

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



BÁO CÁO ĐỒ ÁN

**Môn học: Khai thác dữ liệu
truyền thông xã hội**

**Phát hiện tin tức có đáng tin cậy hay
không dựa trên ngôn ngữ tự nhiên**

Giảng viên hướng dẫn: ThS. Nguyễn Thành Luân

Sinh viên thực hiện:		
STT	Họ tên	MSSV
1	Kiều Thanh Danh	22550001
2	Nguyễn Trung Hiếu	22550004
3	Ngô Trần Tuấn Phong	22550016
4	Đào Nhâm Phúc	22550017
5	Phạm Hoàng Sang	22550019

Thành phố Hồ Chí Minh – 04/2024

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

Độc Lập - Tự Do - Hạnh Phúc

TP. HCM, ngày 05 tháng 04 năm 2024

NHẬN XÉT ĐỒ ÁN MÔN HỌC

Tên đồ án:

Phát hiện tin tức có đáng tin cậy hay không dựa trên ngôn ngữ tự nhiên

Nhóm SV thực hiện:

Họ và tên: Kiều Thanh Danh

MSSV: 22550001

Họ và tên: Nguyễn Trung Hiếu

MSSV: 22550004

Họ và tên: Ngô Trần Tuấn Phong

MSSV: 22550016

Họ và tên: Đào Nhâm Phúc

MSSV: 22550017

Họ và tên: Phạm Hoàng Sang

MSSV: 22550019

Giảng viên phụ trách:

ThS. Nguyễn Thành Luân

Đánh giá Đồ án:

1. Về cuốn báo cáo:

Số trang	_____	Số chương	_____
Số bảng số liệu	_____	Số hình vẽ	_____
Số tài liệu tham khảo	_____	Sản phẩm	_____

Một số nhận xét về hình thức cuốn báo cáo:

2. Về nội dung nghiên cứu:

3. Về thái độ làm việc của sinh viên:

Đánh giá chung:

Điểm sinh viên:

Kiều Thanh Danh:...../10

Nguyễn Trung Hiếu:...../10

Ngô Trần Tuấn Phong:...../10

Đào Nhâm Phúc:...../10

Phạm Hoàng Sang:...../10

Người nhận xét
(Ký tên và ghi rõ họ tên)

LỜI CẢM ƠN

Trải qua thời gian học tập và rèn luyện cùng Thầy và các bạn, bản thân chúng em đã tiếp thu được rất nhiều những kiến thức.

Chúng em xin chân thành cảm ơn Thầy Nguyễn Thành Luân đã tận tình giảng dạy, hướng dẫn và giúp đỡ chúng em trong suốt quá trình học tập.

Cuối cùng, chúng em chúc Thầy công tác tốt và luôn dồi dào sức khỏe để có thể cống hiến hết mình cho những thế hệ sinh viên tiếp theo.

Chúng em xin chân thành cảm ơn!

Sinh viên thực hiện

Kiều Thanh Danh

Nguyễn Trung Hiếu

Ngô Trần Tuấn Phong

Đào Nhâm Phúc

Phạm Hoàng Sang

ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN
Độc Lập - Tự Do - Hạnh Phúc

ĐỀ CƯƠNG CHI TIẾT

TÊN ĐỒ ÁN: Phát hiện tin tức có đáng tin cậy hay không dựa trên ngôn ngữ tự nhiên	
Cán bộ hướng dẫn: Ts. Nguyễn Thành Luân	
Thời gian thực hiện: Từ ngày 12-03-2024 đến ngày 12-04-2024	
Sinh viên thực hiện: Kiều Thanh Danh – 22550001 Nguyễn Trung Hiếu – 22550004 Ngô Trần Tuấn Phong – 22550016 Đào Nhâm Phúc – 22550017 Phạm Hoàng Sang – 22550019	
Nội dung đồ án: <ul style="list-style-type: none">- Giới thiệu về mô hình Bert.- Tìm hiểu về cách mô hình Bert hoạt động.- Ứng dụng các mô hình BERT, PhoBERT, RoBERTa vào xử lý ngôn ngữ tự nhiên để phát hiện tin tức chính thống và không chính thống.- Kết luận.	
Kế hoạch thực hiện: Tìm Data về các tin tức chính thống và không chính thống sau đó thực hiện train các mô hình BERT, PhoBERT, RoBERTa để sử dụng các mô hình xác định được đoạn text nhập vào có đáng tin cậy hay không	
Xác nhận của Giảng viên (Ký tên và ghi rõ họ tên)	TP. HCM, ngày 08 tháng 04 năm 2024 Sinh viên (Ký tên và ghi rõ họ tên) Kiêu Thanh Danh, Nguyễn Trung Hiếu, Ngô Trần Tuấn Phong, Đào Nhâm Phúc, Phạm Hoàng Sang

MỤC LỤC

I.	Giới thiệu chung.....	1
1.	Tầm quan trọng của BERT hiện nay.....	1
2.	Nền tảng của BERT	1
II.	Tìm hiểu về các mô hình Bert.....	2
1.	Fine-tuning model BERT.....	2
2.	Masked ML (MLM).....	2
3.	Next Sentence Prediction (NSP)	2
III.	Các kiến trúc model BERT	3
1.	Các thông số cơ bản của BERT	3
2.	Hai phiên bản chính của BERT.....	3
2.1	BERT Base	3
2.2	BERT Large	3
3.	Mô hình PhoBERT.....	4
3.1)	Kiến trúc.....	4
3.2)	Huấn luyện.....	4
3.3)	Cải tiến	4
3.4)	Ứng dụng	4
4.	Mô hình RoBERTa	4
4.1)	Kiến trúc.....	4
4.2)	Huấn luyện.....	5
4.3)	Cải tiến	5
4.4)	Ứng dụng	5
IV.	Ứng dụng vào xử lý ngôn ngữ tự nhiên để phát hiện tin tức chính thống và không chính thống. ..	5
1.	Giới thiệu đề tài.....	5
2.	Cách thu thập dữ liệu	6
2.1	Tin chính thống	6
2.2	Tin không chính thống	6
2.3	Thu thập tin tức từ các nguồn chính thống và không chính thống.....	6
3.	Xử lý dữ liệu và gán nhãn	7
4.	Sử dụng các mô hình để huấn luyện	8
4.1	BERT.....	8
4.2	PhoBERT.....	8
4.3	RoBERTa.....	9
5.	Đánh giá giữa các mô hình	9
V.	Kết luận.....	10

TÀI LIỆU THAM KHẢO.....	23
-------------------------	----

DANH MỤC CHỮ VIẾT TẮT

STT	Ký hiệu chữ viết tắt	Chữ viết đầy đủ
1	BERT	Bidirectional Encoder Representations from Transformers
2	NLP	Natural Language Processing
3		

I. Giới thiệu chung

BERT là viết tắt của Bidirectional Encoder Representations from Transformers được hiểu là một mô hình học sâu hay còn gọi là pre-train model, học ra các vector đại diện theo ngữ cảnh 2 chiều của từ (từ trái qua phải và từ phải qua trái), được sử dụng để transfer sang các bài toán khác trong lĩnh vực xử lý ngôn ngữ tự nhiên.

BERT được coi như là đột phá lớn trong Machine Learning bởi vì khả năng ứng dụng của nó vào nhiều bài toán NLP khác nhau: Question Answering, Natural Language Inference,... với kết quả rất tốt.

1. Tầm quan trọng của BERT hiện nay

Một trong những thách thức lớn nhất của NLP là vấn đề dữ liệu. Trên internet có hàng tá dữ liệu, nhưng những dữ liệu đó không đồng nhất; mỗi phần của nó chỉ được dùng cho một mục đích riêng biệt, do đó khi giải quyết một bài toán cụ thể, ta cần trích ra một bộ dữ liệu thích hợp cho bài toán của mình, và kết quả là ta chỉ có một lượng rất ít dữ liệu

Do đó một vấn đề được đặt ra: làm thế nào để tận dụng được nguồn dữ liệu vô cùng lớn có sẵn để giải quyết bài toán của mình. Đó là tiền đề cho một kỹ thuật mới ra đời: Transfer Learning. BERT là một trong những đại diện ưu tú nhất trong Transfer Learning cho NLP, nó gây tiếng vang lớn không chỉ bởi kết quả mang lại trong nhiều bài toán khác nhau, mà còn bởi vì nó hoàn toàn miễn phí, tất cả chúng ta đều có thể sử dụng BERT cho bài toán của mình.

2. Nền tảng của BERT

BERT sử dụng Transformer là một mô hình attention (attention mechanism) học mối tương quan giữa các từ (hoặc 1 phần của từ) trong một văn bản. Transformer gồm có 2 phần chính: Encoder và Decoder, encoder thực hiện đọc dữ liệu đầu vào và decoder đưa ra dự đoán. Ở đây, BERT chỉ sử dụng Encoder.

Khác với các mô hình directional (các mô hình chỉ đọc dữ liệu theo 1 chiều duy nhất - trái→phải, phải→trái) đọc dữ liệu theo dạng tuần tự, Encoder đọc toàn bộ dữ liệu trong 1 lần, việc này làm cho BERT có khả năng huấn luyện dữ liệu theo cả hai chiều, qua đó mô hình có thể học được ngữ cảnh (context) của từ tốt hơn bằng cách sử dụng những từ xung quanh nó (phải&trái). [Hình 1]

II. Tìm hiểu về các mô hình Bert

1. Fine-tuning model BERT

Một điểm đặc biệt ở BERT mà các model embedding trước đây chưa từng có đó là kết quả huấn luyện có thể fine-tuning được. Chúng ta sẽ thêm vào kiến trúc model một output layer để tùy biến theo tác vụ huấn luyện. [Hình 2]

Một kiến trúc tương tự được sử dụng cho cả pretrain-model và fine-tuning model. Chúng ta sử dụng cùng một tham số pretrain để khởi tạo mô hình cho các tác vụ down stream khác nhau. Trong suốt quá trình fine-tuning thì toàn bộ các tham số của layers học chuyển giao sẽ được fine-tune.

Trong quá trình huấn luyện chúng ta sẽ fine-tune lại toàn bộ các tham số của model BERT đã cut off top linear layer và huấn luyện lại từ đầu các tham số của linear layer mà chúng ta thêm vào kiến trúc model BERT để customize lại phù hợp với bài toán.

2. Masked ML (MLM)

Là một tác vụ cho phép chúng ta fine-tuning lại các biểu diễn từ trên các bộ dữ liệu unsupervised-text bất kỳ. Chúng ta có thể áp dụng Masked ML cho những ngôn ngữ khác nhau để tạo ra biểu diễn embedding cho chúng. Các bộ dữ liệu của tiếng anh có kích thước lên tới vài vài trăm tới vài nghìn GB được huấn luyện trên BERT đã tạo ra những kết quả khá ấn tượng. [Hình 3]

Hàm loss function của BERT sẽ bỏ qua mất mát từ những từ không bị che dấu và chỉ đưa vào mất mát của những từ bị che dấu. Do đó mô hình sẽ hội tụ lâu hơn nhưng đây là đặc tính bù trừ cho sự gia tăng ý thức về bối cảnh. Việc lựa chọn ngẫu nhiên 15% số lượng các từ bị che dấu cũng tạo ra vô số các kịch bản input cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.

3. Next Sentence Prediction (NSP)

Đây là một bài toán phân loại học có giám sát với 2 nhãn (hay còn gọi là phân loại nhị phân). Input đầu vào của mô hình là một cặp câu (pair-sequence) sao cho 50% câu thứ 2 được lựa chọn là câu tiếp theo của câu thứ nhất và 50% được lựa chọn một cách ngẫu nhiên từ bộ văn bản mà không có mối liên hệ gì với câu thứ nhất. Nhãn của mô hình sẽ tương ứng với IsNext khi cặp câu là liên tiếp hoặc NotNext nếu cặp câu không liên tiếp. [Hình 4]

Véc tơ input sẽ bằng tổng của cả ba thành phần embedding theo từ, câu và vị trí.

III. Các kiến trúc model BERT

1. Các thông số cơ bản của BERT

BERT đã trở thành một trong những mô hình ngôn ngữ sâu học sâu nhất và phổ biến nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Hiện tại có nhiều phiên bản khác nhau của model BERT. Các phiên bản đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở 3 tham số:

- L : số lượng các block sub-layers trong transformer,
- H : kích thước của embedding véc tơ (hay còn gọi là hidden size),
- A : Số lượng head trong multi-head layer, mỗi một head sẽ thực hiện một self-attention.

2. Hai phiên bản chính của BERT

BERT hiện tại đang có hai phiên bản chính là BERT Base và BERT Large.

Tên gọi của 2 kiến trúc bao gồm:

- $BERT_{BASE}$ ($L = 12, H = 768, A = 12$): Tổng tham số 110 triệu.
- $BERT_{LARGE}$ ($L = 24, H = 1024, A = 16$): Tổng tham số 340 triệu.

2.1 BERT Base

BERT Base là phiên bản nhỏ của BERT, nhưng vẫn rất mạnh mẽ với các đặc điểm sau:

- *Kiến trúc*: BERT Base bao gồm 12 lớp Transformer và mỗi lớp có 12 head attention.
- *Tham số*: Tổng cộng khoảng 110 triệu tham số.
- *Kích thước Embedding*: Mỗi từ được biểu diễn trong không gian vector 768 chiều.
- *Sử dụng phổ biến*: BERT Base thường được sử dụng trong các ứng dụng NLP đòi hỏi tính linh hoạt và hiệu suất cao, như phân loại văn bản, dự đoán từ tiếp theo và tóm tắt văn bản.

2.2 BERT Large

BERT Large là phiên bản lớn hơn và mạnh mẽ hơn của BERT Base, với các đặc điểm sau:

- *Kiến trúc*: BERT Large có 24 lớp Transformer và mỗi lớp có 16 head attention, tạo ra một mạng lưới sâu hơn và phức tạp hơn so với BERT Base.
- *Tham số*: Tổng cộng khoảng 340 triệu tham số, gấp khoảng 3 lần so với BERT Base.
- *Kích thước Embedding*: Mỗi từ được biểu diễn trong không gian vector 1024 chiều, giúp BERT Large có khả năng biểu diễn ngôn ngữ chi tiết hơn.

- *Sử dụng phổ biến*: BERT Large thường được sử dụng trong các nhiệm vụ phức tạp hơn và yêu cầu hiểu biết sâu rộng về ngữ cảnh, như dịch máy và phát hiện thực thể trong văn bản y khoa.

3. Mô hình PhoBERT

3.1) Kiến trúc

PhoBERT là một biến thể của mô hình BERT được tinh chỉnh và cải tiến để phù hợp với ngôn ngữ và nhiệm vụ xử lý ngôn ngữ tự nhiên tiếng Việt.

Mô hình PhoBERT vẫn dựa trên kiến trúc Transformer với một hoặc nhiều lớp encoder và decoder như trong kiến trúc gốc của BERT.

3.2) Huấn luyện

PhoBERT được huấn luyện trên một lượng lớn dữ liệu tiếng Việt. Quá trình huấn luyện có thể sử dụng các biện pháp tăng cường dữ liệu và các kỹ thuật tiền xử lý dữ liệu tiếng Việt để cải thiện hiệu suất và khả năng tổng quát hoá của mô hình.

Sau khi huấn luyện, PhoBERT có thể được fine-tuned cho các nhiệm vụ cụ thể như phân loại văn bản, dự đoán từ khoá, tóm tắt văn bản và các nhiệm vụ NLP khác.

3.3) Cải tiến

Một số điều chỉnh có thể được thực hiện để tinh chỉnh mô hình BERT gốc cho tiếng Việt, bao gồm điều chỉnh siêu tham số huấn luyện, sử dụng tiếng Việt phong phú hơn và tối ưu hoá kiến trúc mô hình cho ngôn ngữ cụ thể.

3.4) Ứng dụng

PhoBERT được thiết kế để cải thiện hiệu suất của BERT trong các tác vụ NLP tiếng Việt như phân loại văn bản, dịch máy, tóm gọn văn bản, và nhiều ứng dụng khác.

Hiệu suất của mô hình có thể đánh giá thông qua các thước đo như độ chính xác, độ phủ và thời gian đáp ứng.

4. Mô hình RoBERTa

RoBERTa là một mô hình học sâu dựa trên kiến trúc Transformer được phát triển bởi Facebook AI.

4.1) Kiến trúc

RoBERTa được dựa trên kiến trúc Transformer, được sử dụng để mã hoá và phân tích các chuỗi văn bản.

Kiến trúc này bao gồm một số lớp Transformer, mỗi lớp bao gồm các đầu vào embedding, các lớp encoded và decoded.

RoBERTa sử dụng một số lượng lớn các lớp Transformer và được huấn luyện với nhiều hơn các bài toán so với BERT.

4.2) Huấn luyện

Mô hình RoBERTa được huấn luyện trên một lượng lớn dữ liệu không có nhãn từ các nguồn khác nhau như tiếng Anh từ Wikipedia, Common Crawl, BookCorpus và một số nguồn khác.

Điều này giúp mô hình học được diễn biến ngôn ngữ tự nhiên tổng quát hơn và phù hợp với nhiều dữ liệu khác nhau.

4.3) Cải tiến

Một số cải tiến của RoBERTa so với BERT bao gồm: loại bỏ token type embeddings, huấn luyện trên các câu có chiều dài tối đa cố định, sử dụng chuỗi mask không overlapping và huấn luyện với thêm dữ liệu và siêu tham số.

4.4) Ứng dụng

RoBERTa có thể được sử dụng cho nhiều tác vụ NLP như phân loại văn bản, dự đoán từ tiếp theo, dịch máy và tóm tắt văn bản.

Mô hình này đã đạt được kết quả ấn tượng trên nhiều benchmark và bài toán NLP.

IV. Ứng dụng vào xử lý ngôn ngữ tự nhiên để phát hiện tin tức chính thống và không chính thống.

1. Giới thiệu đề tài

Nghiên cứu này tập trung vào việc phát triển hệ thống phân loại tin tức sử dụng các công nghệ xử lý ngôn ngữ tự nhiên tiên tiến như BERT, PhoBERT và RoBERTa, với mục tiêu đánh giá độ tin cậy của các tin tức được chia sẻ trên mạng xã hội.

Trong môi trường truyền thông hiện đại, việc đánh giá độ tin cậy của thông tin trở nên ngày càng quan trọng, đặc biệt là trên các nền tảng mạng xã hội nơi thông tin lan truyền một cách nhanh chóng và rộng lớn. Sự lan truyền của tin tức không chính xác hoặc thiếu minh bạch có thể gây ra những hậu quả nghiêm trọng đối với xã hội và cá nhân.

Để giải quyết vấn đề này, nghiên cứu này đề xuất sử dụng các mô hình học sâu như BERT, PhoBERT và RoBERTa, các mô hình đã được chứng minh là hiệu quả trong việc hiểu và biểu diễn ngôn ngữ tự nhiên. Bằng cách sử dụng các mô hình này, hệ thống sẽ có khả năng phân tích và đánh giá các thông tin từ các tin tức trên mạng xã hội, từ đó xác định độ tin cậy của chúng.

Quy trình làm việc có thể bao gồm việc thu thập dữ liệu từ các nguồn tin tức trên mạng xã hội, internet để chuẩn bị cho việc đưa vào mô hình, huấn luyện các mô hình BERT, PhoBERT và RoBERTa trên dữ liệu đã thu thập, và cuối cùng là đánh giá hiệu suất của hệ thống phân loại dựa trên độ tin cậy của các tin tức được phân loại.

Kết quả của nghiên cứu sẽ cung cấp thông tin quan trọng về độ tin cậy của các tin tức trên mạng xã hội, từ đó hỗ trợ người đọc và cộng đồng trong việc đánh giá và chọn lựa thông tin một cách thông minh và hiệu quả.

2. Cách thu thập dữ liệu

Trong quá trình thu thập tin tức để đánh giá và phân loại, việc hiểu rõ về khái niệm tin chính thống và không chính thống là rất quan trọng.

2.1 Tin chính thống

Tin tức chính thống là những thông tin được công bố và phổ biến bởi các nguồn thông tin uy tín và có trách nhiệm, như các tờ báo, trang tin tức, hoặc cơ quan truyền thông có uy tín và độc lập. Những tin chính thống thường tuân thủ các nguyên tắc báo chí như độc lập, minh bạch, và trung thực. Các tin tức chính thống thường được xác định bởi sự đa dạng trong các nguồn thông tin, việc kiểm chứng và xác thực thông tin trước khi công bố, cũng như việc trình bày thông tin một cách khách quan và công bằng.

2.2 Tin không chính thống

Ngược lại, tin tức không chính thống thường được định nghĩa là những thông tin không tuân thủ các nguyên tắc cơ bản của báo chí và thường được phổ biến với mục đích lợi ích cá nhân hoặc mục tiêu nhất định. Những tin tức này có thể bao gồm việc biến tướng sự thật, tin đồn không chứng minh được, hoặc việc sử dụng tiêu đề gây chú ý để thu hút lượt xem mà không cung cấp thông tin đầy đủ và chính xác.

2.3 Thu thập tin tức từ các nguồn chính thống và không chính thống

Trong quá trình thu thập tin tức để đánh giá và phân loại, chúng ta cần sự cân nhắc và kỹ lưỡng trong việc lựa chọn nguồn thông tin. Các nguồn tin tức chính thống như Vnexpress, Báo điện tử Dân trí, Bảo vệ pháp luật, Lao động, Nhân dân và Tuổi Trẻ thường cung cấp những thông tin được kiểm chứng và đáng tin cậy.

Tuy nhiên, các trang tin không chính thống như Saigon24.net, Genk.vn thường xuất hiện với nội dung giật gân và cảm xúc, thường có xu hướng tạo ra tiêu đề gây chú ý để thu hút lượt xem mà không cung cấp thông tin chính xác và đầy đủ.

Để thu thập tin tức đủ lượng và đa dạng cho việc đánh giá, có thể sử dụng các công cụ tự động như web crawler để thu thập dữ liệu. Sau đó, dữ liệu thu thập được có thể được xem xét và đánh giá để xác định rõ ràng giữa tin chính thống và không chính thống, từ đó đánh nhãn cho mỗi mẫu tin tức.

Các nguồn web được sử dụng trong quá trình thu tập như sau:

- 1) <https://emdep.vn/>
- 2) <https://genk.vn/>
- 3) <https://kenh14.vn/>
- 4) <https://laodong.vn/>
- 5) <https://nhandan.vn/>
- 6) <https://soha.vn/>
- 7) <https://thanhnien.vn/>
- 8) <https://tuoitre.vn/>
- 9) <https://vietnamnet.vn/>
- 10) <https://vnexpress.net/>

3. Xử lý dữ liệu và gán nhãn

Sau khi thu thập dữ liệu từ các nguồn tin, tiến trình tiếp theo là phân biệt xem mỗi liên kết (URL) trong dữ liệu thu thập có đến từ nguồn tin đáng tin cậy hay không đáng tin cậy. Quá trình này có thể thực hiện một cách tự động thông qua việc phân tích các đặc điểm của từng nguồn tin, hoặc thông qua một quy trình đánh giá thủ công. Sau khi đã xác định được tính đáng tin cậy của mỗi liên kết, chúng ta gán nhãn (label) cho từng mẫu dữ liệu dựa trên điều này.

Khi quá trình crawling dữ liệu đã hoàn thành và chúng ta đã thu thập được thông tin về tiêu đề (title) và nội dung (content) của mỗi mẫu tin tức, chúng ta tiến hành tạo một cột mới trong bộ dữ liệu và gán nhãn cho từng mẫu dữ liệu dựa trên quyết định từ bước phân biệt trước đó. Cụ thể, chúng ta sẽ gán nhãn 1 cho các mẫu dữ liệu được xác định là tin đáng tin cậy và nhãn 0 cho các mẫu dữ liệu được xác định là không đáng tin cậy.

Quá trình này giúp chúng ta tạo ra một bộ dữ liệu được gán nhãn đầy đủ và chuẩn xác, sẵn sàng cho quá trình huấn luyện mô hình máy học. Việc gán nhãn dựa trên nội dung của mỗi mẫu tin tức giúp mô hình hiểu được sự tương quan giữa tiêu đề, nội dung và tính đáng tin cậy của thông tin, từ đó nâng cao hiệu suất của mô hình trong việc phân loại tin tức.

4. Sử dụng các mô hình để huấn luyện

4.1 BERT

BERT đã đạt được sự chú ý lớn từ cộng đồng nghiên cứu và làm việc trong lĩnh vực xử lý ngôn ngữ tự nhiên. Với kiến trúc transformer, BERT có khả năng hiểu được ngôn ngữ thông qua việc đào tạo trên một lượng lớn văn bản không được gán nhãn trên Internet. Điểm đặc biệt của BERT là khả năng hiểu được ngữ cảnh và mối quan hệ giữa các từ trong câu thông qua việc xử lý các mảnh văn bản dưới dạng "mặt nạ".

Nhờ vào việc sử dụng pre-training và fine-tuning, BERT đã đạt được kết quả ấn tượng trên nhiều nhiệm vụ ngôn ngữ tự nhiên như phân loại văn bản, dịch máy, và tiên xử lý ngôn ngữ tự nhiên.

Với sự đóng góp của BERT, lĩnh vực xử lý ngôn ngữ tự nhiên đã có những tiến bộ đáng kể, và nền tảng này đã mở ra nhiều cơ hội mới trong việc hiểu và tương tác với ngôn ngữ tự nhiên.

Quá trình huấn luyện:

[Hình 5], [Hình 6], [Hình 7], [Hình 8], [Hình 9], [Hình 10], [Hình 11], [Hình 12]

4.2 PhoBERT

PhoBERT có một số điểm chính như sau:

Đây là một pre-trained được huấn luyện monolingual language, tức là chỉ huấn luyện dành riêng cho tiếng Việt. Việc huấn luyện dựa trên kiến trúc và cách tiếp cận giống RoBERTa của Facebook được Facebook giới thiệu giữa năm 2019. Đây là một cái tiến so với BERT trước đây.

Tương tự như Bert, PhoBert cũng có 2 phiên bản là PhoBertbase với 12 transformers block và PhoBertlarge với 24 transformers block

PhoBERT được train trên khoảng 20GB dữ liệu bao gồm khoảng 1GB Vietnamese Wikipedia corpus và 19GB còn lại lấy từ Vietnamese news corpus. Đây là một lượng dữ liệu khá ổn để train một mô hình như BERT.

PhoBERT sử dụng RDRSegmenter của VnCoreNLP để tách từ cho dữ liệu đầu vào trước khi qua BPE encoder.

Như đã nói ở trên, do tiếp cận theo tư tưởng của RoBERTa, PhoBERT chỉ sử dụng task Masked Language Model để train, bỏ đi task Next Sentence Prediction.

Với việc sử dụng PhoBERT, các nhà nghiên cứu và nhà phát triển ứng dụng có thể tận dụng khả năng mạnh mẽ của mô hình này trong nhiều tác vụ ngôn ngữ tự nhiên như phân loại văn bản, phát hiện cảm xúc, dịch máy và nhiều ứng dụng khác.

Nhờ vào sự phát triển của PhoBERT, việc nghiên cứu và phát triển các ứng dụng ngôn ngữ tự nhiên cho cộng đồng Việt Nam đã được đẩy mạnh, đồng thời giúp nâng cao hiệu suất và chất lượng của các ứng dụng này trên môi trường ngôn ngữ Việt.

Quá trình huấn luyện:

[Hình 13], [Hình 14], [Hình 15], [Hình 16], [Hình 17], [Hình 18], [Hình 19], [Hình 20]

4.3 RoBERTa

RoBERTa được giới thiệu bởi Facebook là một phiên bản được huấn luyện lại của BERT với một phương pháp huấn luyện tốt hơn với dữ liệu được tăng gấp 10 lần.

Để tăng cường quá trình huấn luyện, RoBERTa không sử dụng cơ chế dự đoán câu kế tiếp (NSP) từ BERT mà sử dụng kỹ thuật mặt nạ động (dynamic masking), theo đó các token mặt nạ sẽ bị thay đổi trong quá trình huấn luyện. Sử dụng kích thước batch lớn hơn cho thấy hiệu quả tốt hơn khi huấn luyện.

Một điều quan trọng nữa, RoBERTa sử dụng 160GB văn bản để huấn luyện. Trong đó, 16GB là sách và Wikipedia tiếng Anh được sử dụng trong huấn luyện BERT. Phần còn lại bao gồm CommonCrawl News dataset (63 triệu bản tin, 76 GB), ngữ liệu văn bản Web (38 GB) và Common Crawl Stories (31 GB). Mô hình này được huấn luyện với GPU của Tesla 1024 V100 trong một ngày.

Kết quả là, RoBERTa vượt trội hơn cả BERT và XLNet trên dữ liệu đánh giá GLUE:

[Hình 21]

Quá trình huấn luyện:

[Hình 22], [Hình 23], [Hình 24], [Hình 25], [Hình 26], [Hình 27], [Hình 28], [Hình 29], [Hình 30], [Hình 31], [Hình 32], [Hình 33], [Hình 34]

5. Đánh giá giữa các mô hình

Kết quả của các mô hình trong thử nghiệm này là một phần quan trọng đối với việc đánh giá hiệu suất của các mô hình xử lý ngôn ngữ tự nhiên.

Đầu tiên, mô hình BERT đã cho thấy một hiệu suất ấn tượng với tỉ lệ chính xác đạt 0.9391. Điều này chỉ ra rằng BERT đã có khả năng hiểu và phân loại văn bản một cách chính xác đáng kể. [Hình 35]

Tiếp theo, PhoBERT đã đạt được tỉ lệ chính xác là 0.9142 trong thử nghiệm này. Kết quả này cho thấy PhoBERT không chỉ hiệu quả trong việc đào tạo trên dữ liệu tiếng Việt mà còn có hiệu suất gần bằng mô hình gốc, BERT. [Hình 36]

Cuối cùng, RoBERTa đã đạt được tỉ lệ chính xác 0.9461, vượt trội hơn cả hai mô hình trước. Kết quả này cho thấy RoBERTa có khả năng xử lý ngôn ngữ tự nhiên tốt hơn và đạt được hiệu suất cao trong việc phân loại tin tức là đáng tin cậy hay không tin cậy. [Hình 37]

Nhìn chung, việc đánh giá kết quả này đã cung cấp cái nhìn tổng quan về hiệu suất của ba mô hình khác nhau, từ đó giúp cho các nhà nghiên cứu và phát triển ứng dụng có cái nhìn rõ ràng hơn về lựa chọn mô hình phù hợp cho nhu cầu cụ thể của họ.

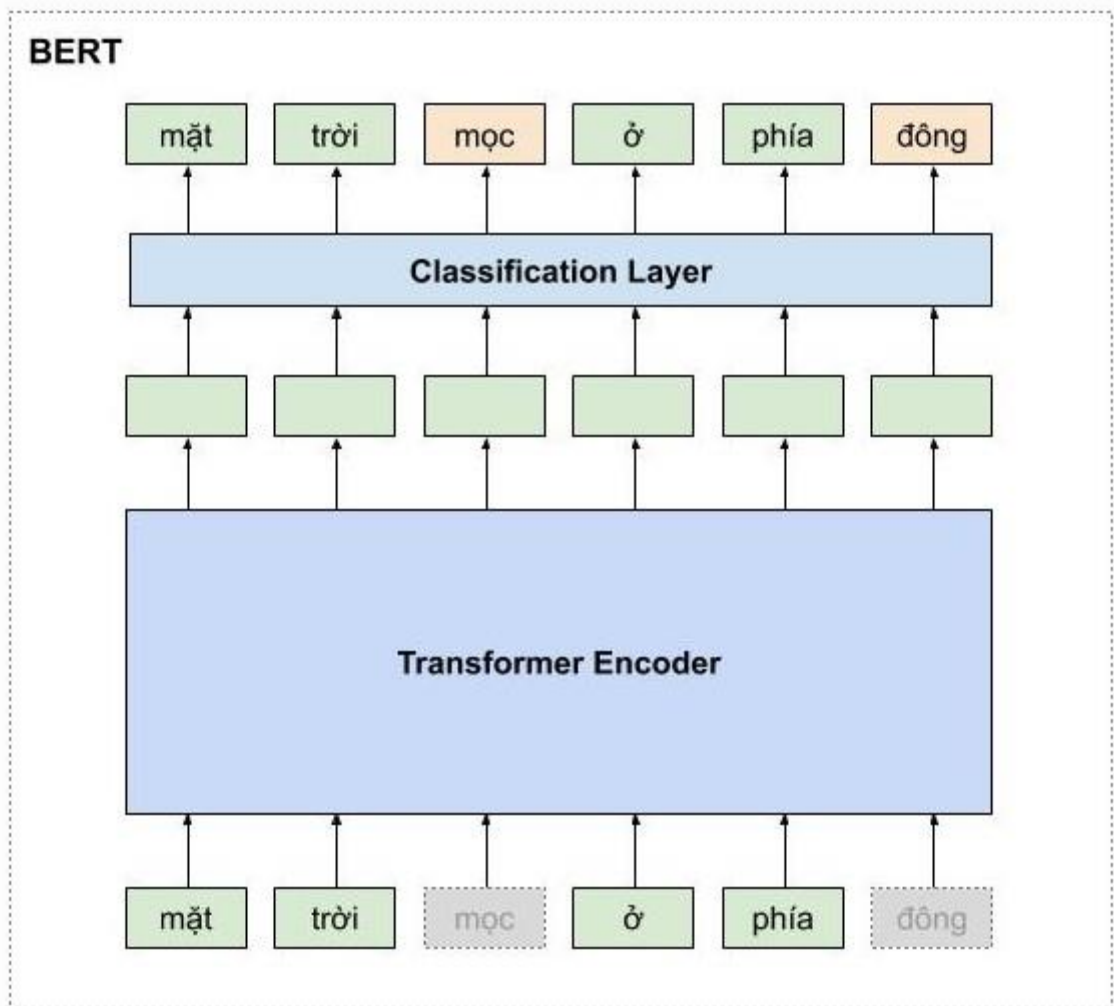
V. Kết luận

Kết luận của nghiên cứu này là một bước quan trọng trong việc ứng dụng các mô hình học sâu như BERT, PhoBERT và RoBERTa để đánh giá và phân biệt độ tin cậy của các nguồn tin trên mạng. Kết quả cho thấy rằng, thông qua việc huấn luyện và sử dụng các mô hình này, chúng ta có thể đạt được kết quả rất khả quan trong việc phân loại tin đáng tin cậy và không đáng tin cậy.

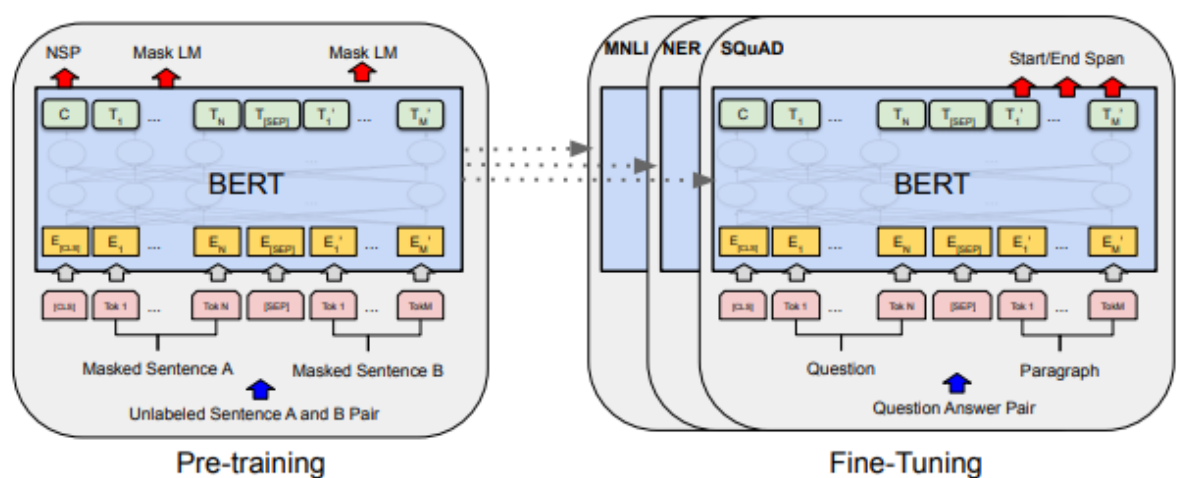
Tỷ lệ huấn luyện để phân biệt giữa các nguồn tin đã được đánh giá với mức độ chính xác cao, điều này cho thấy tính hiệu quả và tiềm năng của các mô hình học sâu trong việc xử lý ngôn ngữ tự nhiên và phân loại tin tức. Cụ thể, việc sử dụng BERT, PhoBERT và RoBERTa đã cho thấy khả năng phân biệt rõ ràng giữa các nguồn tin chính thống và không chính thống, đồng thời cung cấp cái nhìn tổng quan và đáng tin cậy về tin tức trên mạng.

Bằng cách sử dụng các mô hình này, chúng ta có thể tạo ra các hệ thống tự động hoặc hỗ trợ quyết định cho các tổ chức truyền thông, cơ quan chính phủ, và cộng đồng mạng trong việc xác định độ tin cậy của các nguồn tin, giúp họ có thể tiếp cận thông tin một cách chín chắn và đáng tin cậy hơn. Điều này không chỉ giúp cải thiện môi trường truyền thông mà còn góp phần vào việc xây dựng một xã hội thông tin hiểu biết và minh bạch hơn.

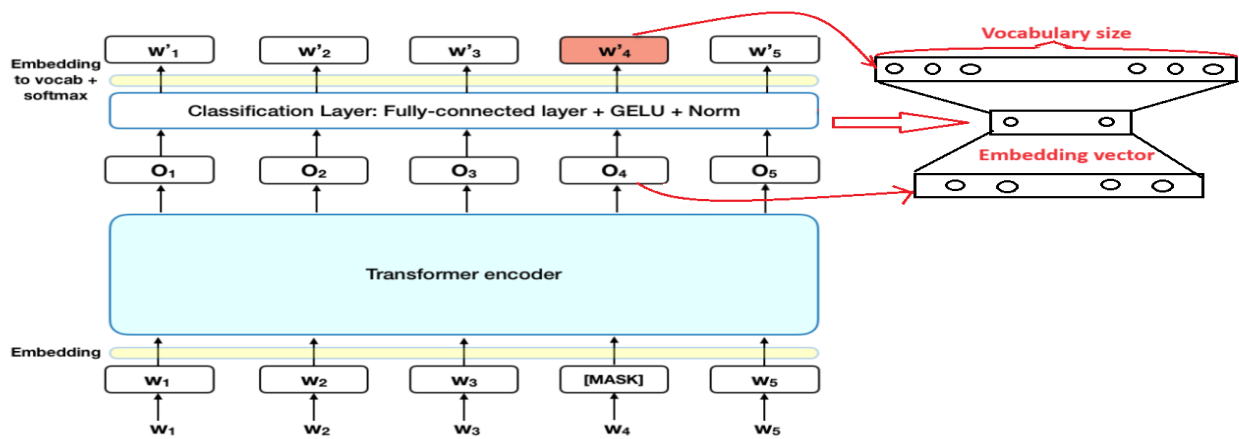
PHỤ LỤC HÌNH



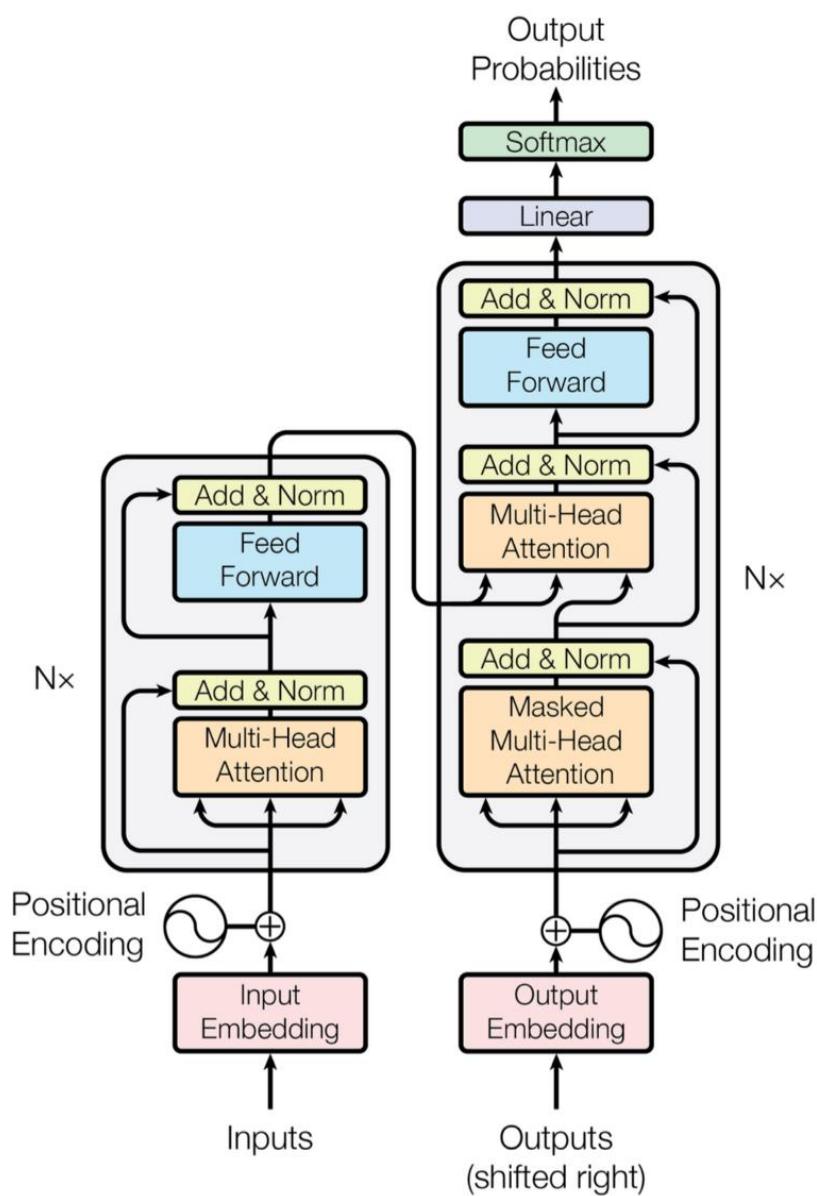
Hình 1: Transformer



Hình 2: Toàn bộ tiến trình pre-training và fine-tuning của BERT.



Hình 3: Sơ đồ kiến trúc BERT cho tác vụ Masked ML.



Hình 4: Sơ đồ kiến trúc model BERT cho tác vụ NSP.

```
import pandas as pd
import torch
from torch.utils.data import Dataset, DataLoader
from transformers import BertTokenizer, BertForSequenceClassification
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from tqdm import tqdm
```

Hình 5: Mô hình BERT - Khai báo các thư viện BertTokenizer, BertForSequenceClassification,

```
data = pd.read_excel('news_data1.xlsx')
```

 news_data1.xlsx

Hình 6: Mô hình BERT - Load Data

```
class NewsDataset(Dataset):
    def __init__(self, data, tokenizer, max_length):
        self.data = data
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        title = str(self.data['title'].iloc[idx])
        content = str(self.data['content'].iloc[idx])
        label = int(self.data['labels'].iloc[idx])

        input_text = title + " " + content

        encoding = self.tokenizer.encode_plus(
            input_text,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )

        input_ids = encoding['input_ids'].squeeze(0)
        attention_mask = encoding['attention_mask'].squeeze(0)

        return {
            'input_ids': input_ids,
            'attention_mask': attention_mask,
            'labels': torch.tensor(label, dtype=torch.long)
        }
```

Hình 7: Mô hình BERT = Xác định Lớp tập dữ liệu

```
tokenizer = BertTokenizer.from_pretrained('bert-base-multilingual-cased')
model = BertForSequenceClassification.from_pretrained('bert-base-multilingual-cased', num_labels=2)
```

Hình 8: Mô hình BERT = Khởi tạo mô hình và mã thông báo BERT

```
# Prepare data and dataloaders
max_length = 512
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
train_dataset = NewsDataset(train_data, tokenizer, max_length)
test_dataset = NewsDataset(test_data, tokenizer, max_length)

batch_size = 16
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Hình 9: Mô hình BERT - Chuẩn bị dữ liệu và bộ nạp dữ liệu

```
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)
```

Hình 10: Mô hình BERT - Xác định trình tối ưu hóa và lập lịch

```
# Training loop
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
model.train()

num_epochs = 3
for epoch in range(num_epochs):
    print(f'Epoch {epoch + 1}/{num_epochs}')
    running_loss = 0.0

    for batch in tqdm(train_loader):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        optimizer.zero_grad()

        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    avg_loss = running_loss / len(train_loader)
    print(f'Training Loss: {avg_loss:.4f}')

    # Adjust learning rate
    scheduler.step()
```

Hình 11: Mô hình BERT - Vòng đào tạo

```
# Evaluation
model.eval()
y_true = []
y_pred = []

with torch.no_grad():
    for batch in tqdm(test_loader):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        predictions = torch.argmax(logits, dim=1)

        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predictions.cpu().numpy())


# Calculate metrics
accuracy = accuracy_score(y_true, y_pred)
classification_rep = classification_report(y_true, y_pred, target_names=['Unreliable', 'Reliable'])
model.save_pretrained('saved_model1')
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(classification_rep)
```

Hình 12: Mô hình BERT - Sự đánh giá & Tính toán số liệu

```
train_phobert.py > ...
1 import pandas as pd
2 import torch
3 from torch.utils.data import Dataset, DataLoader
4 from transformers import PhobertTokenizer, BertForSequenceClassification
5 from sklearn.model_selection import train_test_split
6 from sklearn.metrics import accuracy_score, classification_report
7 from tqdm import tqdm
8
```

Hình 13: Mô hình PhoBERT - Khai báo các thư viện *PhobertTokenizer*, *BertForSequenceClassification*,

```
data = pd.read_excel('news_data1.xlsx')
```

 news_data1.xlsx

Hình 14: Mô hình PhoBERT - Load data

```
class NewsDataset(Dataset):
    def __init__(self, data, tokenizer, max_length):
        self.data = data
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        title = str(self.data['title'].iloc[idx])
        content = str(self.data['content'].iloc[idx])
        label = int(self.data['labels'].iloc[idx])

        input_text = title + " " + content

        encoding = self.tokenizer.encode_plus(
            input_text,
            add_special_tokens=True,
            max_length=self.max_length,
            padding='max_length',
            truncation=True,
            return_tensors='pt'
        )

        input_ids = encoding['input_ids'].squeeze(0)
        attention_mask = encoding['attention_mask'].squeeze(0)

        return {
            'input_ids': input_ids,
            'attention_mask': attention_mask,
            'labels': torch.tensor(label, dtype=torch.long)
        }
```

Hình 15: Mô hình PhoBERT - Xác định lớp tập dữ liệu

```
tokenizer = PhobertTokenizer.from_pretrained('vinai/phobert-base')
model = BertForSequenceClassification.from_pretrained('vinai/phobert-base', num_labels=2)
```

Hình 16: Mô hình PhoBERT - Khởi tạo mã thông báo và mô hình PhoBERT

```
max_length = 512
train_data, test_data = train_test_split(data, test_size=0.2, random_state=42)
train_dataset = NewsDataset(train_data, tokenizer, max_length)
test_dataset = NewsDataset(test_data, tokenizer, max_length)

batch_size = 16
train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Hình 17: Mô hình PhoBERT - Chuẩn bị dữ liệu và bộ nạp dữ liệu


```
optimizer = torch.optim.AdamW(model.parameters(), lr=2e-5)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=10, gamma=0.1)
```

Hình 18: Mô hình PhoBERT - Xác định trình tối ưu hóa và lập lịch

```
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
model.to(device)
model.train()

num_epochs = 10
for epoch in range(num_epochs):
    print(f'Epoch {epoch + 1}/{num_epochs}')
    running_loss = 0.0

    for batch in tqdm(train_loader):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        optimizer.zero_grad()

        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

        running_loss += loss.item()

    avg_loss = running_loss / len(train_loader)
    print(f'Training Loss: {avg_loss:.4f}')

    # Adjust learning rate
    scheduler.step()
```

Hình 19: Mô hình PhoBERT - Vòng đào tạo

```
model.eval()
y_true = []
y_pred = []

with torch.no_grad():
    for batch in tqdm(test_loader):
        input_ids = batch['input_ids'].to(device)
        attention_mask = batch['attention_mask'].to(device)
        labels = batch['labels'].to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        predictions = torch.argmax(logits, dim=1)

        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predictions.cpu().numpy())

# Calculate metrics
accuracy = accuracy_score(y_true, y_pred)
classification_rep = classification_report(y_true, y_pred, target_names=['Unreliable', 'Reliable'])
model.save_pretrained('saved_model2')
print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(classification_rep)
```

Hình 20: Mô hình PhoBERT - Đánh giá & Tính toán số liệu

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
<i>Single-task single models on dev</i>										
BERT _{LARGE}	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
XLNet _{LARGE}	89.8/-	93.9	91.8	83.8	95.6	89.2	63.6	91.8	-	-
RoBERTa	90.2/90.2	94.7	92.2	86.6	96.4	90.9	68.0	92.4	91.3	-
<i>Ensembles on test (from leaderboard as of July 25, 2019)</i>										
ALICE	88.2/87.9	95.7	90.7	83.5	95.2	92.6	68.6	91.1	80.8	86.3
MT-DNN	87.9/87.4	96.0	89.9	86.3	96.5	92.7	68.4	91.1	89.0	87.6
XLNet	90.2/89.8	98.6	90.3	86.3	96.8	93.0	67.8	91.6	90.4	88.4
RoBERTa	90.8/90.2	98.9	90.2	88.2	96.7	92.3	67.8	92.2	89.0	88.5

Hình 21: So sánh RoBERTa vượt trội hơn cả BERT và XLNet trên dữ liệu đánh giá GLUE:

```

train.py > ...
1  import pandas as pd
2  import torch
3  from torch.utils.data import Dataset, DataLoader
4  from transformers import XLMRobertaTokenizer, XLMRobertaForSequenceClassification
5  from sklearn.metrics import accuracy_score, f1_score, classification_report
6  from tqdm import tqdm

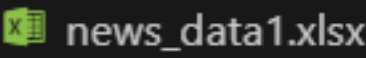
```

Hình 22: Mô hình RoBERTa - Khai báo các thư viện XLMRobertaTokenizer, XLMRobertaForSequenceClassification....

```

data = pd.read_excel('news_data1.xlsx')

```



Hình 23: Mô hình RoBERTa - Load data

```

# Initialize RoBERTa tokenizer
tokenizer = XLMRobertaTokenizer.from_pretrained("xlm-roberta-large")

```

Hình 24: Mô hình RoBERTa - Khởi tạo mã thông báo RoBERTa

```
# Custom dataset class
class NewsDataset(Dataset):
    def __init__(self, data, tokenizer, max_length):
        self.data = data.dropna(subset=['labels']) # Drop rows with missing labels
        if len(self.data) == 0:
            raise ValueError("No valid samples found in the dataset.")
        self.tokenizer = tokenizer
        self.max_length = max_length

    def __len__(self):
        return len(self.data)

    def __getitem__(self, idx):
        title = str(self.data['title'].iloc[idx]) # Convert to string
        content = str(self.data['content'].iloc[idx]) # Convert to string
        label = int(self.data['labels'].iloc[idx]) # Convert to integer

        input_text = title + " " + content

        encoding = self.tokenizer(input_text, truncation=True, padding='max_length', max_length=self.max_length, return_tensors='pt')

        return {
            'input_ids': encoding['input_ids'].flatten(),
            'attention_mask': encoding['attention_mask'].flatten(),
            'labels': torch.tensor(label, dtype=torch.long)
        }
```

Hình 25: Mô hình RoBERTa - Lớp tập dữ liệu tùy chỉnh

```
max_length = 512
```

Hình 26: Mô hình RoBERTa - Xác định độ dài chuỗi tối đa

```
dataset = NewsDataset(data, tokenizer, max_length)
train_size = int(0.8 * len(dataset))
test_size = len(dataset) - train_size
train_dataset, test_dataset = torch.utils.data.random_split(dataset, [train_size, test_size])

batch_size = 8

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Hình 27: Mô hình RoBERTa - Tạo tập dữ liệu và trình tải dữ liệu

```
# Fine-tune RoBERTa for sequence classification
model = XLMRobertaForSequenceClassification.from_pretrained("xlm-roberta-large", num_labels=2)
```

Hình 28: Mô hình RoBERTa - Tinh chỉnh RoBERTa để phân loại trình tự

```
# Define optimizer and learning rate scheduler
optimizer = torch.optim.AdamW(model.parameters(), lr=1e-5)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=1, gamma=0.1)
```

Hình 29: Mô hình RoBERTa - Xác định trình tối ưu hóa và lập lịch tốc độ học tập

```
epochs = 3
```

Hình 30: Mô hình RoBERTa - Xác định các thông số đào tạo

```
# Train the model
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model.to(device)
model.train()
for epoch in range(epochs):
    running_loss = 0.0
    for batch in tqdm(train_loader, desc=f"Epoch {epoch+1}/{epochs}"):
        print(batch, 'batch')
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        optimizer.zero_grad()

        outputs = model(input_ids, attention_mask=attention_mask, labels=labels)
        loss = outputs.loss
        logits = outputs.logits

        loss.backward()
        optimizer.step()

    avg_loss = running_loss / len(train_loader)
    print(f'Training Loss: {avg_loss:.4f}')
    # Adjust learning rate
    scheduler.step()
```

Hình 31: Mô hình RoBERTa - Train model

```
# Evaluate the model
model.eval()
y_true = []
y_pred = []

with torch.no_grad():
    for batch in tqdm(test_loader, desc="Evaluating"):
        input_ids = batch["input_ids"].to(device)
        attention_mask = batch["attention_mask"].to(device)
        labels = batch["labels"].to(device)

        outputs = model(input_ids, attention_mask=attention_mask)
        logits = outputs.logits
        predictions = torch.argmax(logits, dim=1)

        y_true.extend(labels.cpu().numpy())
        y_pred.extend(predictions.cpu().numpy())

model.save_pretrained('saved_model3')
```

Hình 32: Mô hình RoBERTa - Đánh giá mô hình

```
accuracy = accuracy_score(y_true, y_pred)
macro_f1 = f1_score(y_true, y_pred, average='macro')
micro_f1 = f1_score(y_true, y_pred, average='micro')
weighted_f1 = f1_score(y_true, y_pred, average='weighted')
```

Hình 33: Mô hình RoBERTa - Tính toán số liệu

```
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=["Unreliable", "Reliable"]))

# Print metrics
print(f"Validation Accuracy: {accuracy:.4f}")
print(f"Macro F1 Score: {macro_f1:.4f}")
print(f"Micro F1 Score: {micro_f1:.4f}")
print(f"Weighted F1 Score: {weighted_f1:.4f}")
```

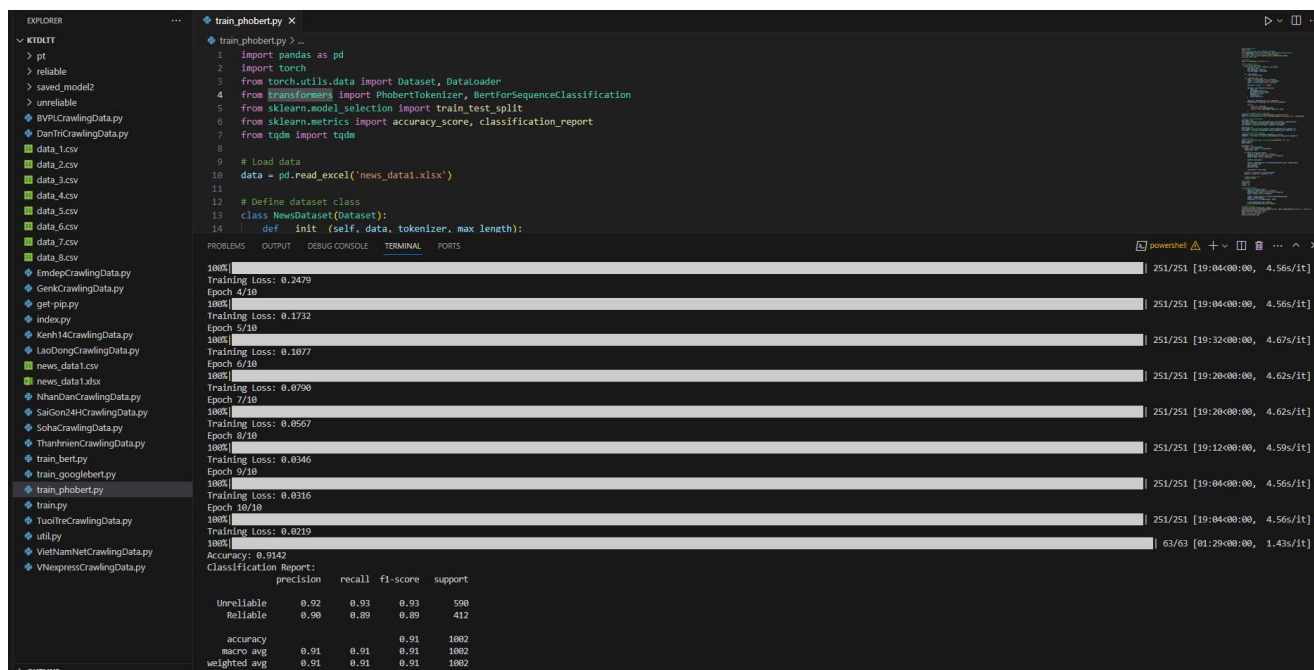
Hình 34: Mô hình RoBERTa - In báo cáo phân loại và số liệu in

```
mrkda@KieuDanh: /d/KTDLTT 2/KTDLTT
$ python train_bert.py
Some weights of BertForSequenceClassification were not initialized from the model checkpoint at bert-base-multilingual-cased and are newly initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Epoch 1/2
100% | 251/251 [11:43:47<00:00, 168.24s/it]
Training Loss: 0.3356
Epoch 2/2
100% | 251/251 [28:15:14<00:00, 405.24s/it]
Training Loss: 0.1742
Accuracy: 0.9391
Classification Report:
      precision    recall  f1-score   support

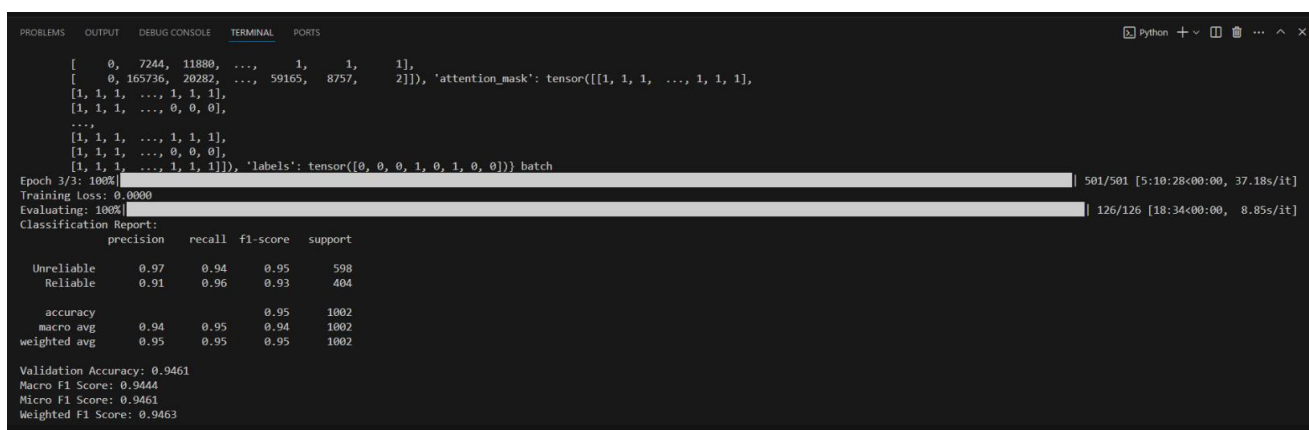
Unreliable      0.97      0.93      0.95         590
Reliable        0.90      0.96      0.93         412

   accuracy      0.94         1002
  macro avg      0.93      0.94      0.94         1002
 weighted avg      0.94      0.94      0.94         1002
```

Hình 35: Kết quả mô hình BERT



Hình 36: Kết quả mô hình PhoBERT



Hình 37: Kết quả mô hình RoBERTa

TÀI LIỆU THAM KHẢO

- [1] Tìm hiểu về BERT (mô hình ngôn ngữ). Link: [https://vi.wikipedia.org/wiki/BERT_\(m%C3%B4_h%C3%ACnh_ng%C3%B4n_ng%E1%BB%AF\)](https://vi.wikipedia.org/wiki/BERT_(m%C3%B4_h%C3%ACnh_ng%C3%B4n_ng%E1%BB%AF)) (Ngày truy cập 15/3/2024)
- [2] BERT. RoBERTa, PhoBERT, BERTweet: Ứng dụng state-of-the-art pre-trained model cho bài toán phân loại văn bản. Link: <https://viblo.asia/p/bert-roberta-phobert-bertweet-ung-dung-state-of-the-art-pre-trained-model-cho-bai-toan-phan-loai-van-ban-4P856PEWZY3> (Ngày truy cập 15/3/2024)
- [3] Hiểu mô hình BERT. Link: <https://ichi.pro/vi/hieu-mo-hinh-bert-176257345598377> (Ngày truy cập 12/3/2024)