BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh viên : Nguyễn Thành Long

Lớp : CT2101C

Giảng Viên Hướng Dẫn: Ths.Nguyễn Thị Xuân Hương

Hải Phòng – 2021

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

TÌM HIỂU VỀ MÔ HÌNH NGÔN NGỮ PHOBERT CHO BÀI TOÁN PHÂN LOẠI QUAN ĐIỂM BÌNH LUẬN TIẾNG VIỆT

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: CÔNG NGHỆ THÔNG TIN

Sinh Viên : Nguyễn Thành Long

Lớp : CT2101C

Giảng Viên Hướng Dẫn : Ths.Nguyễn Thị Xuân Hương

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Sinh viên: Nguyễn Thành Long **Mã SV** : 1712111008

Lớp : CT2101C

Ngành : Công nghệ thông tin

Tên đề tài: Tìm hiểu mô hình ngôn ngữ PhoBert cho bài toán phân loại quan điểm bình luận tiếng Việt

CÁN BỘ HƯỚNG DẪN ĐỀ TÀI TỐT NGHIỆP

Họ và tên : **Nguyễn Thị Xuân Hương**

Học hàm, học vị : Thạc sĩ

Cơ quan công tác : Trường Đại học Quản lý và Công nghệ Hải Phòng

Nội dung hướng dẫn:

+ Tìm hiểu về mô hình ngôn ngữ PhoBert.

+ Tìm hiểu về bài toán phân tích quan điểm người dùng, phân loại quan điểm bình luận Tiếng Việt.

+ Tìm hiểu về ngôn ngữ lập trình Python.

Đề tài tốt nghiệp được giao ngày 16 tháng 07 năm 2021

Yêu cầu phải hoàn thành xong trước ngày 03 tháng 10 năm 2021

Đã nhận nhiệm vụ ĐTTN	Đã giao nhiệm vụ ĐTTN
Sinh viên	*Giảng viên hướng dẫn*

Hải Phòng, ngày.....tháng.....năm 2021

TRƯỞNG KHOA

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN TỐT NGHIỆP

Họ và tên giảng viên: **Nguyễn Thị Xuân Hương**

Đơn vị công tác: Khoa Công nghệ thông tin, Trường Đại học Quản lý và Công nghệ Hải Phòng

Họ và tên sinh viên: **Nguyễn Thành Long** Ngành: Công nghệ thông tin

Nội dung hướng dẫn:

+ Tìm hiểu về mô hình ngôn ngữ PhoBert.

+ Tìm hiểu về bài toán phân tích quan điểm người dùng, phân loại quan điểm bình luận Tiếng Việt.

+ Tìm hiểu về ngôn ngữ lập trình Python.

1. Tinh thần thái độ của sinh viên trong quá trình làm đề tài tốt nghiệp.

……

……

……

……

2. Đánh giá chất lượng của đồ án/khóa luận (so với nội dung yêu cầu đã đề ra trong nhiệm vụ Đ.T. T.N trên các mặt lý luận, thực tiễn, tính toán số liệu…).

……...

……...

……...

……...

3. Ý kiến của giảng viên hướng dẫn tốt nghiệp.

Đạt ☐ Không đạt ☐ Điểm:……...

Hải Phòng, ngày.....tháng 10 năm 2021

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

PHIẾU NHẬN XÉT CỦA GIẢNG VIÊN CHẤM PHẢN BIỆN

Họ và tên giảng viên: **Đỗ Văn Chiểu**

Đơn vị công tác: Khoa Công nghệ thông tin, Trường Đại học Quản lý và Công nghệ Hải Phòng.

Họ và tên sinh viên: Nguyễn Thành Long Ngành: Công nghệ thông tin

Đề tài tốt nghiệp: Tìm hiểu mô hình ngôn ngữ PhoBert cho bài toán phân loại quan điểm bình luận Tiếng Việt

1. **Phần nhận xét của giảng viên chấm phản biện.**

.....
.....
.....
.....

2. **Những mặt còn hạn chế.**

.....
.....
.....
.....

3. Ý kiến của giảng viên chấm phản biện.

Được bảo vệ ☐ Không được bảo vệ ☐ Điểm:…….

Hải Phòng, ngày.....tháng 10 năm 2021

Giảng viên chấm phản biện

(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN

Lời đầu tiên cho em gửi lời cảm ơn sâu sắc đến gia đình, người thân của em đã động viên, giúp đỡ, cổ vũ, tạo cho em thêm động lực để em có thể hoàn thành đồ án trong thời gian được giao.

Em xin gửi lời cảm ơn đến Ban Giám Hiệu Trường Đại học Quản lý và Công nghệ Hải Phòng, các Ban, Ngành đã hỗ trợ hết mức tạo điều kiện tốt nhất để em có thể đăng kí đồ án tốt nghiệp.

Em xin cảm ơn đến các thầy, các cô Khoa Công nghệ thông tin, Trường Đại học Quản lý và Công nghệ Hải Phòng, đã giúp em có những kiến thức cực kì bổ ích trong vòng 4 năm vừa qua, giúp em có được nền tảng kiến thức vững chắc để em có thẻ thực hiện được đồ án.

Em xin gửi lời cảm ơn chân thành đến cô Ths. Nguyễn Thị Xuân Hương, đã dành rất nhiều thời gian công sức, cả về vật chất và tinh thần giúp em có thể thể hoàn thành được đồ án một cách trơn tru nhất.

Em xin chân thành cảm ơn!

Hải Phòng, ngày.....tháng.....năm 2021

Sinh viên

Nguyễn Thành Long

MỞ ĐẦU

Trong bất kỳ xã hội nào con người luôn có nhu cầu được giao tiếp và thể hiện, hình thức được sử dụng phổ biến đó là diễn đạt bằng ngôn ngữ. Ngôn ngữ sử dụng từ ngữ hoặc dấu hiệu để diễn tả được thể hiện qua lời nói, chữ viết hoặc các hình ảnh. Với sự bùng nổ của Internet và các trang mạng xã hội, các trang web tài liệu, sách báo, các trang sản phẩm, email,.. một lượng lớn dữ liệu văn bản của ngôn ngữ được tạo ra mỗi ngày. Để giúp máy tính hiểu được những dữ liệu này là công việc quan trọng để hỗ trợ hoặc quyết định dựa trên ngôn ngữ.

Xử lý ngôn ngữ tự nhiên nghiên cứu sự tương tác bằng ngôn ngữ tự nhiên giữa máy tính và con người. Trong thực tế, việc sử dụng các kỹ thuật xử lý ngôn ngữ tự nhiên để xử lý và phân tích dữ liệu văn bản (ngôn ngữ tự nhiên của con người) rất phổ biến, chẳng hạn như các mô hình ngôn ngữ trong hay các mô hình dịch máy. Để có thể xây dựng các phương pháp xử lý ngôn ngữ thì trước tiên chúng ta cần quan tâm đến việc biểu diễn ngôn ngữ tự nhiên như thế nào. Một số phương pháp biểu diễn ngôn ngữ đã được giới thiệu được sử dụng trong các nhiệm vụ xử lý ngôn ngữ tự nhiên như: sự xuất hiện (Presence) và tần suất xuất hiện (Frequency), mô hình ngôn ngữ (n-gram), thông tin nhãn từ loại (Parts of Speech), thông tin phân tích ngữ pháp (Syntactic parsing), biểu diễn véc tơ từ (Word2Vec), nhúng ký tự (Character Embedding), mạng ngữ nghĩa (WordNet), mạng từ điển quan điểm (SentiWordNet), v.v. Các phương pháp biểu diễn ngôn ngữ này giúp trích xuất các đặc trưng từ ngôn ngữ sử dụng cho các mô hình xử lý ngôn ngữ tự nhiên giúp nâng cao hiệu quả cho các phương pháp phân tích. Do đó, nghiên cứu về các phương pháp biểu diễn ngôn ngữ nhằm tìm ra các đặc trưng hữu ích cho bài toán NLP là nhiệm vụ quan trọng.

Gần đây, Google AI 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. Với sự phát triển của các trang mạng xã hội và các trang đánh giá sản phẩm, dữ liệu bình luận khen chê của khách hàng đang gia tăng một cách nhanh chóng tạo thành kho dữ liệu đánh giá khổng lồ. Việc hiểu xem khách hàng đánh giá về một sản phẩm, dịch vụ hay vấn đề được quan tâm là tích cực hay tiêu cực là nhiệm vụ được các nhà nghiên cứu quan tâm trong những thập niên gần đây và đã có nhiều ứng dụng trong thực tế. Chính vì những lý do đó, em chọn đề tài " Tìm hiểu mô hình PhoBert cho bài toán phân loại quan

điểm bình luận Tiếng Việt ”nhằm tìm hiểu các phương pháp mới biểu diễn cho ngôn ngữ tiếng Việt và áp dụng nó cho bài toán phân loại bình luận tiếng Việt. Đồ án thiết kế gồm 3 chương: Chương 1 Mô hình BERT trình bày về mô hình BERT và các khái niệm liên quan, chương 2: Mô hình PhoBERT trình bày về các tìm hiểu cho mô hình PhoBERT, Chương 3: Ứng dụng PhoBERT cho bài toán phân loại bình luận tiếng Việt trong đó trình bày về bài toán, công cụ sử dụng và các cài đặt thử nghiệm, cuối cùng là phần kết luận.

DANH MỤC CÁC HÌNH VẼ VÀ CÁC BẢNG

BẢNG CÁC TỪ VIẾT TẮT

Viết tắt	Đầy đủ	Ý nghĩa
BERT	Bidirectional Encoder Representations from Transformers	Mô hình ngôn ngữ
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
NSP	Next Sentence Prediction	Dữ báo câu tiếp theo
NER	Name Entity Recognition	Nhận diện thực thể trong câu
NLI	Natural Languague Inference	Suy luận ngôn ngữ tự nhiên
SQuAD	Stanford Question Answering Dataset	Tác vụ hỏi đáp
SOTA	State-Of-Art	Hiện đại nhất
GLUE	General Language Understanding Evaluation	Điểm khái quát đánh giá mức độ hiểu ngôn ngữ
MLM	Masked Language Model	Mô hình ngôn ngữ Masked
RNN	Recurrent Neural Network	Mạng neural hồi quy
ELMo	Embeddings from Language Model	Nhúng từ Mô hình Ngôn ngữ

CHƯƠNG 1. MÔ HÌNH BERT

1.1. Khái niệm

BERT (Bidirectional Encoder Representations from Transformers) là một mô hình ngôn ngữ (Language Model) được tạo ra bởi Google AI và được giới thiệu vào năm 2008. BERT được coi như là đột phá lớn trong Machine Learning bởi vì khả năng ứng dụng của nó vào nhiều bài toán NLP (Natural Language Processing) khác nhau: Question Answering, Natural Language Inference,... với kết quả rất tốt.

Các nhà nghiên cứu làm việc tại Google AI tái khẳng định, sự thiếu hụt dữ liệu huấn luyện là một trong những thách thức lớn nhất trong lĩnh vực xử lý ngôn ngữ tự nhiên. Đây là một lĩnh vực rộng lớn và đa dạng với nhiều nhiệm vụ riêng biệt, hầu hết các tập dữ liệu đều chỉ đặc thù cho từng nhiệm vụ. Để thực hiện được tốt những nhiệm vụ này ta cần những bộ dữ liệu lớn chứa hàng triệu thậm chí hàng tỷ ví dụ mẫu. Tuy nhiên, trong thực tế hầu hết các tập dữ liệu hiện giờ chỉ chứa vài nghìn hoặc vài trăm nghìn mẫu được đánh nhãn bằng tay bởi con người (các chuyên gia ngôn ngữ học). Sự thiếu hụt dữ liệu có nhãn chất lượng cao để huấn luyện mô hình gây cản trở lớn cho sự phát triển của NLP nói chung.

Để giải quyết thách thức này, các mô hình xử lý ngôn ngữ tự nhiên sử dụng một cơ chế tiền xử lý dữ liệu huấn luyện bằng việc transfer từ một mô hình chung được huấn luyện từ một lượng lớn các dữ liệu không được gán nhãn. Ví dụ một số mô hình đã được nghiên cứu trước đây để thực hiện nhiệm vụ này như Word2vec, Glove hay FastText.

Việc nghiên cứu các mô hình này sẽ giúp thu hẹp khoảng cách giữa các tập dữ liệu chuyên biệt cho huấn luyện bằng việc xây dựng mô hình tìm ra đại diện chung của ngôn ngữ sử dụng một số lượng lớn các văn bản chưa được gán nhãn lấy từ các trang web.

Các mô hình được huấn luyện trước khi được tinh chỉnh lại trên các nhiệm vụ khác nhau với các bộ dữ liệu nhỏ như Question Answering, Sentiment Analysis,...sẽ dẫn đến sự cải thiện đáng kể về độ chính xác cho so với các mô hình được huấn luyện trước với các bộ dữ liệu này.

Tuy nhiên, các mô hình kể trên có những yếu điểm riêng của nó, đặc biệt là không thể hiện được sự đại diện theo ngữ cảnh cụ thể của từ trong từng lĩnh vực hay văn cảnh cụ thể.

Tiếp nối sự thành công nhất định của các mô hình trước đó, Google đã công bố thêm 1 kỹ thuật mới được gọi là Bidirectional Encoder Representations from Transformers (BERT).

1.2. Tại sao lại cần BERT

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.

Ví dụ : Trong OpenAI GPT, các tác giả sử dụng đã kiến trúc left-to-right, nghĩa là các từ chỉ phụ thuộc vào các từ ở trước đó.

Nhưng có một nghịch lý là các mô hình Deep Learning cần lượng dữ liệu rất lớn - lên tới hàng triệu - để có thể cho ra kết quả tốt. 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. Với Transfer Learning, các mô hình (model) "chung" nhất với tập dữ liệu khổng lồ trên internet (pre-training) được xây dựng và có thể được "tinh chỉnh" (fine-tune) cho các bài toán cụ thể.

Nhờ có kỹ thuật này mà kết quả cho các bài toán được cải thiện rõ rệt, không chỉ trong xử lý ngôn ngữ tự nhiên mà còn trong các lĩnh vực khác như Computer Vision,... BERT là một trong những đại diện ưu tú nhất trong Transfer Learning cho xử lý ngôn ngữ tự nhiên, 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.

1.3. Một số khái niệm

1.3.1. Nhiệm vụ phía sau (Downstream task)

Là những nhiệm vụ học hỏi được giám sát được cải thiện dựa trên những mô hình được huấn luyện trước.

Ví dụ: Chúng ta sử dụng lại các biểu diễn từ học được từ những mô hình được huấn luyện trước trên bộ văn bản lớn vào một nhiệm vụ phân tích cảm xúc huấn luyện trên bộ văn bản có kích thước nhỏ hơn. Áp dụng nhúng huấn luyện trước (pretrain-embedding)

đã giúp cải thiện mô hình. Như vậy nhiệm vụ sử dụng nhúng huấn luyện trước được gọi là nhiệm vụ sau.

1.3.2. Điểm khái quát đánh giá mức độ hiểu ngôn ngữ (GLUE score benchmark)

GLUE score benchmark là một tập hợp các chỉ số được xây dựng để đánh giá khái quát mức độ hiểu ngôn ngữ của các mô hình NLP.

Các đánh giá được thực hiện trên các bộ dữ liệu tiêu chuẩn được qui định tại các convention về phát triển và thúc đẩy NLP. Mỗi bộ dữ liệu tương ứng với một loại tác NLP vụ như:

- Phân tích tình cảm (Sentiment Analysis)
- Hỏi đáp (Question and Answering)
- Suy luận ngôn ngữ tự nhiên (NLI - Natural Languague Inference)
- Dự báo câu tiếp theo (NSP - Next Sentence Prediction)
- Nhận diện thực thể trong câu (NER - Name Entity Recognition)

1.3.3. Phân tích cảm xúc (Sentiment Analysis)

Phân loại cảm xúc văn bản thành 2 nhãn tích cực (positive) và tiêu cực (negative). Thường được sử dụng trong các hệ thống đánh giá bình luận của người dùng.

1.3.4. Hỏi đáp (Question and Answering)

Là thuật toán hỏi và đáp. Đầu vào là một cặp câu (pair sequence) bao gồm: câu hỏi (question) có chức năng hỏi và đoạn văn bản (paragraph) chứa thông tin trả lời cho câu hỏi. Một bộ dữ liệu chuẩn nằm trong GLUE dataset được sử dụng để đánh giá nhiệm vụ hỏi và đáp là SQuAD - Stanford Question Answering Dataset.

1.3.5. Suy luận ngôn ngữ (Natural Language Inference)

Là các nhiệm vụ suy luận ngôn ngữ đánh giá mối quan hệ giữa các cặp câu, cũng tương tự như Textual Entailment.

1.3.6. Quan hệ văn bản (Textual Entailment)

Là nhiệm vụ đánh giá mối quan hệ định hướng giữa 2 văn bản. Nhãn đầu ra của các cặp câu được chia thành đối lập (contradiction), trung lập (neutral) hay có quan hệ đi kèm (textual entailment).

Ví dụ, chúng ta có các câu:

- A: Hôm nay trời mưa.
- B: Tôi mang ô tới trường.
- C: Hôm nay trời không mưa.
- D: Hôm nay là thứ 3.

Khi đó (A, B) có mối quan hệ đi kèm. Các cặp câu (A, C) có mối quan hệ đối lập và (A, D) là trung lập.

1.3.7. Ngữ cảnh (Contextual)

Là ngữ cảnh của từ. Một từ được định nghĩa bởi một cách phát âm nhưng khi được đặt trong những câu khác nhau thì có thể mang ngữ nghĩa khác nhau. ngữ cảnh có thể coi là môi trường xung quanh từ để góp phần định nghĩa từ.

Ví dụ:

- Câu A: Tôi đồng ý với ý kiến của anh.
- Câu B: Lão Hạc phải kiếm từng đồng để nuôi cậu Vàng.

Thì từ " đồng " trong câu A và B có ý nghĩa khác nhau. Chúng ta biết điều này vì dựa vào ngữ cảnh của từ.

1.3.8. Phương pháp Hiện đại nhất (SOTA)

Viết tắt của state-of-art là những phương pháp, kỹ thuật tốt nhất mang lại hiệu quả cao nhất từ trước đến nay.

Mô hình biểu diễn mã hóa 2 chiều dựa trên biến đổi (BERT-Bidirectional Encoder Representation from Transformer)

Mô hình BERT. Đây là lớp mô hình SOTA trong nhiều nhiệm vụ của GLUE score benchmark.

1.3.9. Mô hình LTR

Là mô hình học bối cảnh theo một chiều duy nhất từ trái sang phải. Chẳng hạn như lớp các model RNN.

1.3.10. Mô hình ngôn ngữ được đánh dấu MLM (Masked Language Model)

Là mô hình mà bối cảnh của từ được học từ cả 2 phía bên trái và bên phải cùng một lúc từ những bộ dữ liệu không có giám sát.

Dữ liệu vào sẽ được đánh dấu (tức thay bằng một mã đánh dấu (token MASK)) một cách ngẫu nhiên với tỷ lệ thấp. Huấn luyện mô hình dự báo từ mã được đánh dấu dựa trên bối cảnh xung quanh là những từ không được đánh dấu nhằm tìm ra biểu diễn của từ.

1.4. Ngữ cảnh (Contextual) và vai trò trong NLP

Bản chất của ngôn ngữ là âm thanh được phát ra để diễn giải dòng suy nghĩ của con người. Trong giao tiếp, các từ thường không đứng độc lập mà chúng sẽ đi kèm với các từ khác để liên kết mạch lạc thành một câu. Hiệu quả biểu thị nội dung và truyền đạt ý nghĩa sẽ lớn hơn so với từng từ đứng độc lập.

Ngữ cảnh trong câu có một sự ảnh hưởng rất lớn trong việc giải thích ý nghĩa của từ. Dựa trên đó, các thuật toán xử lý ngôn ngữ tự nhiên tốt nhất đều cố gắng đưa ngữ cảnh vào mô hình nhằm tạo ra sự đột phá và cải tiến. Trong đó mô hình BERT cũng sử dụng tiếp cận này.

Phân cấp mức độ phát triển của các phương pháp nhúng từ trong NLP có thể bao gồm các nhóm:

- **Không bối cảnh (Non-context)** Là các thuật toán không tồn tại bối cảnh trong biểu diễn từ. Đó là các thuật như " WORD2VEC, GLOVE, FASTTEXT ". Chúng ta chỉ có duy nhất một biểu diễn véc tơ cho mỗi một từ mà không thay đổi theo bối cảnh.

  Ví dụ :

  - 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

Thì từ đồng sẽ mang 2 ý nghĩa khác nhau nên phải có hai biểu diễn từ riêng biệt. Các thuật toán không có bối cảnh đã không đáp ứng được sự đa dạng về ngữ nghĩa của từ trong NLP.

- **Một chiều (Uni-directional)**: Là các thuật toán đã bắt đầu xuất hiện bối cảnh của từ. Các phương pháp nhúng từ dựa trên RNN là những phương pháp nhúng từ một chiều. Các kết quả biểu diễn từ đã có bối cảnh nhưng chỉ được giải thích bởi một chiều từ trái qua phải hoặc từ phải qua trái.

  Ví dụ:

- Câu C: Hôm nay tôi mang 200 tỷ “ gửi ” ở ngân hàng.
- Câu D: Hôm nay tôi mang 200 tỷ “ gửi ” ….

Như vậy véc tơ biểu diễn của từ gửi được xác định thông qua các từ liền trước với nó. Nếu chỉ dựa vào các từ liền trước “ Hôm nay tôi mang 200 tỷ ” thì ta có thể nghĩ từ phù hợp ở vị trí hiện tại là cho vay, mua, thanh toán,....

Ví dụ đơn giản trên đã cho thấy các thuật toán biểu diễn từ có bối cảnh tuân theo theo một chiều sẽ gặp hạn chế lớn trong biểu diễn từ hơn so với biểu diễn 2 chiều.

ELMo là một ví dụ cho phương pháp một chiều. Mặc dù phương pháp ELMO có kiến trúc dựa trên một mạng BiLSTM xem xét bối cảnh theo hai chiều từ trái sang phải và từ phải sang trái nhưng những chiều này là độc lập nhau nên ta coi như đó là biểu diễn một chiều. Thuật toán ELMO đã cải tiến hơn so với WORD2VEC và FASTTEXT đó là tạo ra nghĩa của từ theo bối cảnh. Trong ví dụ về từ “đồng” thì ở mỗi câu A và B chúng ta sẽ có một biểu diễn từ khác biệt.

- **Hai chiều (Bi-directional)**: Ngữ nghĩa của một từ không chỉ được biểu diễn bởi những từ liền trước mà còn được giải thích bởi toàn bộ các từ xung quanh. Luồng giải thích tuân theo **đồng thời** từ trái qua phải và từ phải qua trái **cùng một lúc**. Đại diện cho các phép biểu diễn từ này là những mô hình sử dụng kỹ thuật transformer. Gần đây, những thuật toán NLP theo trường phái hai chiều như BERT, ULMT, OpenAI GPT đã đạt được những kết quả SOTA trên hầu hết các nhiệm vụ của GLUE benchmark.

1.5. Tiếp cận nông và học sâu trong ứng dụng huấn luyện trước (pre-training) trong NLP

1.5.1. Tiếp cận nông (shallow approach)

- Imagenet trong Computer Vision

Trong xử lý ảnh, chúng ta đều biết tới những mô hình được huấn luyện trước (pretrained models) nổi tiếng trên bộ dữ liệu Imagenet với 1000 classes. Nhờ số lượng classes lớn nên hầu hết các nhãn trong phân loại ảnh thông thường đều xuất hiện trong Imagenet và chúng ta có thể học chuyển giao lại các nhiệm vụ xử lý ảnh rất nhanh và tiện lợi. Chúng ta cũng kỳ vọng NLP có một tợp hợp các mô hình được huấn luyện trước như

vậy, tri thức từ mô hình được huấn luyện trên các nguồn tài nguyên văn bản không nhãn (unlabeled text) rất dồi dào và sẵn có.

- Khó khăn học chuyển giao trong NLP

Tuy nhiên trong NLP việc học chuyển giao là không hề đơn giản như Computer Vision. Các kiến trúc mạng học sâu CNN của Computer Vision cho phép học chuyển giao trên đồng thời cả các đặc trưng ở mức độ thấp (low-level) và mức độ cao (high-level) thông qua việc tận dụng lại các tham số từ những tầng của mô hình được huấn luyện trước.

Nhưng trong NLP, các thuật toán cũ hơn như GLOVE, WORD2VEC, FASTTEXT chỉ cho phép sử dụng các biểu diễn véc tơ nhúng của từ là các đặc trưng ở mức độ thấp như là đầu vào cho tầng đầu tiên của mô hình. Các tầng còn lại giúp tạo ra đặc trưng ở mức độ cao lại được huấn luyện lại từ đầu.

Như vậy chúng ta chỉ chuyển giao được các đặc trưng ở mức độ rất nông nên phương pháp này còn được gọi là tiếp cận nông. Việc tiếp cận với các tầng sâu hơn là không thể. Điều này tạo ra một hạn chế rất lớn đối với NLP so với Computer Vision trong việc học chuyển giao. Cách tiếp cận nông trong học chuyển giao còn được xem như là dựa trên đặc trưng (**feature-based)**.

Khi áp dụng dựa trên đặc trưng, chúng ta sẽ tận dụng lại các biểu diễn từ được huấn luyện trước trên những kiến trúc mô hình cố định và những bộ văn bản có kích thước **rất lớn** để nâng cao khả năng biểu diễn từ trong không gian đa chiều. Một số đặc trưng được huấn luyện trước có thể áp dụng trong tiếng anh đã được huấn luyện sẵn đó là GLOVE, WORD2VEC, FASTTEXT, ELMO.

1.5.2. Học sâu (deep-learning)

Các mô hình NLP đột phá trong hai năm trở lại đây như BERT ELMO, ULMFIT, OpenAI GPT đã cho phép việc chuyển giao các tầng trong NLP khả thi hơn.

Chúng ta không chỉ học chuyển giao được các đặc trưng mà còn chuyển giao được kiến trúc của mô hình nhờ số lượng các tầng nhiều hơn, chiều sâu của mô hình sâu hơn trước đó.

Các kiến trúc mới phân cấp theo tầng có khả năng chuyển giao được những cấp độ khác nhau của đặc trưng từ mức thấp tới mức cao. Trong khi học nông chỉ chuyển giao được mức thấp tại tầng đầu tiên. Tất nhiên mức thấp cũng đóng vai trò quan trọng trong

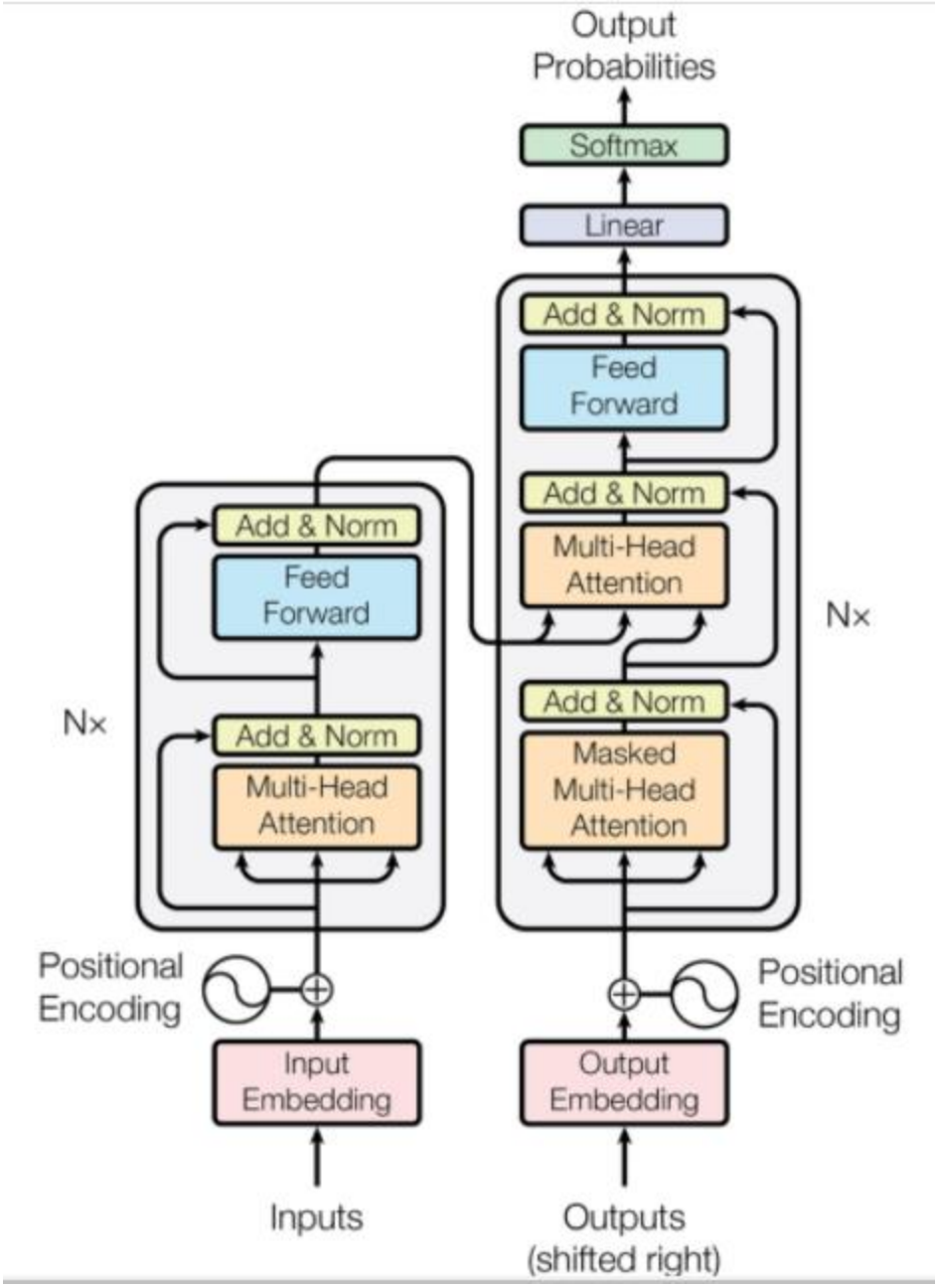
các nhiệm vụ NLP. Nhưng ở mức cao là những đặc trưng có ý nghĩa hơn vì đó là những đặc trưng đã được tinh luyện.

Người ta kỳ vọng rằng ULMFIT, OpenAI GPT, BERT sẽ là những mô hình được huấn luyện trước giúp tiến gần hơn tới việc xây dựng một lớp các mô hình được huấn luyện trước như ImageNet cho NLP. Khi học chuyển giao theo phương pháp học sâu chúng ta sẽ tận dụng lại kiến trúc từ mô hình được huấn luyện trước và bổ sung một số tầng phía sau để phù hợp với nhiệm vụ huấn luyện. Các tham số của các các tầng gốc sẽ được tinh chỉnh (**fine-tunning)** lại. Chỉ một số ít các tham số ở các tầng bổ sung được huấn luyện lại từ đầu.

1.6. Phương pháp TRANSFORMER

1.6.1. Encoder và Decoder trong BERT

Trước khi hiểu về BERT chúng ta cùng tìm hiểu về kỹ thuật transformer. Đây là một lớp mô hình SEQ2SEQ gồm 2 pha mã hóa (Encoder) và giải mã (Decoder). Mô hình hoàn toàn không sử dụng các kiến trúc mạng hồi quy của RNN mà chỉ sử dụng các các tầng chú ý (attention) để nhúng các từ trong câu. Kiến trúc cụ thể của mô hình như sau:



Hình 1. Sơ đồ kiến trúc transformer kết hợp với chú ý

Mô hình sẽ bao gồm 2 pha:

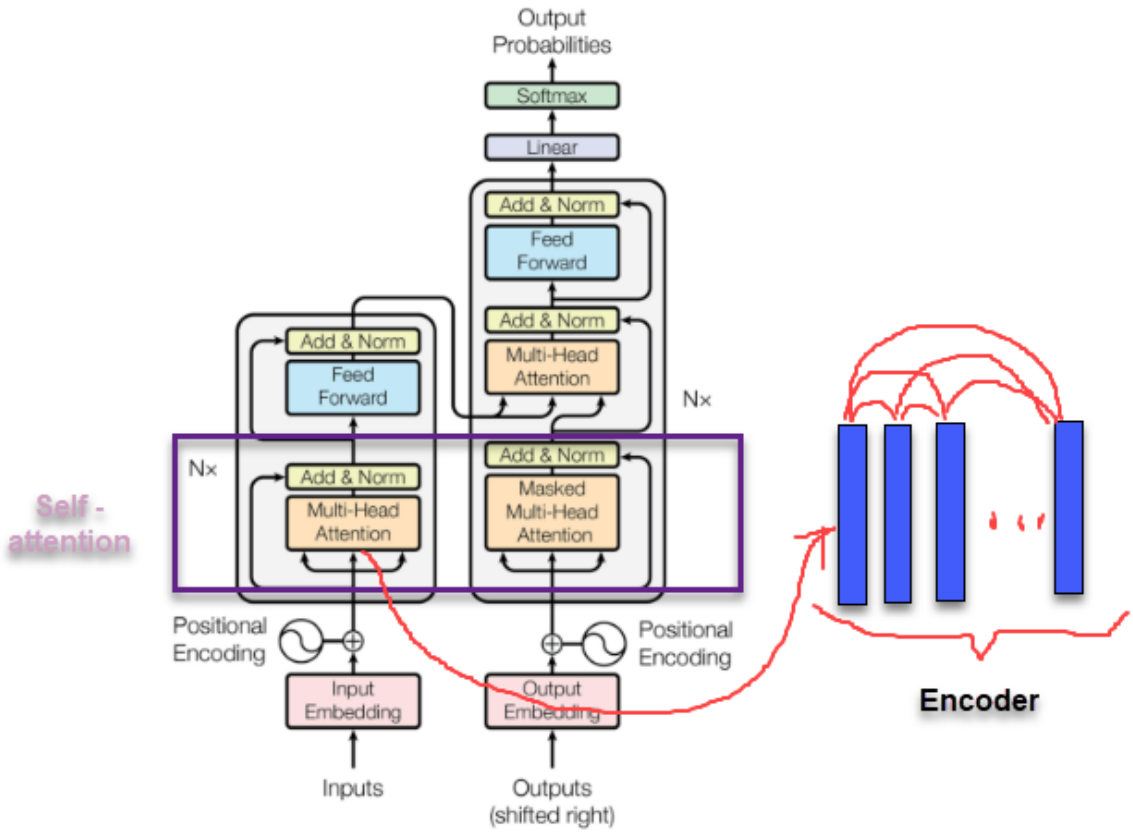
- Encoder: Bao gồm 6 tầng liên tiếp nhau. Mỗi một tầng sẽ bao gồm một tầng con (sub-layer) là Multi-Head Attention kết hợp với tầng kết nối đầy đủ (fully-connected layer) như mô tả ở nhánh encoder bên trái của hình vẽ. Kết thúc quá trình encoder ta thu được một véc tơ đầu ra nhúng cho mỗi từ.

- Decoder: Kiến trúc cũng bao gồm các tầng liên tiếp nhau. Mỗi một tầng của Decoder cũng có các tầng con gần tương tự như tầng của Encoder nhưng bổ sung thêm tầng con đầu tiên là Masked Multi-Head Attention có tác dụng loại bỏ các từ trong tương lai khỏi quá trình chú ý (attention).

1.6.2. Các tiến trình self-attention và encoder-decoder attention (phương pháp transformer)

Trong kiến trúc transformer chúng ta sẽ áp dụng 2 dạng chú ý khác nhau tại từng bước huấn luyện.

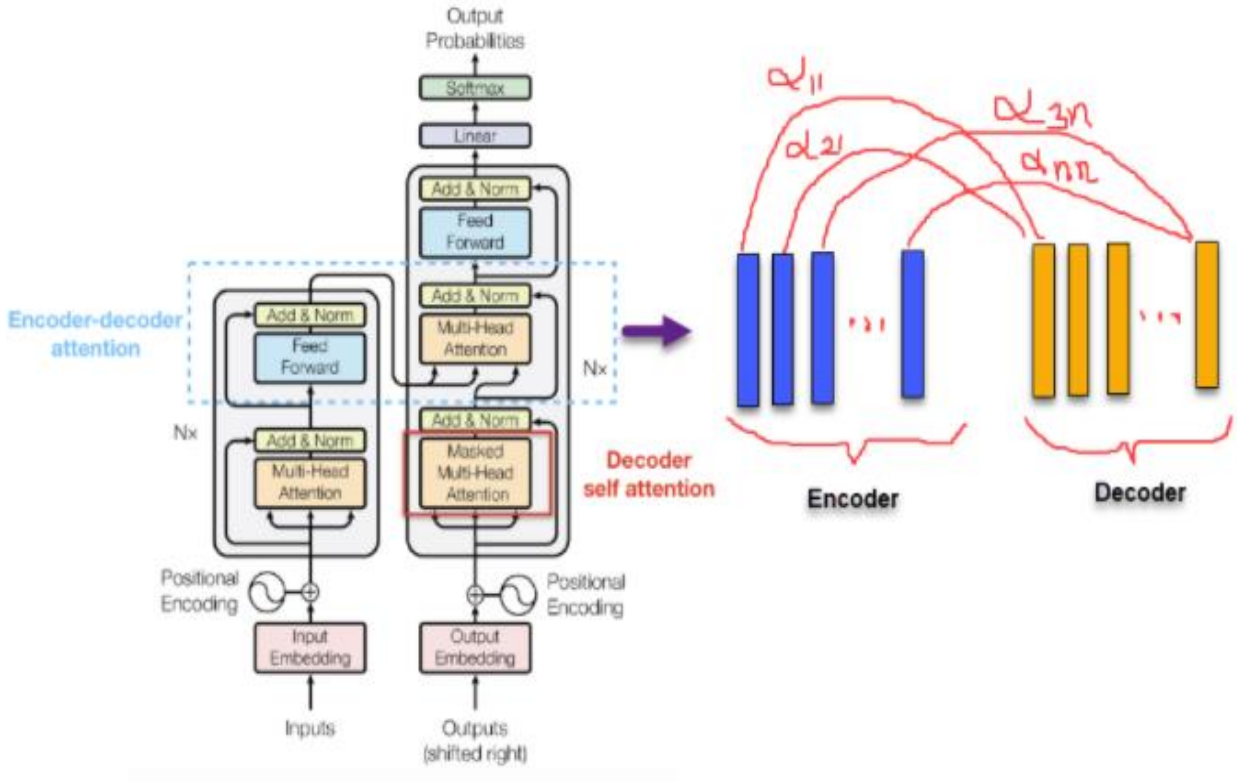
- **self-attention**: Được sử dụng trong cùng một câu đầu vào, tại encoder hoặc tại decoder. Đây chính là chú ý được áp dụng tại các Multi-Head Attention ở đầu vào của cả 2 pha encoder và decoder.



Hình 2. Sơ đồ vị trí áp dụng self-attention trong kiến trúc transformer.

Các véc tơ nhúng của cùng một chuỗi encoder hoặc decoder tự liên kết với nhau để tính toán chú ý như hình bên phải.

- **Chú ý mã hóa và giải mã (Encoder-decoder attention):**



Hình 3. Sơ đồ chú ý tương tác giữa các véc tơ nhúng của encoder và decoder

Bên trái là vị trí áp dụng chú ý mã hóa và giải mã. Bên phải là cách tính trọng số chú ý khi kết hợp mỗi véc tơ nhúng ở decoder với toàn bộ các véc tơ nhúng ở encoder.

Chú ý mã hóa và giải mã là kiến trúc chú ý tương tác giữa các véc tơ nhúng của encoder và decoder. Véc tơ ngữ cảnh được tính toán trên encoder đã được tính tương quan với véc tơ decoder nên sẽ có ý nghĩa giải thích bối cảnh của từ tại vị trí bước thời gian giải mã (time step decoder) tương ứng. Sau khi kết hợp giữa véc tơ ngữ cảnh và véc tơ decoder ta sẽ thực hiện tiếp qua một tầng kết nối đầy đủ (fully connected layer) để tính phân phối xác suất cho đầu ra.

Mặc dù có kiến trúc chỉ gồm các biến đổi chú ý nhưng Transformer lại có kết quả rất tốt trong các nhiệm vụ NLP như phân tích tình cảm và dịch máy.

1.7. Mô hình BERT

BERT là viết tắt của cụm từ Bidirectional Encoder Representation from Transformer có nghĩa là mô hình biểu diễn từ theo 2 chiều ứng dụng kỹ thuật Transformer. BERT được thiết kế để huấn luyện trước các từ nhúng (pre-train word embedding). Điểm đặc biệt ở BERT đó là nó có thể điều hòa cân bằng bối cảnh theo cả 2 chiều trái và phải.

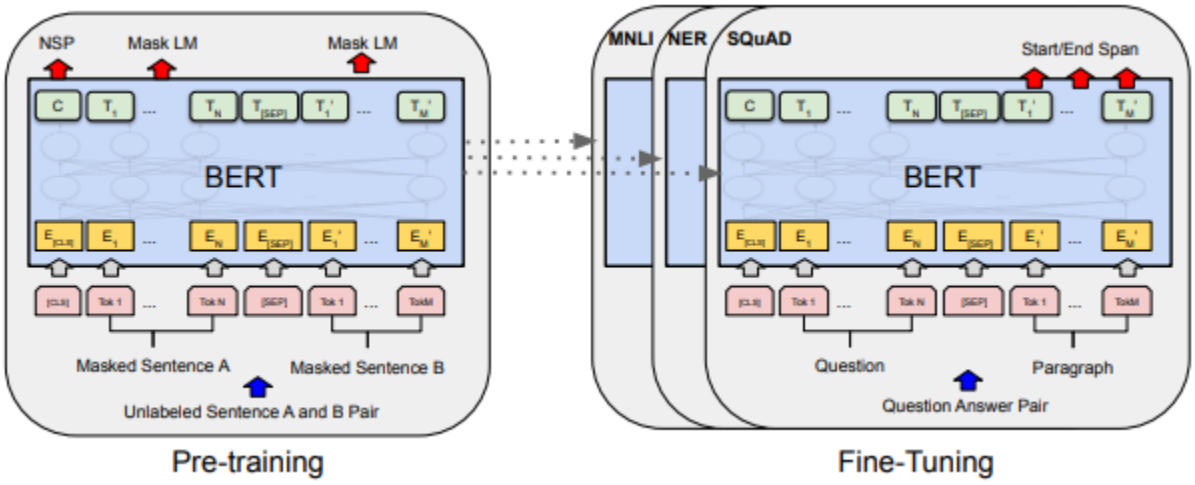
Cơ chế chú ý của Transformer sẽ truyền toàn bộ các từ trong câu văn đồng thời vào mô hình một lúc mà không cần quan tâm đến chiều của câu.

Do đó Transformer được xem như là huấn luyện hai chiều (bidirectional) mặc dù trên thực tế chính xác hơn chúng ta có thể nói rằng đó là huấn luyện không chiều (non-directional).

Đặc điểm này cho phép mô hình học được bối cảnh của từ dựa trên toàn bộ các từ xung quanh nó bao gồm cả từ bên trái và từ bên phải.

1.7.1. Mô hình BERT tinh chỉnh (Fine-tuning model BERT)

Một điểm đặc biệt ở BERT mà các mô hình nhúng trước đây chưa từng có đó là kết quả huấn luyện có thể tinh chỉnh được. Chúng ta sẽ thêm vào kiến trúc mô hình một tầng đầu ra để tùy biến theo nhiệm vụ huấn luyện.



Hình 4. Toàn bộ tiến trình pre-training và fine-tuning của BERT

Một kiến trúc tương tự được sử dụng cho cả mô hình huấn luyện trước và mô hình tinh chỉnh. Chúng ta sử dụng cùng một tham số huấn luyện trước để khởi tạo mô hình cho các nhiệm vụ sau khác nhau.

Trong suốt quá trình tinh chỉnh thì toàn bộ các tham số của các tầng học chuyển giao sẽ được điều chỉnh. Đối với các nhiệm vụ sử dụng đầu váo là một cặp chuỗi (pair-sequence) ví dụ như câu hỏi và trả lời thì ta sẽ thêm mã khởi tạo là [CLS] ở đầu câu, mã [SEP] ở giữa để ngăn cách 2 câu.

Tiến trình áp dụng tinh chỉnh sẽ như sau:

- Bước 1: Nhúng toàn bộ các mã của cặp câu bằng các véc tơ nhúng từ mô hình huấn luyện trước. Các mã nhúng bao gồm cả 2 mã là [CLS] và [SEP] để đánh dấu vị trí bắt đầu của câu hỏi và vị trí ngăn cách giữa 2 câu. Hai mã này sẽ được dự báo ở đầu ra để xác định các phần mở rộng bắt đầu/kết thúc (Start/End Spand) của câu đầu ra.
- Bước 2: Các véc tơ nhúng sau đó sẽ được truyền vào kiến trúc chú ý nhiều đầu vào (multi-head attention) với nhiều mã khối (block code) (thường là 6, 12 hoặc 24 khối tùy theo kiến trúc BERT). Ta thu được một véc tơ đầu ra ở encoder.
- Bước 3: Để dự báo phân phối xác suất cho từng vị trí từ ở decoder, ở mỗi bước thời gian chúng ta sẽ truyền vào decoder véc tơ đầu ra của encoder và véc tơ nhúng đầu vào của decoder để tính chú ý mã hóa và giải mã. Sau đó ánh xạ qua tầng tuyến tính (liner layer) và hàm softmax để thu được phân phối xác suất cho đầu ra tương ứng ở bước thời gian t.
- Bước 4: Trong kết quả trả ra ở đầu ra của transformer ta sẽ cố định kết quả của câu hỏi sao cho trùng với câu hỏi ở đầu vào. Các vị trí còn lại sẽ là thành phần mở rộng bắt đầu/kết thúc tương ứng với câu trả lời tìm được từ câu đầu váo.

Quá trình huấn luyện chúng ta sẽ tinh chỉnh lại toàn bộ các tham số của mô hình BERT đã loại bỏ tầng tuyến tính ở đỉnh (cut off top linear layer) và huấn luyện lại từ đầu các tham số của tầng tuyến tính mà chúng ta thêm vào kiến trúc mô hình BERT để tùy chỉnh lại phù hợp với bài toán.

1.8. Cách huấn luyện BERT

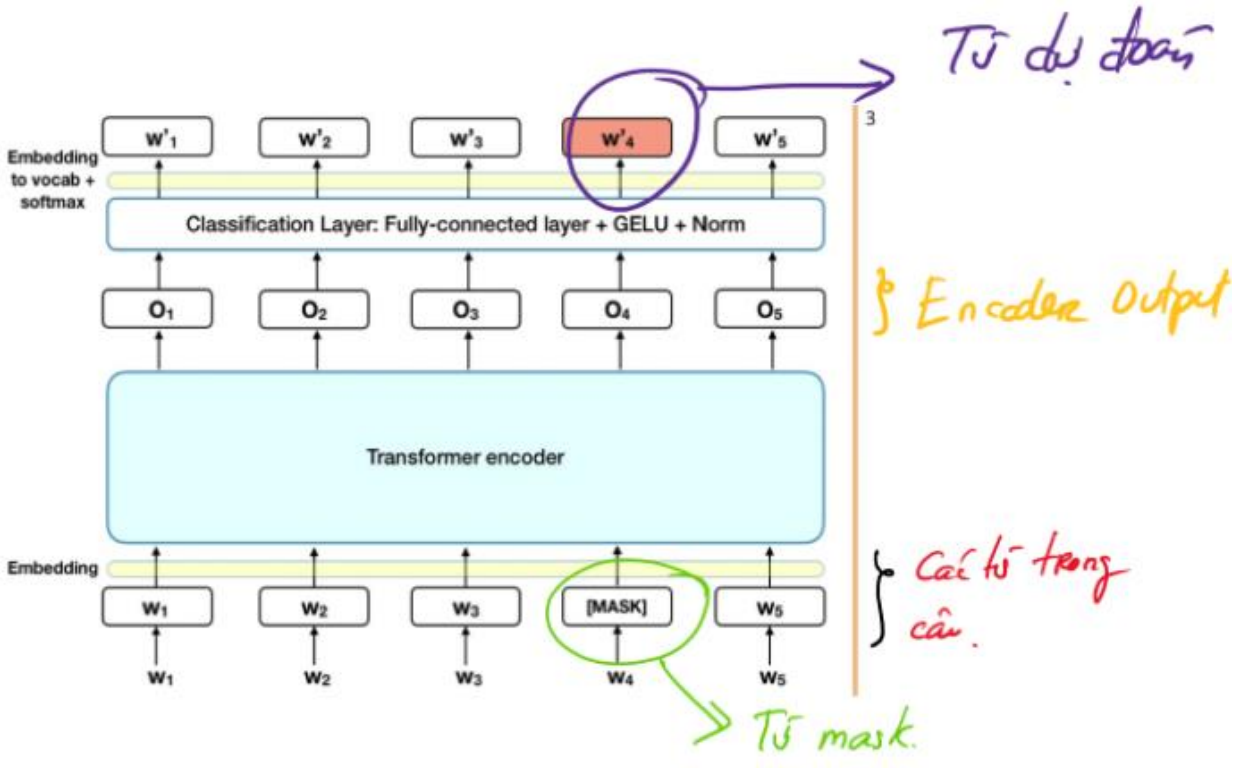
BERT được huấn luyện đồng thời 2 nhiệm vụ gọi là **Masked LM** (để dự đoán từ thiếu trong câu) và **Next Sentence Prediction** (NSP – dự đoán câu tiếp theo câu hiện tại). Hai nhiệm vụ này được huấn luyện đồng thời và loss tổng sẽ là kết hợp loss của 2 nhiệm vụ và mô hình sẽ cố gắng minimize loss tổng này. Chi tiết 2 nhiệm vụ này như sau:

1.8.1. Mô hình ngô ngữ được đánh dấu (Masked Language Model)

Với nhiệm vụ này, ta huấn luyện sẽ thực hiện che đi tầm 15% số từ trong câu và đưa vào mô hình. Và ta sẽ huấn luyện để mô hình predict ra các từ bị che đó dựa vào các từ còn lại

Cụ thể là:

- Thêm một lớp classification lên trên encoder đầu ra
- Đưa các véc tơ trong encoder ouput về véc tơ bằng với vocab size, sau đó softmax để chọn ra từ tương ứng tại mỗi vị trí trong câu.
- Loss sẽ được tính tại vị trí masked và bỏ qua các vị trí khác (để đánh giá xem mô hình dự đoán từ mask đúng/sai ntn mà, các từ khác đâu có liên quan).



Hình 5. Sơ đồ kiến trúc BERT cho nhiệm vụ ngôn ngữ mô hình được đánh dấu

Theo đó:

- Khoảng 15 % các mã của câu đầu váo được thay thế bởi [MASK] mã trước khi truyền vào mô hình đại diện cho những từ bị che dấu (masked). Mô hình sẽ dựa trên các từ không được che (non-masked) dấu xung quanh [MASK] và đồng thời là bối cảnh của [MASK] để dự báo giá trị gốc của từ được che dấu. Số lượng từ được che dấu được lựa chọn là một số ít (15%) để tỷ lệ bối cảnh chiếm nhiều hơn (85%).
- Bản chất của kiến trúc BERT vẫn là một mô hình seq2seq gồm 2 pha encoder giúp nhúng các từ đầu váo và decoder giúp tìm ra phân phối xác suất của các từ ở đầu ra. Kiến trúc Transfomer encoder được giữ lại trong nhiệm vụ Masked ML. Sau khi thực hiện self-attention và feed forward ta sẽ thu được các véc tơ nhúng ở đầu ra là O1,O2,…,O5
- Để tính toán phân phối xác suất cho từ đầu ra, chúng ta thêm một Fully connect layer ngay sau Transformer Encoder. Hàm softmax có tác dụng tính toán phân phối xác suất. Số lượng units của fully connected layer phải bằng với kích thước của từ điển.

- Cuối cùng ta thu được véc tơ nhúng của mỗi một từ tại vị trí MASK sẽ là nhúng véc tơ giảm chiều của véc tơ Oi sau khi đi qua fully connected layer như mô tả trên hình vẽ bên phải.

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 đầu váo cho mô hình huấn luyện nên mô hình sẽ cần phải huấn luyện rất lâu mới học được toàn diện các khả năng.

1.8.2. Next Sentence Prediction (NSP)

Với nhiệm vụ này thì mô hình sẽ được feed cho một cặp câu và nhiệm vụ của nó là đầu ra ra giá trị 1 nếu câu thứ hai đúng là câu đi sau câu thứ nhất và 0 nếu không phải. Trong quá trinh huấn luyện, ta chọn 50% mẫu là Positive (đầu ra là 1) và 50% còn lại là Negative được ghép linh tinh (đầu ra là 0).

Cụ thể cách huấn luyện như sau:

- Bước 1: Ghép 2 câu vào nhau và thêm 1 số mã đặc biệt để phân tách các câu. Mã [CLS] thêm vào đầu cầu thứ nhất, mã [SEP] thêm vào cuối mỗi câu.

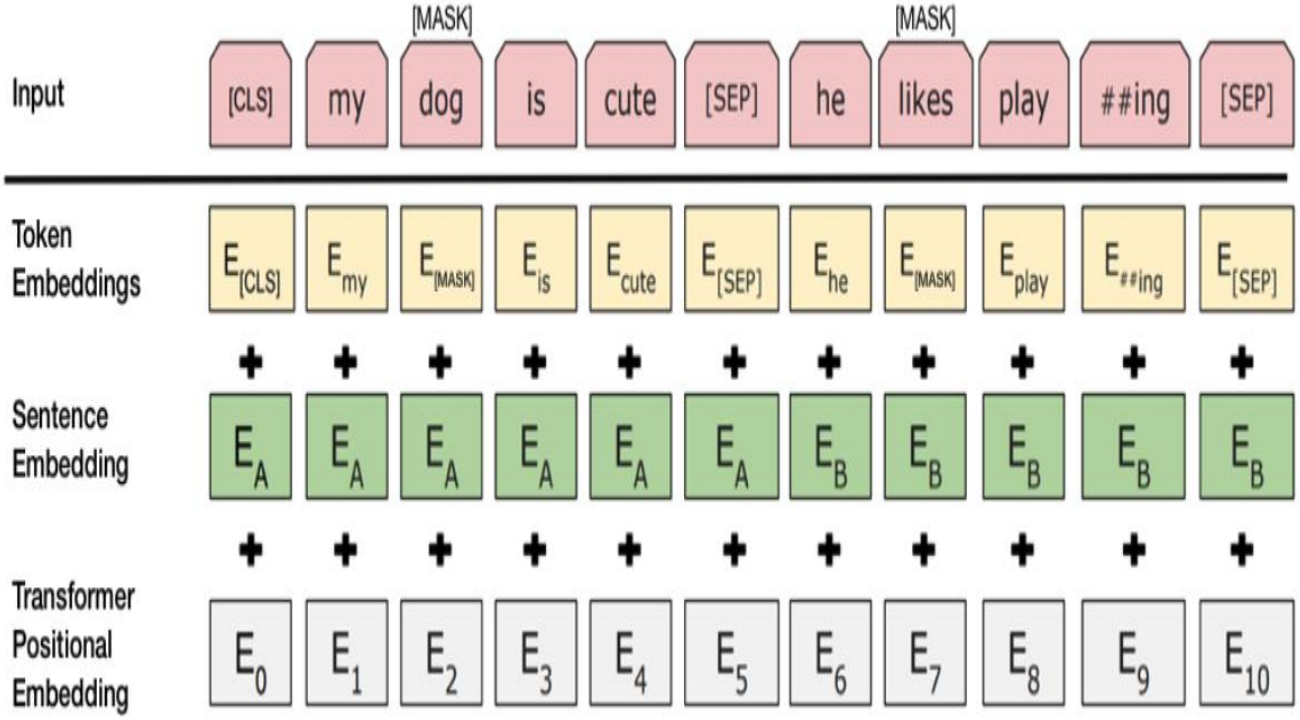
Ví dụ: ghép 2 câu “ Hôm nay em đi học ” và “ Học ở trường rất vui ” thì sẽ thành [CLS] Hôm nay em đi học [SEP] Học ở trường rất vui [SEP]

- Bước 2. Mỗi mã trong câu sẽ được cộng thêm một véc tơ gọi là Nhúng câu (Sentence Embedding), thực ra là đánh dấu xem từ đó thuộc câu Thứ nhất hay câu thứ 2.

Ví dụ: nếu thuộc câu thứ nhất thì cộng thêm 1 véc tơ toàn số “ 0 ” có kích thước bằng Từ nhúng, và nếu thuộc câu thứ 2 thì cộng thêm một véc tơ toàn số “ 1 ”.

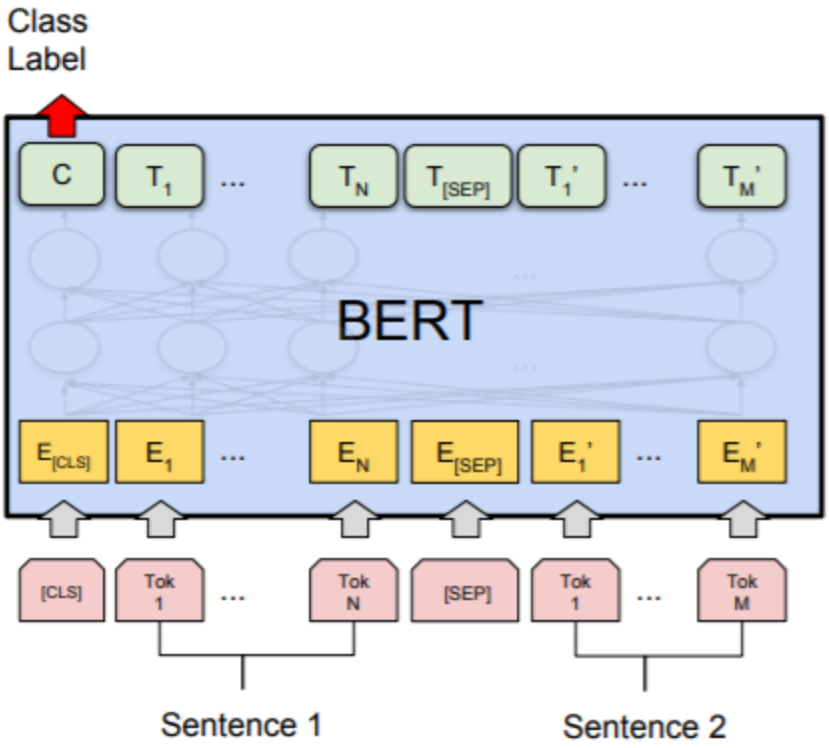
- Bước 3. Sau đó các từ trong câu đã ghép sẽ được thêm véc tơ mã hóa vị trí (Positional Encoding) vào để đánh dấu vị trí từng từ trong câu đã ghép.
- Bước 4. Đưa chuỗi sau bước 3 vào mạng.
- Bước 5. Lấy encoder đầu ra tại vị trí mã [CLS] được biến đổi (transform) sang một véc tơ có 2 phần tử [c1 c2].
- Bước 6. Tính softmax trên véc tơ đó và đầu ra ra khả năng của 2 lớp: Đi sau và Không đi sau. Để thể hiện câu thứ hai là đi sau câu thứ nhất hay không, ta lấy argmax là được.

Các bước tạo Đầu vào:



Hình 6. Các bước tạo Đầu váo trong tác vụ NSP

Và đây là cách lấy đầu ra:



Hình 7. Mô hình đầu ra của NSP

Thông tin đầu vào được tiền xử lý trước khi đưa vào mô hình huấn luyện bao gồm:

- Ngữ nghĩa của từ (token embeddings): Thông qua các nhúng véc tơ cho từng từ. Các véc tơ được khởi tạo từ mô hình huấn luyện trước.

Ngoài nhúng biểu diễn từ của các từ trong câu, mô hình còn nhúng thêm một số thông tin:

- Loại câu (segment embeddings): Gồm hai véc tơ là EA nếu từ thuộc câu thứ nhất và EB nếu từ thuộc câu thứ hai.
- Vị trí của từ trong câu (position embedding): là các véc tơ E0,…,E10. Tương tự như nhúng vị trí (positional embedding) trong transformer.

Véc tơ đầu vào sẽ bằng tổng của cả ba thành phần nhúng theo từ, câu và vị trí.

1.9. Các kiến trúc mô hình BERT

Hiện tại có nhiều phiên bản khác nhau của mô hình BERT. Các phiên bản đều dựa trên việc thay đổi kiến trúc của Transformer tập trung ở 3 tham số:

- L: số lượng các khối các tầng con trong transformer
- H: kích thước của véc tơ nhúng (hay còn gọi là hidden size)
- A: Số lượng từ đầu (head) trong tầng nhiều từ đầu (multi-head layer), mỗi một từ đầu sẽ thực hiện một có chế tự chú ý (self-attention).

Tên gọi của 2 kiến trúc bao gồm:

- BERTBASE(L=12,H=768,A=12): Tổng tham số 110 triệu.
- BERTLARGE(L=24,H=1024,A=16): Tổng tham số 340 triệu.

Như vậy ở kiến trúc BERT Large chúng ta tăng gấp đôi số tầng, tăng kích thước ẩn của véc tơ nhúng gấp 1.33 lần và tăng số lượng từ đầu trong multi-head layer gấp 1.33 lần.

1.10. RoBerta

1.10.1. Khái niệm RoBerta

RoBERTa là một project của facebook kế thừa lại các kiến trúc và thuật toán của mô hình BERT trên framework pytorch (pytorch cũng là một framework do facebook phát triển, rất được ưa chuộng bởi cộng đồng AI). Đây là một project hỗ trợ việc huấn luyện lại các mô hình BERT trên những bộ dữ liệu mới cho các ngôn ngữ khác ngoài một số ngôn ngữ phổ biến. Kể từ khi ra đời, đã có rất nhiều các mô hình pretrain cho những ngôn ngữ khác nhau được huấn luyện trên RoBERTa.

Ở bài báo gốc cho biết mặc dù RoBERTa lặp lại các thủ tục huấn luyện từ mô hình BERT, nhưng có một thay đổi đó là huấn luyện mô hình lâu hơn, với batch size lớn hơn và trên nhiều dữ liệu hơn. Ngoài ra để nâng cao độ chuẩn xác trong biểu diễn từ thì RoBERTa đã loại bỏ nhiệm vụ dự đoán câu tiếp theo và huấn luyện trên các câu dài hơn. Đồng thời mô hình cũng thay đổi linh hoạt kiểu masking (tức ẩn đi một số từ ở câu đầu ra bằng mã <mask>) áp dụng cho dữ liệu huấn luyện.

1.10.2. Dữ liệu

Quan sát thấy rằng việc huấn luyện BERT trên các bộ dữ liệu lớn hơn, cải thiện đáng kể hiệu suất của nó. Vì vậy, RoBERTa được huấn luyện về một tập dữ liệu khổng lồ có hơn 160GB văn bản không nén. Tập dữ liệu này bao gồm kho tài liệu sau:

- BookCorpus + Wikipedia tiếng Anh (16GB) : Đây là dữ liệu mà BERT được huấn luyện.

- CC-News (76GB) : Các tác giả đã thu thập dữ liệu này từ phần tiếng Anh của CommonCrawl News Data. Nó chứa 63 triệu tin bài tiếng Anh được thu thập từ tháng 9 năm 2016 đến tháng 2 năm 2019.
- OpenWebText (38GB) : Phiên bản mã nguồn mở của tập dữ liệu WebText được sử dụng để huấn luyện OpenAI GPT .
- Câu chuyện (31GB) : Một tập hợp con dữ liệu CommonCrawl được lọc để phù hợp với phong cách giống câu chuyện của lược đồ Winograd.

Mục tiêu mô hình ngôn ngữ được che giấu trong huấn luyện trước BERT về cơ bản là che giấu một vài mã thông báo từ mỗi chuỗi một cách ngẫu nhiên và sau đó dự đoán các mã thông báo này. Tuy nhiên, trong quá trình triển khai ban đầu của BERT, các trình tự bị che chỉ một lần trong quá trình tiền xử lý. Điều này ngụ ý rằng cùng một mẫu mặt nạ được sử dụng cho cùng một trình tự trong tất cả các bước huấn luyện.

Để tránh điều này, trong quá trình triển khai lại BERT, các tác giả đã sao chép dữ liệu huấn luyện 10 lần để mỗi trình tự được che thành 10 mẫu khác nhau. Điều này được huấn luyện trong 40 kỷ nguyên, tức là mỗi chuỗi được huấn luyện cho cùng một mẫu mặt nạ 4 lần.

Ngoài ra, chúng tôi đã thử tạo mặt nạ động, trong đó một mẫu mặt nạ được tạo ra mỗi khi một trình tự được cung cấp cho mô hình.

Các kết quả nói trên cho thấy rằng việc thực hiện lại với tạo mặt nạ tĩnh mang lại kết quả gần như tương tự như cách tiếp cận tạo mặt nạ ban đầu của BERT. Mặt nạ động có kết quả tương đương hoặc tốt hơn một chút so với phương pháp tiếp cận tĩnh. Do đó, trong RoBERTa, phương pháp tạo mặt nạ động được sử dụng để huấn luyện trước.

1.10.2.1. Kích thước hàng loạt lớn

Nghiên cứu trước đây đã chỉ ra rằng các mô hình Transformer và BERT có thể đáp ứng được với kích thước lô lớn. Có kích thước lô lớn giúp tối ưu hóa nhanh hơn và có thể cải thiện hiệu suất nhiệm vụ cuối khi được điều chỉnh chính xác (trong trường hợp của các mô hình này).

Model	data	bsz	steps	SQuAD (v1.1/2.0)	MNLI-m	SST-2
RoBERTa						
with BOOKS + WIKI	16GB	8K	100K	93.6/87.3	89.0	95.3
+ additional data (§3.2)	160GB	8K	100K	94.0/87.7	89.3	95.6
+ pretrain longer	160GB	8K	300K	94.4/88.7	90.0	96.1
+ pretrain even longer	160GB	8K	500K	**94.6/89.4**	**90.2**	**96.4**
$\text{BERT}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	90.9/81.8	86.6	93.7
$\text{XLNet}_{\text{LARGE}}$						
with BOOKS + WIKI	13GB	256	1M	94.0/87.8	88.4	94.4
+ additional data	126GB	2K	500K	94.5/88.8	89.8	95.6

Bảng 4. Hiệu suất với các kích thước lô khác nhau của các mô hình

1.10.2.2. Mã hóa BPE

Toknenize là quá trình mã hóa các văn bản thành các index dạng số mang thông tin của văn bản để cho máy tính có thể huấn luyện được. Khi đó mỗi một từ hoặc ký tự sẽ được đại diện bởi một index.

Trong NLP có một số kiểu tokenize như sau:

1. **Tokenize theo word level**: Chúng ta phân tách câu thành các mã được ngăn cách bởi khoảng trắng hoặc dấu câu. Khi đó mỗi mã là một từ đơn âm tiết. Đây là phương pháp mã được sử dụng trong các thuật toán nhúng từ truyền thống như GloVe, word2vec.
2. **Tokenize theo multi-word level**: Tiếng Việt và một số ngôn ngữ khác tồn tại từ đơn âm tiết (từ đơn) và từ đa âm tiết (từ ghép). Do đó nếu mã theo từ đơn âm tiết sẽ làm nghĩa của từ bị sai khác. Ví dụ cụm từ vô xác định nếu được chia thành vô, xác và định sẽ làm cho từ bị mất đi nghĩa phủ định của nó. Do đó để tạo ra được các từ với nghĩa chính xác thì chúng ta sẽ sử dụng thêm từ điển bao gồm cả từ đa âm tiết và đơn âm để tokenize câu. Trong Tiếng Việt có khá nhiều các module hỗ trợ tokenize dựa trên từ điển như VnCoreNLP, pyvivn, underthesea.
3. **Tokenize theo character level:** Việc tokenize theo word level thường sinh ra một từ điển với kích thước rất lớn, điều này làm gia chi phí tính toán. Hơn nữa nếu tokenize theo word level thì đòi hỏi từ điển phải rất lớn thì mới hạn chế được những trường hợp từ nằm ngoài từ điển. Tuy nhiên nếu phân tích ta sẽ thấy hầu hết các từ

đều có thể biểu thị dưới một nhóm các ký tự là chữ cái, con số, dấu xác định. Như vậy chỉ cần sử dụng một lượng các ký tự rất nhỏ có thể biểu diễn được mọi từ. Từ được token dựa trên level ký tự sẽ có tác dụng giảm kích thước từ điển mà vẫn biểu diễn được các trường hợp từ nằm ngoài từ điển. Đây là phương pháp được áp dụng trong mô hình fasttext.

Phương pháp mới BPE (SOTA): Nhược điểm của phương pháp tokenize theo character level đó là các token không có ý nghĩa nếu đứng độc lập. Do đó đối với các bài toán sentiment analysis, áp dụng tokenize theo character level sẽ mang lại kết quả kém hơn. Token theo word level cũng tồn tại hạn chế đó là không giải quyết được các trường hợp từ ngằm ngoài từ điển.

Một phương pháp mới đã được đề xuất trong bài báo Neural Machine Translation of Rare Words with Subword Units vào năm 2016, có khả năng tách từ theo level nhỏ hơn từ và lớn hơn ký tự được gọi là subword. Phương pháp đó chính là BPE (byte pair encoding). Theo phương pháp mới này, hầu hết các từ đều có thể biểu diễn bởi subword và chúng ta sẽ hạn chế được một số lượng đáng kể các token <unk> đại diện cho từ chưa từng xuất hiện trước đó. Rất nhanh chóng, Phương pháp mới đã được áp dụng ở hầu hết các phương pháp NLP hiện đại từ các lớp mô hình BERT cho tới các biến thể của nó như OpenAI GPT, RoBERTa, DistilBERT, XLMNet. Kết quả áp dụng tokenize theo phương pháp mới đã cải thiện được độ chính xác trên nhiều nhiệm vụ dịch máy, phân loại văn bản, dự báo câu tiếp theo, hỏi đáp, dự báo mối quan hệ văn bản.

Thuật toán BPE:

BPE (Byte Pair Encoding) là một kỹ thuật nén từ cơ bản giúp chúng ta index được toàn bộ các từ kể cả trường hợp từ mở (không xuất hiện trong từ điển) nhờ mã hóa các từ bằng chuỗi các từ phụ (subwords). Nguyên lý hoạt động của BPE dựa trên phân tích trực quan rằng hầu hết các từ đều có thể phân tích thành các thành phần con.

Chẳng hạn như từ: low, lower, lowest đều là hợp thành bởi low và những đuôi phụ er, est. Những đuôi này rất thường xuyên xuất hiện ở các từ. Như vậy khi biểu diễn từ lower chúng ta có thể mã hóa chúng thành hai thành phần từ phụ (subwords) tách biệt là low và er. Theo cách biểu diễn này sẽ không phát sinh thêm một index mới cho từ lower và đồng thời tìm được mối liên hệ giữa lower, lowest và low nhờ có chung thành phần từ phụ là low.

Phương pháp BPE sẽ thống kê tần suất xuất hiện của các từ phụ cùng nhau và tìm cách gộp chúng lại nếu tần suất xuất hiện của chúng là lớn nhất. Cứ tiếp tục quá trình gộp từ phụ cho tới khi không tồn tại các subword để gộp nữa, ta sẽ thu được tập subwords cho toàn bộ bộ văn bản mà mọi từ đều có thể biểu diễn được thông qua subwords.

Code của thuật toán BPE đã được tác giả chia sẻ tại subword-nmt.

Quá trình này gồm các bước như sau:

- Bước 1: Khởi tạo từ điển (vocabulary).
- Bước 2: Biểu diễn mỗi từ trong bộ văn bản bằng kết hợp của các ký tự với token <\w> ở cuối cùng đánh dấu kết thúc một từ (lý do thêm token sẽ được giải thích bên dưới).
- Bước 3: Thống kê tần suất xuất hiện theo cặp của toàn bộ token trong từ điển.
- Bước 4: Gộp các cặp có tần suất xuất hiện lớn nhất để tạo thành một n-gram theo level character mới cho từ điển.
- Bước 5: Lặp lại bước 3 và bước 4 cho tới khi số bước triển khai merge đạt đỉnh hoặc kích thước kỳ vọng của từ điển đạt được.

	MNLI	QNLI	QQP	RTE	SST	MRPC	CoLA	STS	WNLI	Avg
Single-task single models on dev										
$\text{BERT}_{\text{LARGE}}$	86.6/-	92.3	91.3	70.4	93.2	88.0	60.6	90.0	-	-
$\text{XLNet}_{\text{LARGE}}$	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**	-
Ensembles on test (from leaderboard as of July 25, 2019)										
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**

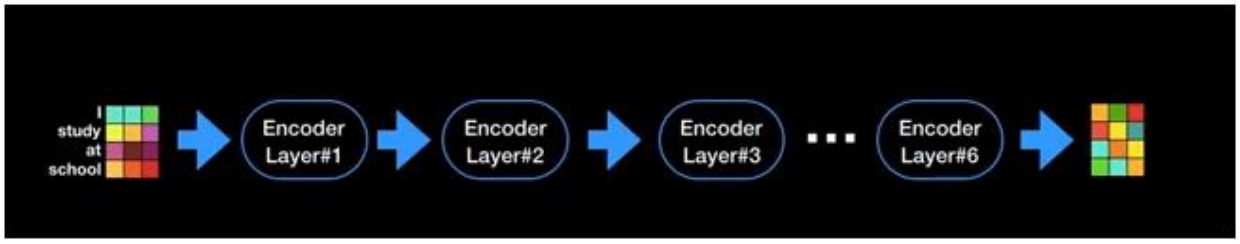
Bảng 5. Hiệu suất trên GLUE BenchMARK

1.10.3. Extract fearture từ RoBerta

Có 2 versions chính trả về 2 kích thước nhúng khác nhau khi huấn luyện theo RoBERTa đó là:

- BERT base: 12 tầng con, kích thước nhúng 768, số lượng head attention là 12.
- BERT large: 24 tầng con, kích thước nhúng 1024, số lượng head attention là 16.

Chúng ta có thể trích xuất được các đặc trưng được tạo ra từ BERT của pha Encoder tại các tầng cuối cùng hoặc toàn bộ các tầng. Ngoài phương án trích suất véc tơ nhúng tại layer cuối cùng, một số nhiệm vụ classification trong NLP đã áp dụng trích suất đặc trưng từ những tầng trước đó chẳng hạn như trong nhiệm vụ PhoBERT Sentiment Classification tác giả đã trích suất đặc trưng từ 4 tầng cuối cùng thay vì chỉ trích suất từ một layer cuối.



Hình 8. Kiến trúc gồm nhiều tầng tại encoder của mô hình BERT

Mô hình huấn luyện từ RoBERTa cho phép ta trích suất các đặc trưng từ những tầng của encoder. Có thể là layer cuối hoặc toàn bộ các tầng. Kích thước đầu ra của mỗi một layer sẽ là batch_size x seq_len x d_model

1.10.4. Điền từ (Filling Mask)

Trên thực tế có rất nhiều ứng dụng của bài toàn filling mask như xây dựng hệ thống suggestion search, gợi ý gõ văn bản, tìm từ đồng nghĩa, tagging.

Mô hình BERT tạo ra các biểu diễn từ, từ quá trình ẩn các vị trí token một cách ngẫu nhiên trong câu đầu váo và dự báo chính, từ đó ở đầu ra dựa trên bối cảnh là các từ xung quanh.

Như vậy khi đã biết các từ xung quanh, chúng ta hoàn toàn có thể dự báo được từ phù hợp nhất với vị trí đã được masking.

1.10.5. Trích suất đặc trưng (Extract feature) cho từ

Sau khi load được mô hình BERT, chúng ta hoàn toàn có thể trích suất đặc trưng cho một từ bất kỳ từ mô hình huấn luyện trước. Từ các véc tơ nhúng được trích suất cho một từ hoặc một câu, chúng ta có thể đo lường similarity để tìm ra các câu tương đồng về nội dung hoặc các từ đồng nghĩa. Một ứng dụng khác đó là chúng ta có thể tận dụng các biểu diễn ngữ nghĩa của từ thông qua véc tơ nhúng được huấn luyện từ BERT để chuyển giao sang các nhiệm vụ phân loại văn bản, phân tích cảm xúc bình luận. Phương pháp tiếp cận sẽ tương tự như áp dụng các mô hình GloVe, word2vec, fasttext trong học nông (shallow learning)

CHƯƠNG 2. PHOBERT

2.1. Sự ra đời của PhoBERT

Mặc dù BERT là một nghiên cứu mới mang đầy tính đột phá, một bước nhảy vọt thực sự của Google trong lĩnh vực xử lý ngôn ngữ tự nhiên. Sự ra đời của mô hình huấn luyện trước BERT đã mang lại những cải tiến đáng kể cho rất nhiều bài toán như Question Answering, Sentiment Analysis,...

Tuy nhiên , huấn luyện mô hình BERT cho Tiếng Việt lại không hề đơn giản do đó rất khó để có thể áp dụng BERT cho các nhiệm vụ Tiếng Việt dù cho Google cũng có huấn luyện trước cho nhiều ngôn ngữ (pre-trained multilingual) bao gồm cả tiếng Việt nhưng chưa cho kết quả thực hiện tốt nhất.

Cho đến nay BERT vẫn được sử dụng cho nhiều bài toán NLP cho kết quả tốt với các phiên bản cải tiến, biến thể như RoBERTa, ALBERT, DistilBERT,... BERT đã được áp sụng cho nhiều nhiệm vụ xử lý ngôn ngữ tự nhiên và trở lên áp đảo trong các nền tảng thi đấu như Kaggle, AIVIVN cũng như được trình bày tại nhiều hội nghị.

PhoBERT đã được ra đời là một mô hình BERT được huấn luyện trước cho tiếng Việt và đạt được nhiều kết quả tốt nhất cho nhiều nhiệm vụ trong xử lý ngôn ngữ tiếng Việt. Tác giả lấy tên Pho vì đây là món ăn phổ biến của Việt Nam.

PhoBERT dễ sử dụng, nó được xây dựng để sử dụng trong các thư viện như FAIRSeq của Facebook hay Transformers của Hugging Face nên giờ đây BERT lại càng phổ biến ngay cả với ngôn ngữ tiếng Việt hay tiếng Anh.

2.2. Cấu trúc của PhoBERT

Đây là một mô hình huấn luyện trước được huấn luyện cho đơn ngôn ngữ (monolingual language), tức là chỉ huấn luyện dành riêng cho tiếng Việt. Tương tự như BERT, PhoBERT cũng có 2 phiên bản là $\text{PhoBERT}_{\text{base}}$ với 12 khối transformers và $\text{PhoBERT}_{\text{large}}$ với 24 khối transformers.

PhoBERT được huấn luyện trên khoảng 20GB dữ liệu bao gồm khoảng 1GB ngữ liệu Wikipedia Tiếng Việt (Vietnamese Wikipedia corpus) và 19GB còn lại lấy từ ngữ liệu tin tức tiếng Việt. Đây là một lượng dữ liệu đủ lớn để huấn luyện một mô hình như BERT.

PhoBERT sử dụng RDRSegmenter của VnCoreNLP để tách từ cho dữ liệu đầu vào trước khi qua mã hóa BPE. PhoBERT chỉ sử dụng nhiệm vụ Mô hình ngôn ngữ đánh dấu để huấn luyện và không sử dụng nhiệm vụ dự đoán câu tiếp theo.

PhoBERT có hai phiên bản: $\text{PhoBERT}_{\text{base}}$ và $\text{PhoBERT}_{\text{large}}$

Là mô hình ngôn ngữ đơn ngữ có quy mô lớn được huấn luyện huấn luyện dành riêng cho Tiếng Việt. Thực nghiệm kết quả cho thấy PhoBERT luôn hoạt động tốt hơn so với mô hình đa ngôn ngữ được huấn luyện trước tốt nhất gần đây XLM-R và cải thiện tính năng hiện đại trong nhiều nhiệm vụ NLP dành riêng cho tiếng Việt bao gồm các nhiệm vụ như: phân tích giọng nói, Phân tích cú pháp phụ thuộc, Nhận dạng thực thể được đặt tên và Suy luận ngôn ngữ tự nhiên.

Các mô hình ngôn ngữ được huấn luyện trước, đặc biệt là BERT(Devlin và cộng sự, 2019) gần đây đã trở nên cực kỳ phổ biến và tạo ra các cải tiến đáng kể cho các nhiệm vụ NLP khác nhau. Sự thành công của BERT được huấn luyện và các biến thể của nó phần lớn đạt được kết quả tốt trong ngôn ngữ tiếng Anh.

Đối với ngôn ngữ khác, người ta có thể huấn luyện lại bằng cách sử dụng mô hình kiến trúc BERT (Cui ,de Vries, Vu, Martin và các cộng sự ra mắt vào năm 2020) hoặc sử dụng các mô hình dựa trên BERT đa ngôn ngữ đã được huấn luyện trước (Devlin et al., 2019; Conneau và Lample, 2019; Conneau và cộng sự, 2020).

Về mô hình tiếng Việt, có hai vấn đề cần quan tâm chính như sau:

- Kho tài liệu Wikipedia tiếng Việt là dữ liệu duy nhất được sử dụng để huấn luyện các mô hình ngôn ngữ đơn ngữ (Vu et al., 2019) và đây cũng là tập dữ liệu tiếng Việt duy nhất được đưa vào làm dữ liệu tiền huấn luyện được sử dụng bởi tất cả các mô hình đa ngôn ngữ ngoại trừ XLM-R.

Có một lưu ý rằng: Dữ liệu Wikipedia không đại diện cho một chi sử dụng ngôn ngữ và Wikipedia dữ liệu Tiếng Việt tương đối nhỏ (kích thước 1GB không nén), trong khi các mô hình ngôn ngữ được huấn luyện trước có thể được cải thiện đáng kể bằng cách sử dụng nhiều dữ liệu tiền huấn luyện hơn (Liu và cộng sự, 2019).

- Tất cả các mô hình ngôn ngữ dựa trên BERT đơn và đa ngôn ngữ được phát hành công khai đều không nhận thức được sự khác nhau giữa âm tiết tiếng Việt và mã từ.

Sự mơ hồ này đến từ thực tế là khoảng trắng cũng là được sử dụng để tách các âm tiết tạo thành từ khi viết bằng tiếng Việt.

Ví dụ: Một câu văn được viết 6 âm tiết " Tôi là một nghiên cứu viên " tạo thành 4 từ " $Tôi_{I}$ is_{am} $một_{a}$ $nghiên\_cứu\_viên_{researcher}$ ".

Không cần thực hiện một bước tiền xử lý tiếng Việt như phân đoạn từ, những mô hình đó áp dụng trực tiếp phương pháp mã hóa theo cặp (BPE) (Sennrich và cộng sự, 2016; Kudo và Richardson, 2018) với dữ liệu trước khi luyện tập tiếng Việt cấp độ âm tiết. Đặc biệt, đối với các nhiệm vụ NLP tiếng Việt cấp độ từ, những mô hình đó được huấn luyện trước trên dữ liệu cấp độ âm tiết có thể không hoạt động tốt như các mô hình ngôn ngữ được huấn luyện trước về dữ liệu cấp độ từ.

Để giải quyết hai mối quan tâm trên, các tác giả huấn luyện dựa trên mô hình đơn ngữ quy mô lớn đầu tiên $BERT_{base}$ và $BERT_{large}$ tạo ra kho dữ liệu tên Tiếng Việt có tới 20GB cấp độ từ. Họ đánh giá các mô hình của mình trên bốn nhiệm vụ sau NLP cho tiếng Việt: gán nhãn từ loại (Part-of-speech - POS), phân tích cú pháp phụ thuộc (Dependency Paser) và nhận dạng thực thể tên (NER), và nhiệm vụ hiểu ngôn ngữ của suy luận ngôn ngữ tự nhiên (NLI) có thể được xây dựng như một nhiệm vụ cấp độ âm tiết hoặc từ.

Kết quả thử nghiệm cho thấy rằng các mô hình của tác giả cho kết quả tốt nhất hiện tại (SOTA) trên tất cả các nhiệm vụ này. Những đóng góp đó như sau:

- Tạo ra các mô hình ngôn ngữ đơn ngữ quy mô lớn đầu tiên được huấn luyện huấn luyện trước dành riêng cho Tiếng Việt.
- Các mô hình đó giúp tạo ra các kết quả SOTA trên bốn nhiệm vụ cơ bản là gắn nhãn từ loại POS (Part-of-speech), phân tích cú pháp phụ thuộc DP (Dependency Paser), nhận dạng thực thể NER (Name Entity Recognition) và suy luận ngôn ngữ tự nhiên NL I (Natural Languague Inference), do đó cho thấy hiệu quả của việc dựa trên BERT quy mô lớn các mô hình ngôn ngữ đơn ngữ cho tiếng Việt.
- Thực hiện bộ thử nghiệm đầu tiên để so sánh các mô hình ngôn ngữ đơn ngữ với mô hình đa ngôn ngữ tốt nhất gần đây XLM-R trong bốn nhiệm vụ cơ bản trên. Các thử nghiệm cho thấy rằng các mô hình hoạt động tốt hơn XLM-R về tất cả các nhiệm vụ này, do đó có thể thấy rằng các mô hình dành riêng cho ngôn ngữ cụ thể vẫn tốt hơn các mô hình đa ngôn ngữ.

- Đưa ra các mô hình dưới tên PhoBERT có thể được sử dụng với fairseq (Ott và cộng sự, 2019) và transformer (Wolf và cộng sự, 2019). Qua đó hi vọng PhoBERT có thể phục vụ như một cơ sở vững chắc cho NLP tiếng Việt trong tương lai để nghiên cứu và ứng dụng.

2.2.1. Dữ liệu trước khi huấn luyện

Để xử lý mối quan tâm đầu tiên được đề cập trong phần giới thiệu trên , chúng ta sử dụng tập dữ liệu 20GB được huấn luyện trước của các văn bản. Tập dữ liệu này là sự kết hợp của hai kho ngữ liệu:

- Kho ngữ liệu Wikipedia tiếng Việt(~1GB)
- Kho văn bản thứ hai (~19GB) là được tạo ra bằng cách xóa các bài báo tương tự và trùng lặp khỏi kho tin tức tiếng Việt(50GB)

Để giải quyết vấn đề cho kho ngữ liệu thứ hai, chúng ta sử dụng RDRSegmenter (Nguyen et al.,2018) từ VnCoreNLP(Vu et al., 2018) để tách từ và phân đoạn câu trên tập dữ liệu huấn luyện trước, kết quả ~145M câu được tách từ (~3B từ mã thông báo). Khác với RoBERTa, chúng ta áp dụng fastBPE (Sennrich et al., 2016) để phân đoạn những câu với các đơn vị từ khóa phụ, sử dụng từ vựng trong tổng số 64K loại từ khóa phụ. Trung bình có 24,4 mã thông báo từ phụ cho mỗi câu.

2.2.2. Tối ưu hóa

Sử dụng việc triển khai RoBERTa trong fairseq (Ott và cộng sự, 2019). Chúng ta đặt độ dài tối đa là 256 mã thông báo từ khóa phụ, do đó tạo ra câu 145M × 24,4 / 256 ≈ 13,8M các khối. Theo Liu et al. (2019), chúng ta tối ưu hóa sử dụng các mô hình Adam (Kingma và Ba, 2014).

Chúng ta sử dụng kích thước lô 1024 trên 4 GPU V100(16GB mỗi) và tốc độ huấn luyện cao nhất là 0,0004 cho $\text{PhoBERT}_{\text{base}}$ và kích thước lô là 512 và đỉnh tỷ lệ huấn luyện là 0,0002 cho $\text{PhoBERT}_{\text{large}}$. Chúng ta chạy trong 40 kỷ nguyên trong 2 kỷ nguyên), do đó dẫn đến 13,8M × 40/ 1024 ≈ 540K bước huấn luyện cho $\text{PhoBERT}_{\text{base}}$ và 1,08 triệu bước huấn luyện cho $\text{PhoBERT}_{\text{large}}$. $\text{PhoBERT}_{\text{base}}$ được huấn luyện huấn luyện trước trong 3 tuần và sau đó $\text{PhoBERT}_{\text{large}}$ trong 5 tuần.

2.2.3. Thiết lập thử nghiệm

Hiệu suất của PhoBERT được đánh giá trên bốn nhiệm vụ NLP tiếng Việt: gán nhãn từ loại POS, Phân tích cú pháp phụ thuộc, nhận dạng thực thể và xử lý ngôn ngữ tự nhiên.

- Với bốn nhiệm vụ này dữ liệu được phân chia như sau:

Task	#training	#valid	#test
POS tagging†	27,000	870	2,120
Dep. parsing†	8,977	200	1,020
NER†	14,861	2,000	2,831
NLI‡	392,702	2,490	5,010

Bảng 1. Thống kê các bộ dữ liệu

Trong đó : "#Training", "#valid" và "#test" biểu thị kích thước của bộ huấn luyện, xác nhận và đánh giá, tương ứng và xem kích thước tập dữ liệu là số câu và các cặp câu tương ứng.

Để cho nhiệm vụ gán nhãn từ loại, phân tích cú pháp Phụ thuộc và NER, chúng ta thực hiện các bước tiền xử lý sử dụng VnCoreNLP (Vu et al., 2018), sử dụng các đánh giá tiêu chuẩn của VLSP với tập dữ liệu gán nhãn từ loại 2013, ngân hàng câu phân tích phụ thuộc VnDT v1.1 (Nguyen et al., 2014b) với gán nhãn từ loại của VnCoreNLP và VLSP 2016 Bộ dữ liệu NER (Nguyễn và cộng sự, 2019a).

Đối với NLI, sử dụng bộ kiểm tra và xác thực Tiếng Việt được xây dựng thủ công từ kho ngữ liệu NLI (XNLI) phiên bản 1.0 (Conneau et al.,2018) trong đó bộ huấn luyện tiếng Việt được phát hành dưới dạng phiên bản dịch bằng máy của bộ huấn luyện tiếng Anh tương ứng (Williams và cộng sự, 2018). Không giống như gán nhãn từ loại, phân tích cú pháp Phụ thuộc và tập dữ liệu NER cung cấp từ vàng phân đoạn, đối với NLI, chúng tôi sử dụng RDRSegmenter để phân đoạn văn bản thành các từ trước khi áp dụng BPE để tạo ra các từ khóa con từ các mã thông báo từ.

- **Tinh chỉnh**

Theo Devlin và cộng sự. (2019), để gán nhãn từ loại và NER, chúng ta thêm một lớp dự đoán tuyến tính ở trên cùng của kiến trúc PhoBERT (tức là lớp Transformer cuối cùng của PhoBERT). Đối với sự phụ thuộc phân tích cú pháp, theo Nguyen (2019), chúng ta sử dụng sự tái hiện lại (Biaffine) phân tích cú pháp phụ thuộc hiện đại (Dozat và Manning,

2017) từ Ma và cộng sự(2018) với siêu tham số tối ưu mặc định. Sau đó mở rộng phân tích cú pháp này bằng cách thay thế nhúng từ được huấn luyện trước của mỗi từ trong một câu đầu vào bằng cách tính toán nhúng theo ngữ cảnh tương ứng (từ lớp cuối cùng) cho mã thông báo từ phụ đầu tiên của từ.

Đối với gán nhãn từ loại, NER và NLI, chúng ta sử dụng Transformer (Wolf và cộng sự, 2019) để tinh chỉnh PhoBERT cho từng nhiệm vụ và từng tập dữ liệu một cách độc lập. Sử dụng AdamW (Loshchilov và Hutter2019) với tốc độ huấn luyện cố định là 1.e-5 và kích thước khối là 32 (Liu và cộng sự, 2019). Chúng ta tinh chỉnh 30 kỷ nguyên huấn luyện, đánh giá việc thực hiện nhiệm vụ sau mỗi kỷ nguyên trên tập xác thực và sau đó chọn kỷ nguyên tốt nhất điểm đánh giá mô hình để báo cáo kết quả cuối cùng trên bộ đánh giá (lưu ý rằng mỗi điểm số là trung bình trên 5 lần chạy với các hạt ngẫu nhiên khác nhau).

2.2.4. Kết quả thực nghiệm

1) Kết quả cuối

POS tagging (word-level)		**Dependency parsing** (word-level)	
Model	Acc.	Model	LAS / UAS
RDRPOSTagger (Nguyen et al., 2014a) [♣]	95.1	_	_
BiLSTM-CNN-CRF (Ma and Hovy, 2016) [♣]	95.4	VnCoreNLP-DEP (Vu et al., 2018) [★]	71.38 / 77.35
VnCoreNLP-POS (Nguyen et al., 2017) [♣]	95.9	jPTDP-v2 [★]	73.12 / 79.63
jPTDP-v2 (Nguyen and Verspoor, 2018) [★]	95.7	jointWPD [★]	73.90 / 80.12
jointWPD (Nguyen, 2019) [★]	96.0	Biaffine (Dozat and Manning, 2017) [★]	74.99 / 81.19
XLM-R_{base} (our result)	96.2	Biaffine w/ XLM-R_{base} (our result)	76.46 / 83.10
XLM-R_{large} (our result)	96.3	Biaffine w/ XLM-R_{large} (our result)	75.87 / 82.70
$PhoBERT_{base}$	96.7	Biaffine w/ $PhoBERT_{base}$	**78.77 / 85.22**
$PhoBERT_{large}$	**96.8**	Biaffine w/ $PhoBERT_{large}$	77.85 / 84.32

Bảng 2. Điểm hiệu suất (tính bằng%) trên bộ kiểm tra gán nhãn từ loại và phân tích cú pháp phụ thuộc.

Trong đó : “Acc.”, “LAS” và “UAS” viết tắt tương ứng của Độ chính xác, Điểm phần đính kèm được gắn nhãn và Điểm phần đính kèm không được gắn nhãn (ở đây, tất cả các chỉ số đánh giá này được tính toán trên tất cả các mã thông báo từ, bao gồm cả dấu chấm câu). [♣] và [★] biểu thị kết quả được báo cáo tương ứng bởi Nguyen et al. (2017) và Nguyên (2019).

NER (word-level)		**NLI** (syllable- or word-level)	
Model	F_1	Model	Acc.
BiLSTM-CNN-CRF [♦]	88.3	_	_
VnCoreNLP-NER (Vu et al., 2018) [♦]	88.6	BiLSTM-max (Conneau et al., 2018)	66.4
VNER (Nguyen et al., 2019b)	89.6	mBiLSTM (Artetxe and Schwenk, 2019)	72.0
BiLSTM-CNN-CRF + ETNLP [♠]	91.1	multilingual BERT (Devlin et al., 2019) [■]	69.5
VnCoreNLP-NER + ETNLP [♠]	91.3	$XLM_{MLM+TLM}$ (Conneau and Lample, 2019)	76.6
$XLM\text{-}R_{base}$ (our result)	92.0	$XLM\text{-}R_{base}$ (Conneau et al., 2020)	75.4
$XLM\text{-}R_{large}$ (our result)	92.8	$XLM\text{-}R_{large}$ (Conneau et al., 2020)	79.7
$PhoBERT_{base}$	93.6	$PhoBERT_{base}$	78.5
$PhoBERT_{large}$	**94.7**	$PhoBERT_{large}$	**80.0**

Bảng 3. Điểm hiệu suất (tính bằng%) trong bộ dữ liệu đánh giá NER và NLI

Trong đó: [♦], [♠] và [■] biểu thị kết quả được báo cáo tương ứng bởi Vu và các cộng sự. (2018), Vu và các cộng sự (2019) và Wu và Dredze (2019). Lưu ý là kết quả Tiếng Việt NLI được huấn luyện cho XLM-R sẽ cao hơn khi tinh chỉnh sự kết hợp của tất cả 15 bộ dữ liệu huấn luyện từ ngữ liệu XNLI (tức là TRANSLATE-TRAIN-ALL: 79,5% cho $XLM\text{-}R_{base}$ và 83,4% $XLM\text{-}R_{large}$). Tuy nhiên, những kết quả đó có thể không thể so sánh được vì chúng ta chỉ sử dụng dữ liệu huấn luyện tiếng Việt đơn ngữ để tinh chỉnh.

Bảng 2 và 3 so sánh điểm số PhoBERT với kết quả đánh giá cao nhất trước đó, sử dụng cùng một thiết lập thử nghiệm. Rõ ràng là PhoBERT giúp tạo ra kết quả hiệu suất SOTA mới cho tất cả bốn nhiệm vụ ở trên.

Để gán nhãn từ loại, mô hình thần kinh chung WPD để gán nhãn từ loại và phân tích cú pháp phụ thuộc (Nguyen, 2019) và mô hình dựa trên đặc trưng VnCoreNLP-POS (Nguyen et al., 2017) là hai mô hình SOTA trước đó, có được độ chính xác vào khoảng 96,0%. PhoBERT cao hơn 0,8% độ chính xác hai mô hình này.

Đối với phân tích cú pháp Phụ thuộc, mức cao nhất trước đó điểm phân tích cú pháp LAS và UAS có được bởi Biaffine phân tích cú pháp lần lượt là 75,0% và 81,2%. PhoBERT giúp thúc đẩy trình phân tích cú pháp Biaffine với cải thiện tuyệt đối khoảng 4%, đạt được LAS ở mức 78,8% và UAS là 85,2%.

Đối với NER, $PhoBERT_{large}$ tạo ra 1,1 điểm F1 cao hơn $PhoBERT_{base}$. Ngoài ra, $PhoBERT_{base}$ cao hơn 2 điểm so với phương pháp SOTA trước đó và dựa trên mạng nơ-ron mô hình VnCoreNLP-NER (Vu et al., 2018) và BiLSTM-CNN-CRF (Ma và Hovy, 2016) được huấn luyện với bộ 15K dựa trên BERT nhúng từ ETNLP (Vu và cộng sự, 2019).

Đối với NLI, PhoBERT vượt trội hơn BERT đa ngôn ngữ (Devlin và cộng sự, 2019) và mô hình đa ngôn ngữ $BERT_{base}$ với mục tiêu mô hình ngôn ngữ dịch mới XLMMLM + TLM (Conneau và Lample, 2019) với độ chính xác cao. PhoBERT cũng hoạt động tốt hơn mô hình đa ngôn ngữ tốt nhất gần đây được huấn luyện trước XLM-R nhưng sử dụng ít tham số hơn XLM-R: 135M ($PhoBERT_{base}$) so với 250M (XLM-Rbase); 370 triệu ($PhoBERT_{large}$) so với 560M ($XLM\text{-}R_{large}$).

2) Nhận xét

Kết quả cho thấy rằng $PhoBERT_{large}$ đạt được mức thấp hơn 0,9% điểm phân tích cú pháp phụ thuộc hơn $PhoBERT_{base}$. Một lý do có thể là lớp Transformer cuối cùng trong kiến trúc BERT có thể không phải là mã hóa thông tin tối ưu phong phú nhất về cấu trúc cú pháp (Hewitt và Manning, Năm 2019; Jawahar và cộng sự, 2019). Trong tương lai cần nghiên cứu lớp Transformer của PhoBERT chứa thông tin cú pháp phong phú hơn bằng cách đánh giá hiệu suất phân tích cú pháp tiếng Việt từ mỗi lớp.

Việc sử dụng nhiều dữ liệu trước khi huấn luyện hơn có thể nâng cao chất lượng của ngôn ngữ được huấn luyện trước mô hình (Liu và cộng sự, 2019). Vì vậy, không có gì đáng ngạc nhiên rằng PhoBERT giúp tạo ra hiệu suất tốt hơn ETNLP trên NER và BERT đa ngôn ngữ và XLMMLM + TLM trên NLI (ở đây, PhoBERT sử dụng 20GB văn bản tiếng Việt trong khi những mô hình đó sử dụng kho dữ liệu Wikipedia tiếng Việt 1GB).

Theo phương pháp tinh chỉnh đối với PhoBERT, chúng ta đã tinh chỉnh cẩn thận XLM-R cho các nhiệm vụ gán nhãn từ loại tiếng Việt, phân tích cú pháp Phụ thuộc và NER . Bảng 2 và 3 cho thấy rằng PhoBERT cũng hoạt động tốt hơn XLM-R trên ba nhiệm vụ cấp từ này. Cần lưu ý rằng XLM-R sử dụng kho dữ liệu huấn luyện trước 2,5TB chứa 137GB văn bản tiếng Việt (tức là khoảng 137/20 ≈ gấp 7 lần so với huấn luyện trước của ngữ liệu trong PhoBERT. PhoBERT thực hiện phân đoạn từ tiếng Việt để phân đoạn các câu ở cấp độ âm tiết thành các mã thông báo từ trước khi áp dụng BPE để phân đoạn các câu được phân đoạn từ thành các đơn vị từ khóa con, trong khi XLM-R trực tiếp áp dụng BPE cho các câu luyện trước tiếng Việt ở cấp độ âm tiết. Điều này xác nhận lại rằng các mô hình dành riêng cho ngôn ngữ cụ thể vẫn hoạt động tốt hơn đa ngôn ngữ (Martin và cộng sự, 2020).

2.2.5. Kết luận

Có thể thấy rằng PhoBERT – mô hình ngôn ngữ đơn ngữ quy mô lớn đầu tiên được huấn luyện đào tạo dành riêng cho Tiếng Việt - hoạt động tốt hơn so với sản phẩm tốt nhất gần đây mô hình đa ngôn ngữ XLM-R và giúp diễn giải SOTA thực hiện bốn nhiệm vụ NLP sau của Việt Nam là Gán nhãn từ loại, Sự phụ thuộc phân tích cú pháp, NER và NLI. Bằng cách phát hành công khai các mô hình PhoBERT, hy vọng rằng chúng có thể thúc đẩy các nghiên cứu và ứng dụng trong tương lai của NLP Việt Nam.

2.3. Ứng dụng của PhoBert

Trong Tiếng Việt thì chúng ta có thể ứng dụng BERT trong một số nhiệm vụ như:

- Tìm từ đồng nghĩa, trái nghĩa, cùng nhóm dựa trên khoảng cách của từ trong không gian biểu diễn đa chiều.
- Xây dựng các véc tơ nhúng cho các nhiệm vụ NLP như sentiment analysis, phân loại văn bản, nhận dạng thực thể, gắn nhãn từ loại, huấn luyện chatbot.
- Gợi ý từ khóa tìm kiếm trong các hệ thống search.
- Xây dựng các ứng dụng seq2seq như robot viết báo, tóm tắt văn bản, sinh câu ngẫu nhiên với ý nghĩa tương đồng.

CHƯƠNG 3. ỨNG DỤNG PHOBERT VÀO BÀI TOÁN PHÂN TÍCH QUAN ĐIỂM BÌNH LUẬN TIẾNG VIỆT

3.1. Phát biểu bài toán

Phân tích tình cảm (Sentiment Analysis) hay khai phá quan điểm (Opinion Mining) người dùng là lĩnh vực đã và đang thu hút được sự quan tâm của cộng đồng các nhà nghiên cứu cũng như các nhà phát triển ứng dụng. Cùng với sự phát triển của mạng máy tính toàn cầu và các thiết bị di động, người dùng đã tạo ra một lượng dữ liệu đánh giá khổng lồ trong quá trình họ tương tác trên các trang mạng xã hội, các trang diễn đàn, các trang đánh giá sản phẩm, v.v. Người dùng hay nhà sản xuất thường muốn biết sản phẩm hay dịch vụ mà họ quan tâm được đánh giá là tích cực hay tiêu cự để quyết định lựa chọn hay sản xuất nó. Do đó, việc khai thác các thông tin hữu ích từ dữ liệu đã được bình luận trên mạng sẽ giúp họ nắm được xu thế đang được đánh giá, bình luận hay thể hiện tình cảm về các sản phẩm, dịch vụ, sự kiện, v.v. là khen hay chê và được thể hiện như thế nào.

Phân tích cảm xúc (Sentiment analysis) là nhằm phát hiện ra thái độ mang tính lâu dài, màu sắc tình cảm, khuynh hướng niềm tin vào các đối tượng hay người nào đó.

Các vấn đề xung quanh việc phân tích cảm xúc:

- Nguồn gốc của cảm xúc.
- Mục tiêu của cảm xúc.
- Các loại cảm xúc: thích, yêu, ghét, đánh giá, mong mỏi...
- Về mức độ cảm xúc: tích cực, tiêu cực, trung tính.
- Văn bản hàm chứa cảm xúc: một câu hoặc một đoạn văn bản.

Bài toán phân tích cảm xúc thuộc dạng bài toán phân tích ngữ nghĩa văn bản. Vì vậy, ta cần phải xây dựng một mô hình để hiểu được ý nghĩa của câu văn, đoạn văn để quyết định xem câu văn đó hoặc đoạn văn đó mang màu sắc cảm xúc chủ đạo nào.

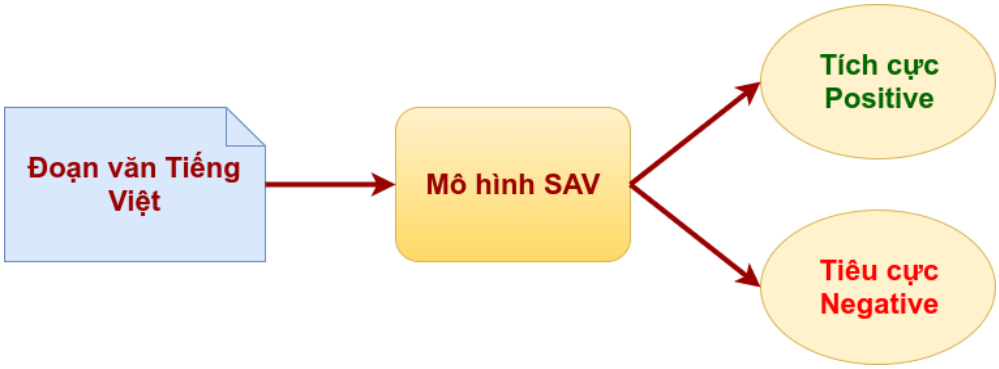
Phát biểu theo góc nhìn của máy học (Machine Learning) thì phân tích cảm xúc là bài toán phân lớp cảm xúc dựa trên văn bản ngôn ngữ tự nhiên. Đầu vào của bài toán là một câu hay một đoạn văn bản, còn đầu ra là các giá trị xác suất (điểm số) của N lớp cảm xúc mà ta cần xác định.

Trong loại bài toán phân tích cảm xúc được phân thành các bài toán có độ khó khác nhau như sau:

- Đơn giản: Phân tích cảm xúc (thái độ) trong văn bản thành 2 lớp: tích cực (positive) và tiêu cực (negative).
- Phức tạp hơn: Xếp hạng cảm xúc (thái độ) trong văn bản từ 1 đến 5.
- Khó: Phát hiện mục tiêu, nguồn gốc của cảm xúc (thái độ) hoặc các loại cảm xúc (thái độ) phức tạp.

Hiện tại thì cộng đồng khoa học mới chỉ giải quyết tốt bài toán phân tích cảm xúc ở cấp độ đơn giản, tức là phân tích cảm xúc với 2 lớp cảm xúc tiêu cực và tích cực với độ chính xác hơn 85%.

Vì vậy, bài toán phân tích cảm xúc trong Tiếng Việt trình bày trong bài viết này là kết quả của nghiên cứu phân tích cảm xúc văn bản Tiếng Việt với 2 lớp cảm xúc là: tiêu cực (negative) và tích cực (positive). Sơ đồ phân tích cảm xúc như sau:



Hình 9. Sơ đồ phân tích cảm xúc

Đầu vào của mô hình xử lý Sentiment Analysis Vietnamese (SAV) là một đoạn văn Tiếng Việt, đầu ra là 2 giá trị xác suất mà đoạn văn đầu vào thuộc về lớp cảm xúc: tiêu cực (negative) hay tích cực (positive).

Việc phân tích cảm xúc trong văn bản được ứng dụng trong hàng loạt các vấn đề như: Quản trị thương hiệu doanh nghiệp, thương hiệu sản phẩm, quản trị quan hệ khách hàng, khảo sát ý kiến xã hội học, phân tích trạng thái tâm lý con người...

Chúng ta đang sống trong kỷ nguyên số, đặc biệt những năm gần đây nổi lên với mạng xã hội, với hàng triệu người dùng trên thế giới, với lượng thông tin nội dung được người

dùng tạo ra hằng ngày cực kỳ lớn, với đa dạng các hình thức như dòng trạng thái, hình ảnh, video. Mạng xã hội có những đặc điểm là: thông tin do người dùng tạo ra, mang tính cá nhân cho nên chất lượng nội dung hay tính đúng đắn, xác thực là tương đối; một thông tin mới được tạo lại có sức lan tỏa nhanh đến đông đảo các người dùng khác, so với các kênh thông tin truyền thống như truyền hình, truyền thành, báo chí, diễn đàn, blog...

Điều này đặt ra cho các doanh nghiệp lớn giải quyết bài toán quản trị thương hiệu doanh nghiệp, quản trị thương hiệu sản phẩm trước các dư luận không tốt trên mạng xã hội rất khó khăn, cả về nguồn xuất phát thông tin, cả về khối lượng thông tin cần xử lý. Chưa kể việc các đối thủ cạnh tranh trên thương trường lợi dụng mạng xã hội để cố ý tạo các thông tin bất lợi cho nhau.

Một ví dụ cụ thể tại Việt Nam là vụ việc "con ruồi trong chai number one" của doanh nghiệp Tân Hiệp Phát gần đây, gây ảnh hưởng xấu đến hình ảnh của Tân Hiệp Phát và việc tiêu thụ sản phẩm nước uống tăng lực number one của doanh nghiệp này. Xét về luật pháp thì Tân Hiệp Phát là đúng nhưng không khéo léo trong việc xử lý quan hệ với khách hàng, gây bất bình trên mạng xã hội, đó lại là bài toán quản trị quan hệ với khách hàng mà doanh nghiệp phải giải quyết. Mà ai biết được các thông tin bất lợi về Tân Hiệp Phát này có được thúc đẩy bởi các đối thủ cạnh tranh hay không? Điều này đòi hỏi phải có một công cụ hỗ trợ đắc lực, mà chỉ có áp dụng công nghệ thông tin mới giải quyết được, chứ không lực lượng con người nào có thể làm xuể.

Rút kinh nghiệm từ Tân Hiệp Phát thì các doanh nghiệp lớn của Việt Nam hiện nay cũng đã đặt hàng các doanh nghiệp công nghệ thông tin giải quyết vấn đề này. Giải pháp công nghệ hiện nay được gọi là "lắng nghe mạng xã hội", tức là các doanh nghiệp CNTT mua các dữ liệu thời gian thực (real time) từ các công ty mạng xã hội về để xử lý các thông tin liên quan đến doanh nghiệp hay các sản phẩm mà doanh nghiệp đó kinh doanh, nhằm phát hiện và ngăn chặn sớm sự lan rộng các thông tin bất lợi trên mạng xã hội, có hình thức đính chính phản hồi đến các khách hàng của mình, đồng thời thương lượng, ngăn chặn tận gốc những người tạo ra các nội dung đó. Điều cốt yếu của giải pháp này chính là phân tích cảm xúc của các dòng trạng thái trên mạng xã hội nhằm lọc ra các thông tin bất lợi để xử lý.

Bài toán phân loại quan điểm bình luận Tiếng Việt

Phân loại quan điểm bình luận Tiếng Việt được coi là nhiệm vụ quan trọng trong việc phân tích tình cảm nhằm xác định một tài liệu hay một câu có chứa yếu tố tích cực hay tiêu cực. Kết quả của bước này sẽ là cho các phân tích tiếp theo như phân loại tình cảm, phân tích tình cảm theo khía cạnh hay tóm tắt quan điểm. Bài toán được phát biểu như sau:

- Đầu vào : Cho một tài liệu/Câu
- Đầu ra : Tài liệu/Câu là tích cực hoặc tiêu cực

Ví dụ 1: Về loại tài liệu chứa yếu tố tích cực: " Chiếc Iphone 13 mới ra này có cấu hình tốt thật đấy , màn hình rộng , chụp ảnh đẹp và dung lượng bộ nhớ cao , tuyệt vời "

Ví dụ 2: Về loại tài liệu chứa yếu tố tiêu cực: " Chiếc Iphone 13 mới ra này đắt quá với lại chẳng có gì khác biệt so với Iphone 12 gì cả "

Trong ví dụ 1 , ta có thể thấy người dùng đưa ra quan điểm bình luận mang hướng tích cực về chiếc Iphone 13 qua các thông tin như: " cấu hình tốt " , " màn hình rộng " , " chụp ảnh đẹp " , " dung lượng bộ nhớ cao " . Trong khi đó , ở ví dụ 2 , ta thấy rằng quan điểm bình luận mang hướng tiêu cực về chiêc Iphone 13 qua các thông tin như : " đắt quá " , " chẳng có gì khác biệt "

3.2. Dữ liệu và Công cụ, môi trường thực nghiệm:

3.2.1. Dữ liệu

Dữ liệu được lấy từ cuộc thi Phân tích quan điểm luận Tiếng Việt. Dữ liệu được cung cấp bởi aivivn.com, trong đó bộ dữ liệu huấn luyện gồm 16087 câu bình luận đã được gắn nhãn, bộ dữ liệu đánh giá gồm 10981 câu bình luận

Khi tải về , ta thấy 2 file gồm test.crash và train.crash lần lượt là dữ liệu đánh giá và dữ liệu huấn luyện cho cuộc thi.

Bộ dữ liệu gồm bình luận và nhãn của bình luận. Bình luận tích cực được gắn nhãn 0, còn bình luận tiêu cực được gắn nhãn 1.

Ví dụ bình luận tích cực: "sản phẩm đẹp quá", "giao hàng hơi trễ 1 chút, nhưng sản phẩm tuyệt vời"

Ví dụ bình luận tiêu cực: “ quá thất vọng”, “sản phẩm quá đắt mà chất lượng bình thường”

3.2.2. Công cụ và môi trường thực nghiệm:

❖ **Công cụ**

Ngôn ngữ lập trình Python

Python được Guido van Rossum phát triển vào cuối những năm tám mươi và đầu những năm chín mươi tại Viện nghiên cứu quốc gia về toán học và khoa học máy tính ở Hà Lan.

Python là ngôn ngữ lập trình hướng đối tượng, cấp cao, mạnh mẽ, được tạo ra bởi Guido van Rossum. Nó dễ dàng để tìm hiểu và đang nổi lên như một trong những ngôn ngữ lập trình nhập môn tốt nhất cho người lần đầu tiếp xúc với ngôn ngữ lập trình. Python hoàn toàn tạo kiểu động và sử dụng cơ chế cấp phát bộ nhớ tự động. Python có cấu trúc dữ liệu cấp cao mạnh mẽ và cách tiếp cận đơn giản nhưng hiệu quả đối với lập trình hướng đối tượng. Cú pháp lệnh của Python là điểm cộng vô cùng lớn vì sự rõ ràng, dễ hiểu và cách gõ linh động làm cho nó nhanh chóng trở thành một ngôn ngữ lý tưởng để viết script và phát triển ứng dụng trong nhiều lĩnh vực, ở hầu hết các nền tảng.

Tính năng chính của Python :

- **Ngôn ngữ lập trình đơn giản, dễ học:** Python có cú pháp rất đơn giản, rõ ràng. Nó dễ đọc và viết hơn rất nhiều khi so sánh với những ngôn ngữ lập trình khác như C++, Java, C#. Python làm cho việc lập trình trở nên thú vị, cho phép bạn tập trung vào những giải pháp chứ không phải cú pháp.
- **Miễn phí, mã nguồn mở:** Bạn có thể tự do sử dụng và phân phối Python, thậm chí là dùng nó cho mục đích thương mại. Vì là mã nguồn mở, bạn không những có thể sử dụng các phần mềm, chương trình được viết trong Python mà còn có thể thay đổi mã nguồn của nó. Python có một cộng đồng rộng lớn, không ngừng cải thiện nó mỗi lần cập nhật.
- **Khả năng di chuyển:** Các chương trình Python có thể di chuyển từ nền tảng này sang nền tảng khác và chạy nó mà không có bất kỳ thay đổi nào. Nó chạy liền mạch trên hầu hết tất cả các nền tảng như Windows, macOS, Linux

- **Khả năng mở rộng và có thể nhúng:** Giả sử một ứng dụng đòi hỏi sự phức tạp rất lớn, bạn có thể dễ dàng kết hợp các phần code bằng C, C++ và những ngôn ngữ khác (có thể gọi được từ C) vào code Python. Điều này sẽ cung cấp cho ứng dụng của bạn những tính năng tốt hơn cũng như khả năng scripting mà những ngôn ngữ lập trình khác khó có thể làm được.
- **Ngôn ngữ thông dịch cấp cao:** Không giống như C/C++, với Python, bạn không phải lo lắng những nhiệm vụ khó khăn như quản lý bộ nhớ, dọn dẹp những dữ liệu vô nghĩa,... Khi chạy code Python, nó sẽ tự động chuyển đổi code sang ngôn ngữ máy tính có thể hiểu. Bạn không cần lo lắng về bất kỳ hoạt động ở cấp thấp nào.
- **Thư viện tiêu chuẩn lớn để giải quyết những nhiệm vụ phổ biến:** Python có một số lượng lớn thư viện tiêu chuẩn giúp cho công việc lập trình của bạn trở nên dễ thở hơn rất nhiều, đơn giản vì không phải tự viết tất cả code. Ví dụ: Bạn cần kết nối cơ sở dữ liệu MySQL trên Web server? Bạn có thể nhập thư viện MySQLdb và sử dụng nó. Những thư viện này được kiểm tra kỹ lưỡng và được sử dụng bởi hàng trăm người. Vì vậy, bạn có thể chắc chắn rằng nó sẽ không làm hỏng code hay ứng dụng của mình.
- **Hướng đối tượng:** Mọi thứ trong Python đều là hướng đối tượng. Lập trình hướng đối tượng (OOP) giúp giải quyết những vấn đề phức tạp một cách trực quan. Với OOP, bạn có thể phân chia những vấn đề phức tạp thành những tập nhỏ hơn bằng cách tạo ra các đối tượng.

Thư viện mã nguồn mở Tensorflow

Tensorflow là một thư viện mã nguồn mở cung cấp khả năng xử lí tính toán số học dựa trên biểu đồ mô tả sự thay đổi của dữ liệu, trong đó các node là các phép tính toán học còn các cạnh biểu thị luồng dữ liệu. Trong tesorflow có một vài khái niệm cơ bản sau.

Tensor là cấu trúc dữ liệu trong tensorflow đại diện cho tất cả các loại dữ liệu. Nói cách khác, tất cả các kiểu dữ liệu khi đưa vào trong tensorflow thì đều được gọi là Tensor. Vậy nên có thể hiểu được Tensorflow là một thư viện mô tả, điều chỉnh dòng chảy của các Tensor. Tensor có 3 thuộc tính cơ bản là rank, shape và type:

Rank là số bậc của tensor. Ví dụ Tensor = [1] thì có rank = 1, Tensor = [[3,4],[5,6]] thì sẽ có rank = 2. Việc phân rank này khá quan trọng vì nó đồng thời cũng giúp phân loại dữ liệu của Tensor. Khi các rank đặc biệt cụ thể, Tensor có những tên gọi riêng như sau:

- Scalar: Khi Tensor có rank bằng 0.
- Véc tơ: Véc tơ là một Tensor rank 1.
- Matrix: Đây là một Tensor rank 2 hay mảng hai chiều theo khái niệm của Python.
- N-Tensor: Khi rank của Tensor tăng lên lớn hơn 2, chúng được gọi chung là N-Tensor.
- Shape là chiều của tensor. Vi dụ Tensor = [[[1,1,1], [178,62,74]]] sẽ có Shape = (1,2,3), Tensor = [[1,1,1], [178,62,74]] sẽ có Shape = (2,3).

Type kiểu dữ liệu của các elements trong Tensor. Vì một Tensor chỉ có duy nhất một thuộc tính Type nên từ đó cũng suy ra là chỉ có duy nhất một kiểu Type duy nhất cho toàn bộ các elements có trong Tensor hiện tại.

Thư viện Transformers : Là một project của huggingface hỗ trợ huấn luyện các model dựa trên kiến trúc transformer như BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, T5, CTRL,… phục vụ cho các tác vụ xử lý ngôn ngữ tự nhiên trên cả nền tảng pytorch và tensorflow.

Thư viện fastBPE : Là package hỗ trợ tokenize từ (word) thành các từ phụ (subwords) theo phương pháp mới nhất được áp dụng cho các pretrain model xử lý ngôn ngữ tự nhiên hiện đại như BERT và các biến thể của BERT.

Thư viện fairseq : Là project của facebook chuyên hỗ trợ các nghiên cứu và dự án liên quan đến model seq2seq

Thư viện VnCoreNLP : Là một package xử lý ngôn ngữ tự nhiên trong Tiếng Việt, hỗ trợ tokenize và các tác vụ xử lý ngôn ngữ khác.

PhoBERT đã được huấn luyện trước.

❖ **Môi trường thực nghiệm:**

- Máy tính Chip: Intel(R) Dual Core I5(R) @ 4300U, Ram: 16.00 GB.
- Hệ điều hành Ubuntu 20.04
- Công cụ lập trình: Python 3.9.

3.3. Các bước thực hiện

3.3.1. Cài đặt các thư viện cần thiết

```
pip install transformers

pip install fastBPE

pip install fairseq
```

3.3.2. Cài đặt thư viện vncorenlp

```
# Install the vncorenlp python wrapper
pip install vncorenlp

# Download VnCoreNLP-1.1.1.jar & its word segmentation component (i.e. RDRSegmenter)
mkdir -p vncorenlp/models/wordsegmenter
wget https://raw.githubusercontent.com/vncorenlp/VnCoreNLP/master/VnCoreNLP-1.1.1.jar
wget https://raw.githubusercontent.com/vncorenlp/VnCoreNLP/master/models/wordsegmenter/vi-vocab
wget https://raw.githubusercontent.com/vncorenlp/VnCoreNLP/master/models/wordsegmenter/wordsegmenter.rdr
mv VnCoreNLP-1.1.1.jar vncorenlp/
mv vi-vocab vncorenlp/models/wordsegmenter/
mv wordsegmenter.rdr vncorenlp/models/wordsegmenter/
```

Để chắc chắn cài đặt vncorenlp thành công , ta có thể sử dụng nó để tách từ một câu đơn giản theo cách dưới đây:

```
from vncorenlp import VnCoreNLP
rdrsegmenter = VnCoreNLP("/Absolute-path-to/vncorenlp/VnCoreNLP-1.1.1.jar", annotators="wseg", max_heap_size='-Xmx500m')
# rdrsegmenter = VnCoreNLP("/content/drive/My Drive/BERT/SA/vncorenlp/VnCoreNLP-1.1.1.jar", annotators="wseg", max_heap_size='-Xmx500m')

text = "Đại học Bách Khoa Hà Nội."

word_segmented_text = rdrsegmenter.tokenize(text)
print(word_segmented_text)
```

Kết quả thu được là:

[['Đại_học', 'Bách_Khoa', 'Hà_Nội', '.']]

3.3.3. Tải về bộ dữ liệu huấn luyện từ trang chủ cuộc thi của AIVIVN và pre-trained của PhoBERT

```
wget https://public.vinai.io/PhoBERT_base_transformers.tar.gz
tar -xzvf PhoBERT_base_transformers.tar.gz
```

Sau đó ta load model và bpe :

```
from fairseq.data.encoders.fastbpe import fastBPE
from fairseq.data import Dictionary
import argparse

parser = argparse.ArgumentParser()
parser.add_argument('--bpe-codes',
    default="/content/drive/My
Drive/BERT/SA/PhoBERT_base_transformers/bpe.codes",
    required=False,
    type=str,
    help='path to fastBPE BPE'
)
args, unknown = parser.parse_known_args()
bpe = fastBPE(args)

# Load the dictionary
vocab = Dictionary()
vocab.add_from_file("/content/drive/My
Drive/BERT/SA/PhoBERT_base_transformers/dict.txt")
```

3.3.4. Tải về dữ liệu của cuộc thi Phân tích sắc thái bình luận

Ta tiến hành đọc dữ liệu và dữ liệu test

```
import re

train_path = '/content/drive/My Drive/BERT/SA/train.crash'
test_path = '/content/drive/My Drive/BERT/SA/test.crash'

train_id, train_text, train_label = [], [], []
test_id, test_text = [], []

with open(train_path, 'r') as f_r:
    data = f_r.read().strip()
```

```
    data = re.findall('train_[\s\S]+?\"\n[01]\n\n', data)

    for sample in data:
        splits = sample.strip().split('\n')

        id = splits[0]
        label = int(splits[-1])
        text = ' '.join(splits[1:-1])[1:-1]
        text = rdrsegmenter.tokenize(text)
        text = ' '.join([' '.join(x) for x in text])

        train_id.append(id)
        train_text.append(text)
        train_label.append(label)


with open(test_path, 'r') as f_r:
    data = f_r.read().strip()
    data = re.findall('train_[\s\S]+?\"\n[01]\n\n', data)

    for sample in data:
        splits = sample.strip().split('\n')

        id = splits[0]
        text = ' '.join(splits[1:])[1:-1]
        text = rdrsegmenter.tokenize(text)
        text = ' '.join([' '.join(x) for x in text])

        test_id.append(id)
        test_text.append(text)
```

3.3.5. Tách dữ liệu ra thành 2 tập train và validation theo tỉ lệ 90:10

```
from sklearn.model_selection import train_test_split

train_sents, val_sents, train_labels, val_labels =
train_test_split(train_text, train_labels, test_size=0.1)
```

Sử dụng bpe đã load ở trên để đưa text đầu vào dưới dạng subword và ánh xạ các subword này về dạng index trong từ điển

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
MAX_LEN = 125

train_ids = []
```

```
for sent in train_sents:
    subwords = '<s> ' + bpe.encode(sent) + ' </s>'
    encoded_sent = vocab.encode_line(subwords, append_eos=True,
add_if_not_exist=False).long().tolist()
    train_ids.append(encoded_sent)

val_ids = []
for sent in val_sents:
    subwords = '<s> ' + bpe.encode(sent) + ' </s>'
    encoded_sent = vocab.encode_line(subwords, append_eos=True,
add_if_not_exist=False).long().tolist()
    val_ids.append(encoded_sent)

train_ids = pad_sequences(train_ids, maxlen=MAX_LEN, dtype="long",
value=0, truncating="post", padding="post")
val_ids = pad_sequences(val_ids, maxlen=MAX_LEN, dtype="long", value=0,
truncating="post", padding="post")
```

3.3.6. Tạo một mask gồm các giá trị `0`, `1` để làm đầu vào cho thư viện transformers

```
train_masks = []
for sent in train_ids:
    mask = [int(token_id > 0) for token_id in sent]
    train_masks.append(mask)

val_masks = []
for sent in val_ids:
    mask = [int(token_id > 0) for token_id in sent]


    val_masks.append(mask)
```

Sử dụng DataLoader của torch để tạo dataloader

```
from torch.utils.data import TensorDataset, DataLoader, RandomSampler,
SequentialSampler
import torch

train_inputs = torch.tensor(train_ids)
val_inputs = torch.tensor(val_ids)
train_labels = torch.tensor(train_labels)
val_labels = torch.tensor(val_labels)
train_masks = torch.tensor(train_masks)
val_masks = torch.tensor(val_masks)

train_data = TensorDataset(train_inputs, train_masks, train_labels)
train_sampler = SequentialSampler(train_data)
```

```
train_dataloader = DataLoader(train_data, sampler=train_sampler,
batch_size=32)

val_data = TensorDataset(val_inputs, val_masks, val_labels)
val_sampler = SequentialSampler(val_data)
val_dataloader = DataLoader(val_data, sampler=val_sampler, batch_size=32)
```

Load model PhoBert

```
from transformers import RobertaForSequenceClassification, RobertaConfig,
AdamW

config = RobertaConfig.from_pretrained(
    "/content/drive/My
Drive/BERT/SA/PhoBERT_base_transformers/config.json", from_tf=False,
num_labels = 2, output_hidden_states=False,
)
BERT_SA = BertForSequenceClassification.from_pretrained(
    "/content/drive/My Drive/BERT/SA/PhoBERT_base_transformers/model.bin",
    config=config
)
```

3.3.7. Huấn luyện mô hình

```
import random
from tqdm import tqdm_notebook
device = 'cpu'
epochs = 10

param_optimizer = list(BERT_SA.named_parameters())
no_decay = ['bias', 'LayerNorm.bias', 'LayerNorm.weight']
optimizer_grouped_parameters = [
    {'params': [p for n, p in param_optimizer if not any(nd in n for nd in
no_decay)], 'weight_decay': 0.01},
    {'params': [p for n, p in param_optimizer if any(nd in n for nd in
no_decay)], 'weight_decay': 0.0}
]

optimizer = AdamW(optimizer_grouped_parameters, lr=1e-5,
correct_bias=False)


for epoch_i in range(0, epochs):
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    print('Training...')

    total_loss = 0
    BERT_SA.train()
```

```
    train_accuracy = 0
    nb_train_steps = 0
    train_f1 = 0

    for step, batch in tqdm_notebook(enumerate(train_dataloader)):
        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)

        BERT_SA.zero_grad()
        outputs = BERT_SA(b_input_ids,
            token_type_ids=None,
            attention_mask=b_input_mask,
            labels=b_labels)
        loss = outputs[0]
        total_loss += loss.item()

        logits = outputs[1].detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()
        tmp_train_accuracy, tmp_train_f1 = flat_accuracy(logits,
label_ids)
        train_accuracy += tmp_train_accuracy
        train_f1 += tmp_train_f1
        nb_train_steps += 1

        loss.backward()
        torch.nn.utils.clip_grad_norm_(BERT_SA.parameters(), 1.0)
        optimizer.step()

    avg_train_loss = total_loss / len(train_dataloader)
    print(" Accuracy: {0:.4f}".format(train_accuracy/nb_train_steps))
    print(" F1 score: {0:.4f}".format(train_f1/nb_train_steps))
    print(" Average training loss: {0:.4f}".format(avg_train_loss))

    print("Running Validation...")
    BERT_SA.eval()
    eval_loss, eval_accuracy = 0, 0
    nb_eval_steps, nb_eval_examples = 0, 0
    eval_f1 = 0
    for batch in tqdm_notebook(val_dataloader):

        batch = tuple(t.to(device) for t in batch)

        b_input_ids, b_input_mask, b_labels = batch

        with torch.no_grad():
            outputs = BERT_SA(b_input_ids,
```

```
            token_type_ids=None,
            attention_mask=b_input_mask)
            logits = outputs[0]
            logits = logits.detach().cpu().numpy()
            label_ids = b_labels.to('cpu').numpy()

            tmp_eval_accuracy, tmp_eval_f1 = flat_accuracy(logits,
label_ids)

            eval_accuracy += tmp_eval_accuracy
            eval_f1 += tmp_eval_f1
            nb_eval_steps += 1
    print(" Accuracy: {0:.4f}".format(eval_accuracy/nb_eval_steps))
    print(" F1 score: {0:.4f}".format(eval_f1/nb_eval_steps))
print("Training complete!")
```

Kết quả thực hiện

```
======== Epoch 1 / 10 ========
Training...
 Accuracy: 0.8370
 F1 score: 0.8262
 Average training loss: 0.3511
Running Validation...
 Accuracy: 0.9118
 F1 score: 0.9087
======== Epoch 2 / 10 ========
Training...
 Accuracy: 0.9071
 F1 score: 0.9025
 Average training loss: 0.2348
Running Validation...
 Accuracy: 0.9167
 F1 score: 0.9131
 ======== Epoch 3 / 10 ========
Training...
 Accuracy: 0.9261
 F1 score: 0.9223
 Average training loss: 0.1954
Running Validation...
 Accuracy: 0.9148
 F1 score: 0.9113
 ======== Epoch 4 / 10 ========
Training...
 Accuracy: 0.9390
 F1 score: 0.9358
 Average training loss: 0.1662
Running Validation...
```

```
 Accuracy: 0.9167
 F1 score: 0.9138
======== Epoch 5 / 10 ========
Training...
 Accuracy: 0.9510
 F1 score: 0.9482
 Average training loss: 0.1443
Running Validation...
 Accuracy: 0.9148
 F1 score: 0.9113
======== Epoch 6 / 10 ========
Training...
 Accuracy: 0.9587
 F1 score: 0.9566
 Average training loss: 0.1271
Running Validation...
 Accuracy: 0.9167
 F1 score: 0.9127
======== Epoch 7 / 10 ========
Training...
 Accuracy: 0.9645
 F1 score: 0.9625
 Average training loss: 0.1099
Running Validation...
 Accuracy: 0.9142
 F1 score: 0.9103
```

KẾT LUẬN

Trong thời gian làm đồ án này, bằng kiến thức đã được học trong trường cùng với sự hướng dẫn tận tình của các thầy cô và bạn bè, đã giúp em vận dụng và hoàn thành đề tài và đồ án tốt nghiệp trong thời gian quy định. Trong quá trình thực hiện đề tài em đã học hỏi và tìm hiểu được những khái niệm cơ bản về xử lý ngôn ngữ tự nhiên, mô hình ngôn ngữ BERT, PhoBer, ngôn ngữ lập trình Python, sử dụng các thư viện trong Tensorflow. Đồ án cũng đã cài đặt thử nghiệm bài toán phân tích quan điểm bình luận Tiếng Việt dựa trên mô hình ngôn ngữ PHoBERT đã được huấn luyện trước và công cụ phân loại văn bản trong Keras. Dữ liệu cho bài toán thực nghiệm là các bình luận được thu thập từ cuộc thi Phân tích quan điểm bình luận Tiếng Việt. Bộ dữ liệu gồm 16087 câu bình luận đã được gắn nhãn, bộ dữ liệu đánh giá gồm 10981 câu bình luận được sử dụng để đánh giá chất lượng của mô hình học. Bộ dữ liệu gồm bình luận và nhãn của bình luận. Bình luận tích cực được gắn nhãn 0, còn bình luận tiêu cực được gắn nhãn 1.

Trong thời gian 12 tuần thực hiện đề tài, do kiến thức còn hạn hẹp, nên đồ án tốt nghiệp của em không thể tránh khỏi những thiếu sót, em mong sẽ nhận được những đóng góp của các thầy cô và các bạn để đồ án của em trở lên hoàn thiện hơn.

TÀI LIỆU THAM KHẢO

1. Python Machine Learning By Example by Yuxi Liu, 2017.
2. Neural Network Embeddings Explained by Will Koehrsen, 2018.
3. Deep Learning: Recurrent Neural Networks in Python: LSTM, GRU, and more RNN machine learning architectures in Python and Theano, 2016
4. Andrew Ng, Machine Learning course, 2020
5. https://viblo.asia/
6. Christopher Olah (2015), Understanding LSTM networks in Colah's blog
7. RoBERTa: A Robustly Optimized BERT Pretraining Approach by Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov.
8. Attention Is All You Need by Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin
9. How to Fine-Tune BERT for Text Classification? by Chi Sun, Xipeng Qiu, Yige Xu, Xuanjing Huang
10. Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-Training with Whole Word Masking for Chinese BERT. arXiv preprint, arXiv:1906.08101
11. Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. 2019. BERTje: A Dutch BERT Model. arXiv preprint, arXiv:1912.09582.
12. Byte-Pair encoding (BPE) methods (Sennrich et al., 2016; Kudo and Richardson, 2018)
13. Xuan-Son Vu, Thanh Vu, Son Tran, and Lili Jiang. 2019. ETNLP: A visual-aided systematic approach to select pre-trained embeddings for a downstream task. In Proceedings of RANLP, pages 1285–1294.
14. Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric Villemonte de la Clergerie, Djamé Seddah, and Benoˆıt Sagot. 2020. CamemBERT: a Tasty French Language Model. In Proceedings of ACL, pages 7203– 7219.

15. Dat Quoc Nguyen, Dai Quoc Nguyen, Son Bao Pham, Phuong-Thai Nguyen, and Minh Le Nguyen. 2014b. From Treebank Conversion to Automatic Dependency Parsing for Vietnamese. In Proceedings of NLDB
16. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pier-ric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing.
17. Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2018. VnCoreNLP: A Vietnamese Natural Language Processing Toolkit. In Proceedings of NAACL: Demonstrations
18. Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In Proceedings of NAACL-HLT 2019: Demonstrations
19. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of NAACL, pages 4171– 4186
20. Alexis Conneau and Guillaume Lample. 2019. Crosslingual Language Model Pretraining. In Proceedings of NeurIPS, pages 7059–7069.
21. Dat Quoc Nguyen, Dai Quoc Nguyen, Thanh Vu, Mark Dras, and Mark Johnson. 2018. A Fast and Accurate Vietnamese Word Segmenter. In Proceedings of LREC, pages 2582–2587.
22. Huyen Nguyen, Quyen Ngo, Luong Vu, Vu Tran, and Hien Nguyen. 2019a. VLSP Shared Task: Named Entity Recognition. Journal of Computer Science and Cybernetics, 34(4):283–294.
23. Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. arXiv preprint, arXiv:1910.03771.
24. Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A Fast, Extensible Toolkit for Sequence Modeling. In Proceedings of NAACL-HLT 2019: Demonstrations, pages 48–53.

25. Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint, arXiv:1907.11692.
26. Thanh Vu, Dat Quoc Nguyen, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2018. VnCoreNLP: A Vietnamese Natural Language Processing Toolkit. In Proceedings of NAACL: Demonstrations, pages 56–60.
27. Dat Quoc Nguyen, Thanh Vu, Dai Quoc Nguyen, Mark Dras, and Mark Johnson. 2017. From word segmentation to POS tagging for Vietnamese. In Proceedings of ALTA, pages 108–113.
28. Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNsCRF. In Proceedings of ACL, pages 1064–1074.
29. Alexis Conneau and Guillaume Lample. 2019. Crosslingual Language Model Pretraining. In Proceedings of NeurIPS, pages 7059–7069.