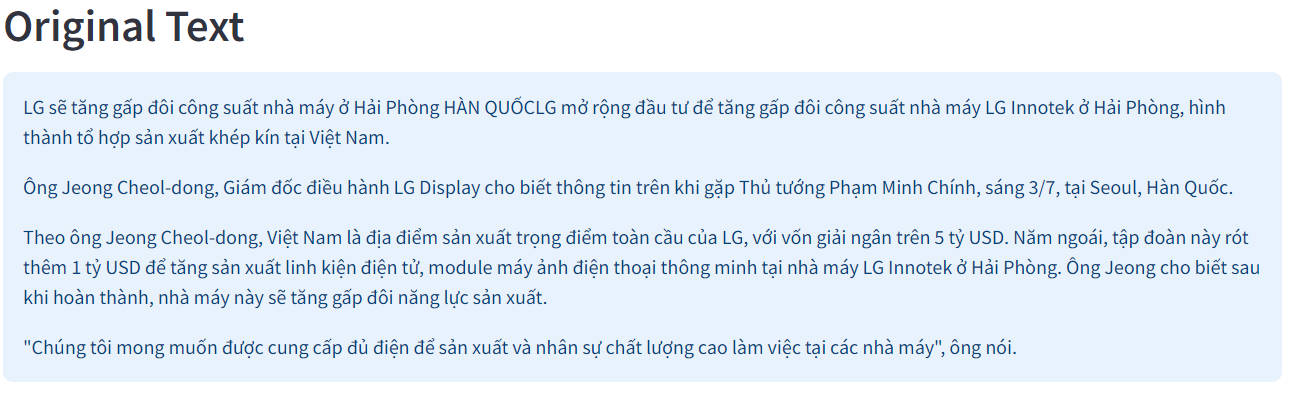


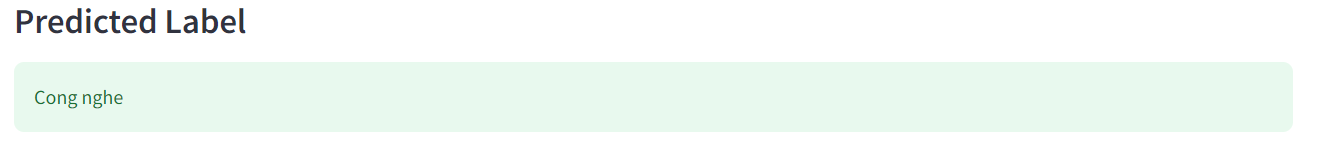
Ở mục train và evaluate sẽ có cách cài đặt các mô hình, siêu tham số và kết quả đánh giá mô hình





ở mục dự đoán, bấm chọn model sau đó nhập vào văn bản mới và chọn Classify để dự đoán





*Kết quả dự đoán*