**PhoBERT:**

**Mô hình ngôn ngữ tiền huấn luyện cho Tiếng Việt**

Nguyễn Quốc Đạt, Nguyễn Tuấn Anh

VinAI Research, Việt Nam; NVIDIA, USA

Nơi phát hành: Hội nghị IJCAI   
(International Joint Conference on Artificial Intelligence) năm 2020.

# Abstract

## 1. Tiền huấn luyện (pre-trained)

Tiền huấn luyện là một phương pháp trong học sâu, trong đó một mô hình được huấn luyện trên một tập dữ liệu lớn để học các đặc trưng ngôn ngữ tổng quát trước khi được fine-tune trên các tác vụ cụ thể. Mô hình được huấn luyện sẽ biết cách ánh xạ từ ngữ liệu đầu vào sang các biểu diễn ngữ nghĩa trong không gian vector. Việc tiền huấn luyện này đã giúp cho các mô hình ngôn ngữ đạt được những thành tựu đáng kể trong nhiều tác vụ xử lý ngôn ngữ tự nhiên.

Việc tiền huấn luyện trên một tập dữ liệu lớn giúp cho mô hình hiểu được những kiến thức cơ bản và tổng quát trong dữ liệu, đồng thời giảm thiểu khả năng overfitting khi fine-tune trên tập dữ liệu nhỏ hơn.

## 2. Transformers Block

PhoBERT sử dụng Transformer block để xử lý ngôn ngữ tự nhiên. Mỗi Transformer block trong PhoBERT bao gồm hai phần chính:

Multi-Head Attention: khối này giúp mô hình chú ý đến các từ quan trọng nhất trong câu và tạo ra biểu diễn mới của các từ đó. Điều này cho phép mô hình tập trung vào các thông tin quan trọng nhất của câu để tạo ra một biểu diễn toàn cục cho cả câu.

Feed-Forward Neural Network: phần này được sử dụng để tính toán phức tạp trên các biểu diễn từ được tạo ra từ phần Multi-Head Attention. Điều này giúp mô hình học được các mối quan hệ giữa các từ trong câu và tạo ra các biểu diễn của câu phù hợp với tác vụ được yêu cầu, chẳng hạn như dịch máy hoặc phân loại văn bản. Feed-Forward Neural Network là một mạng thần kinh nhân tạo trong đó các kết nối giữa các nút không tạo thành một chu kỳ.

Mỗi Transformer block trong PhoBERT cũng sử dụng các kỹ thuật như residual connection và layer normalization để giúp mô hình học tốt hơn và tránh vấn đề mất gradient khi huấn luyện mạng nơ-ron sâu. Hai kỹ thuật giúp cho mô hình huấn luyện nhanh hội tụ hơn và trách mất mát thông tin trong quá trình huấn luyện mô hình. Ví dụ: thông tin của vị trí các từ được mã hóa.

## 3. XLM-R

### 3.1. Định nghĩa

XLM-R là viết tắt của "Cross-lingual Language Model - RoBERTa", là một mô hình học sâu pre-trained trên nhiều ngôn ngữ khác nhau. Mô hình này là sự kết hợp giữa hai phương pháp: pre-training ngôn ngữ đa ngữ và tối ưu hóa mô hình pre-training.

Trong quá trình pre-training, XLM-R sử dụng dữ liệu ngôn ngữ đa ngữ để học các đặc trưng ngôn ngữ chung. Điều này cho phép mô hình hiểu và tổng quát hóa được nhiều ngôn ngữ khác nhau. XLM-R cũng sử dụng RoBERTa, một mô hình pre-trained trên tiếng Anh, để tối ưu hóa quá trình pre-training. Việc sử dụng RoBERTa giúp cho XLM-R có thể học được nhiều đặc trưng ngôn ngữ hơn, đặc biệt là trong các vấn đề liên quan đến cú pháp và đồng nghĩa.

### 3.2. So sánh XLM-R với PhoBERT

XLM-R (Cross-lingual Language Model for Few-shot Learning) và PhoBERT đều là các mô hình ngôn ngữ tiên tiến được huấn luyện trước cho các ngôn ngữ khác nhau. Tuy nhiên, XLM-R được huấn luyện trên nhiều ngôn ngữ khác nhau, trong khi PhoBERT chỉ được huấn luyện trên tiếng Việt.

Một điểm khác biệt quan trọng khác giữa XLM-R và PhoBERT là PhoBERT đạt được hiệu suất tốt hơn trong nhiều tác vụ NLP cụ thể cho tiếng Việt, bao gồm phân loại thực thể định danh (Named-entity recognition), phân loại câu hỏi (Question answering), phân loại văn bản (Text classification) và phân tích ngữ nghĩa tự nhiên (Natural language inference).

Ngoài ra, PhoBERT cũng được cung cấp và tinh chỉnh cho ngôn ngữ Việt, trong khi XLM-R được cung cấp cho nhiều ngôn ngữ khác nhau. Tuy nhiên, XLM-R vẫn là một trong những mô hình tốt nhất cho các nhiệm vụ liên ngôn ngữ và được sử dụng rộng rãi trong cộng đồng NLP.

## 4. VietNamese-specific NLP task

### 4.1. Part-of-speech tagging (POS tagging) – Gán nhãn từ loại

Là tác vụ gán nhãn cho từng từ trong câu để xác định loại từ đó (tính từ, động từ, danh từ, ...). Tác vụ này có nhiều ứng dụng trong xử lý ngôn ngữ tự nhiên, chẳng hạn như phân tích cú pháp, phân tích ý định, dịch máy, ...

### 4.2. Dependency parsing – Phân tích cú pháp phụ thuộc

Là việc phân tích cú pháp của một câu, xác định mối quan hệ phụ thuộc giữa các từ trong câu. Dependency parsing giúp chúng ta hiểu được cấu trúc ngữ pháp của câu, đặc biệt là khi xử lý ngôn ngữ tự nhiên trong các ứng dụng như chatbot hoặc hỗ trợ tư vấn.

### 4.3. Named-entity recognition (NER) – Trích xuất thông tin thực thể

Là tác vụ nhận dạng và phân loại các thực thể trong văn bản, chẳng hạn như người, địa điểm, thời gian, tên tổ chức, tên sản phẩm, ... Tác vụ này có thể được sử dụng để tìm kiếm thông tin, phân tích tình hình thị trường, phân tích ý kiến, ... NER là một tác vụ quan trọng trong việc rút trích thông tin, phân loại văn bản và khai thác dữ liệu.

### 4.4. Natural language inference (NLI) – Suy luận ngôn ngữ tự nhiên

Đây là một nhiệm vụ quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên, trong đó mô hình máy tính được huấn luyện để xác định mối quan hệ giữa hai câu trong tiếng tự nhiên, bao gồm xác định liệu chúng có cùng ý nghĩa, ngược lại hay là không liên quan gì đến nhau. Tác vụ này có nhiều ứng dụng trong xử lý ngôn ngữ tự nhiên, chẳng hạn như hệ thống trả lời tự động, đánh giá nhận thức người dùng, dịch máy, ...

NLI phân loại một cặp câu trong đó có một câu được gọi là giả thuyết (hypothesis) và một câu là ngữ cảnh (premise) theo một trong ba loại quan hệ: tương đồng, mâu thuẫn hoặc trung lập. NLI là tác vụ quan trọng trong việc giải quyết các vấn đề liên quan đến đọc hiểu, phân tích ngôn ngữ tự nhiên và trả lời câu hỏi.

# 1 Introduction

## 1. BERT

### 1.1. Định nghĩa

BERT (Bidirectional Encoder Representations from Transformers) là một kiến trúc mô hình ngôn ngữ trên cơ sở học sâu (deep learning) cho các tác vụ xử lý ngôn ngữ tự nhiên (NLP), được giới thiệu bởi Google vào năm 2018. BERT có khả năng "hiểu" được nghĩa của các từ trong văn bản và đưa ra các dự đoán chính xác cho các tác vụ liên quan đến ngôn ngữ như phân loại văn bản, dịch máy, gợi ý từ khóa và phân tích cảm xúc. BERT được huấn luyện trên một lượng lớn dữ liệu văn bản không có giám sát trước khi được điều chỉnh trên các tác vụ cụ thể, cho phép nó tự động học cách biểu diễn và đại diện cho các từ và câu trong các bài toán xử lý ngôn ngữ tự nhiên. BERT đã đạt được kết quả vượt trội trên nhiều bài toán NLP và trở thành một trong những mô hình NLP tiên tiến nhất hiện nay.

Do tiếp cận theo tư tưởng của RoBERTa, PhoBERT chỉ sử dụng task Masked Language Model để train, bỏ đi task Next Sentence Prediction.

### 1.2. Input và Output của BERT

Đầu vào của BERT là một đoạn văn bản, được mã hóa dưới dạng các chuỗi token (từ hoặc ký tự) và được thêm vào các token đặc biệt để biểu thị đầu/cuối câu hoặc đầu/cuối đoạn văn.

Đầu ra của BERT là một vector biểu diễn cho mỗi token trong câu đầu vào, được gọi là các feature vector. Các feature vector này có số chiều cố định và được huấn luyện để chứa thông tin phong phú về từng từ trong văn bản, bao gồm thông tin về ngữ nghĩa, cú pháp và ngữ cảnh.

Giả sử chúng ta có câu sau đây:

"John likes to watch movies and play football."

Input của BERT sẽ là chuỗi token sau đây:

[CLS] John likes to watch movies and play football. [SEP]

Trong đó, [CLS] và [SEP] là các token đặc biệt. [CLS] được thêm vào đầu câu để BERT biết đó là input của một nhiệm vụ cụ thể. [SEP] được sử dụng để phân tách hai đoạn văn khác nhau trong cùng một input.

Output của BERT là một vector biểu diễn cho toàn bộ câu đầu vào. Cụ thể, BERT sẽ tạo ra một ma trận embedding với kích thước [1 x L], trong đó L là chiều dài của vector embedding. Ma trận embedding này chứa thông tin về toàn bộ câu đầu vào và có thể được sử dụng để giải quyết các nhiệm vụ NLP khác nhau như phân loại văn bản, dự đoán từ tiếp theo trong câu, và phân tích cú pháp.

Hiểu thêm về cách thức hoạt động, oputput và input của BERT tại: https://miai.vn/2020/12/14/bert-series-chuong-1-bert-la-cai-chi-chi/

### 1.3. Ví dụ

Giả sử bạn muốn sử dụng BERT để phân loại email là thư rác (spam) hay thư thường (ham). Bạn sẽ cung cấp cho BERT một chuỗi ký tự đại diện cho email đó làm input. BERT sẽ xử lý và trả về cho bạn một output là xác suất rằng email đó là thư rác hoặc thư thường. Ví dụ, BERT có thể trả về output là "0.8" nghĩa là email đó có khả năng cao là thư rác với xác suất là 80%, hoặc "0.2" nghĩa là email đó có khả năng cao là thư thường với xác suất là 80%.

### 1.4. Ưu điểm PhoBERT so với RoBERTa

PhoBERT là một phiên bản được tinh chỉnh của RoBERTa cho ngôn ngữ tiếng Việt. Một trong những điểm mạnh của PhoBERT so với RoBERTa là việc sử dụng fastBPE để chia các câu thành các phần tử con với kích thước động bằng các subword units. Subword units được sử dụng để xử lý các từ dài, hiếm hoặc chưa từng được thấy trước đó, giúp mô hình có khả năng học được các từ mới và có khả năng tổng quát hóa tốt hơn. Ngoài ra, việc sử dụng fastBPE giúp tối ưu hóa việc xử lý dữ liệu và cải thiện độ chính xác của mô hình.

## 2. Phương pháp mã hóa Byte-Pair (BPE)

### 2.1. Tìm hiểu về mã hóa (Tokenize)

Trong PhoBERT, quá trình tokenize (tách từ) là quá trình tách câu văn thành các "token" (một đơn vị nhỏ nhất trong xử lý ngôn ngữ tự nhiên) để chuẩn bị cho quá trình xử lý tiếp theo như mã hóa và phân loại.

Được thực hiện bằng cách sử dụng mô hình tách từ từ vựng thay vì tách từ theo dạng tuần tự từ trái sang phải như các phương pháp thông thường, mô hình này sử dụng một đồ thị từ vựng để tối ưu hóa quá trình tách từ. Kết quả là việc tách từ được thực hiện hiệu quả hơn, đặc biệt là đối với các từ có cấu trúc phức tạp trong tiếng Việt.

### 2.2. Thuật toán BPE

**Định nghĩa:** 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.

**Cách thức hoạt động:** 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.

**Ví dụ 1:** Cụ thể, BPE sử dụng phương pháp tìm và thay thế các cặp ký tự (byte pair) phổ biến nhất bằng một ký hiệu mới, nhằm giảm thiểu số lượng các ký tự cần được lưu trữ. Ví dụ, trong tiếng Anh, các cặp ký tự "th", "he", "in", "er" thường xuất hiện nhiều trong các từ, vì vậy chúng có thể được thay thế bằng một ký hiệu đại diện.

**Ví dụ 2**: từ "tin tức" trong phoBERT có thể được tách thành hai subword units là "tin" và "tức" bằng mô hình BPE. Sau đó, chúng sẽ được mã hóa thành các vector số để đưa vào mạng nơ-ron.

**Sử dụng trong PhoBERT:** Việc sử dụng mô hình BPE trong phoBERT giúp cho mô hình có thể học được các từ mới mà nó chưa từng gặp trước đó, đồng thời cũng giúp giảm kích thước của từ điển đầu vào. Từ đó, mô hình có thể đạt được kết quả tốt hơn trên các tác vụ NLP cho tiếng Việt.

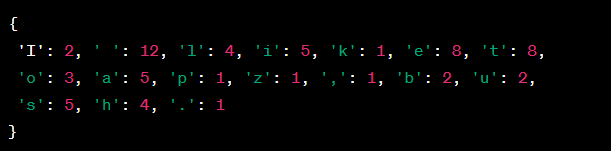
### 2.3. Quá trình thực hiện thuật toán BPE

Quá trình cơ bản gồm các bước như sau:

**Bước 1**: Tính tần suất xuất hiện của các ký tự trong văn bản.

Giả sử chúng ta có đoạn văn bản sau: "I like to eat pizza, but I also like to eat sushi."

Ta tính tần suất xuất hiện của các ký tự trong văn bản:



**Bước 2:** Tạo ra các cặp ký tự phổ biến nhất và thực hiện mã hóa.

Ta lặp lại các bước sau cho đến khi đạt được số lượng mã hóa mong muốn:

* Tìm cặp ký tự phổ biến nhất trong các ký tự đã tính tần suất ở bước 1.
* Thay thế cặp ký tự phổ biến đó bằng một ký tự mới, được đặt tên là "ký tự đặc biệt".
* Cập nhật lại bảng tần suất xuất hiện với ký tự đặc biệt vừa được thêm vào.

Ví dụ về cách thực hiện quá trình BPE trên đoạn văn bản trên:

* Tìm cặp ký tự phổ biến nhất: 'e' và 't'
* Thay thế cặp ký tự bằng ký tự đặc biệt: 'et' -> '@'
* Cập nhật bảng tần suất xuất hiện:

A picture containing indoor

Description automatically generated

**Bước 4**: Lặp lại các bước 1-3 cho đến khi đạt được số lượng mã hóa mong muốn.

Ví dụ, nếu ta muốn tạo ra 100 mã hóa, ta sẽ lặp lại các bước trên cho đến khi đạt được 100 ký tự đặc biệt.

**Bước 5**: ta chọn cặp phổ biến có tần suất cao nhất và thực hiện quá trình ghép nối lại, kết quả là "Th" được tạo thành. Tiếp tục thực hiện từng bước như trên, ta sẽ tạo ra các subword khác như "the", "ll", "er", "ea", "is", "as", "la", "it", "hi", "st". Kết quả cuối cùng là tập hợp các subword sau:



Việc tokenize một từ bằng BPE sẽ thực hiện bằng cách tìm kiếm các subword có trong từ đó, và nối chúng lại để tạo thành chuỗi subword. Ví dụ, từ "hello" sẽ được tokenize thành ['hel', 'l', 'o'], trong đó "hel" là một subword có trong từ điển, còn "l" và "o" là các ký tự đơn.

Chi tiết tại: <https://phamdinhkhanh.github.io/2020/06/04/PhoBERT_Fairseq.html#4-t%C3%ACm-hi%E1%BB%83u-v%E1%BB%81-m%C3%A3-h%C3%B3a-bpe-byte-pair-encoding>

# 2 PhoBERT

## 1. RoBERTa

### 1.1. Định nghĩa

RoBERTa là một mô hình ngôn ngữ được phát triển bởi Facebook AI Research (FAIR) vào năm 2019. Nó là một phiên bản cải tiến của mô hình BERT và được đào tạo trên một lượng lớn dữ liệu hơn so với BERT.

RoBERTa sử dụng một loạt các kỹ thuật tinh chỉnh để cải thiện quá trình huấn luyện và nâng cao hiệu suất của mô hình, bao gồm điều chỉnh các siêu tham số, tăng kích thước mô hình, mở rộng tập dữ liệu huấn luyện và thực hiện các quá trình tiền huấn khác. Nhờ những cải tiến này, RoBERTa đạt được kết quả tốt hơn so với BERT trên nhiều tác vụ ngôn ngữ tự nhiên. Trong bài báo được đề cập tới, PhoBERT sử dụng phương pháp tiền huấn dựa trên RoBERTa để đạt được hiệu suất tốt trên các tác vụ liên quan đến tiếng Việt.

### 1.2. Kỹ thuật cải tiến

RoBERTa sử dụng cùng một kiến trúc mạng nơ-ron như BERT, nhưng có một số cải tiến trong quá trình huấn luyện. Cụ thể, RoBERTa áp dụng các kỹ thuật cải tiến như:

1. **Không sử dụng mã hóa vị trí tương đối** (Relative Positional Encoding): Thay vì sử dụng mã hóa vị trí tương đối để định vị vị trí của từ, RoBERTa sử dụng mã hóa vị trí tuyệt đối (Absolute Positional Encoding) để chỉ định vị trí của từ trong câu.
2. **Không sử dụng token** [MASK]: Thay vì sử dụng token [MASK] như trong BERT để đưa ra câu hỏi cho mô hình, RoBERTa sử dụng một phương pháp khác để tạo ra dữ liệu huấn luyện, đó là sử dụng các phương pháp xáo trộn (data augmentation) để tạo ra các câu mới từ các câu đã có trong bộ dữ liệu.

Cụ thể: Trong quá trình tiền xử lý dữ liệu, BERT sử dụng một ký hiệu đặc biệt là [MASK] để thay thế ngẫu nhiên một số từ trong câu để cho mô hình học được khả năng dự đoán từ đó. Tuy nhiên, phương pháp này có thể gây ra sự hiện diện của [MASK] quá nhiều trong dữ liệu, dẫn đến sự thiếu đa dạng và overfitting.

Với RoBERTa, các tác giả đã thay đổi phương pháp tiền xử lý dữ liệu bằng cách sử dụng một phương pháp mới để xáo trộn chuỗi dữ liệu một cách hiệu quả hơn. Thay vì sử dụng [MASK], RoBERTa sử dụng phương pháp đưa ra một danh sách các câu khác nhau, sau đó xáo trộn các câu đó và lấy ra một số từ ngẫu nhiên để tạo thành câu mới. Phương pháp này giúp tăng tính đa dạng của dữ liệu và giảm thiểu sự xuất hiện quá nhiều của [MASK], giúp cải thiện khả năng mô hình học và tránh hiện tượng overfitting.

1. **Huấn luyện với thêm dữ liệu:** RoBERTa sử dụng một bộ dữ liệu huấn luyện lớn hơn nhiều so với BERT và các mô hình tiền đề khác. Bộ dữ liệu huấn luyện của RoBERTa bao gồm các nguồn khác nhau như Wikipedia, Common Crawl, BooksCorpus và một số nguồn báo chí điện tử. Điều này giúp tăng độ đa dạng của dữ liệu và cải thiện hiệu suất của mô hình.
2. **Tối ưu hóa tham số huấn luyện**: RoBERTa sử dụng các kỹ thuật tối ưu hóa tham số mới để cải thiện hiệu suất của mô hình. Cụ thể, RoBERTa sử dụng kỹ thuật "dynamic masking" để ngăn chặn mô hình quá tập trung vào một phần nhất định của dữ liệu huấn luyện.

Ngoài ra để nâng cao độ chuẩn xác trong biểu diễn từ thì RoBERTa đã **loại bỏ tác vụ dự đoán câu tiếp theo và huấn luyện trên các câu dài hơn**.

### 1.3. Input và Output của RoBERTa

Input của RoBERTa là một đoạn văn bản, bao gồm một chuỗi các từ hoặc ký tự, được mã hóa dưới dạng số. Trước khi được đưa vào RoBERTa, văn bản sẽ được tiền xử lý bằng các phương pháp như tokenization, padding, và segment embedding. Đối với một văn bản dài hơn kích thước tối đa mà mô hình có thể xử lý, RoBERTa sẽ được áp dụng theo kiểu "sliding window" để xử lý từng phần của văn bản.

Output của RoBERTa là một vector biểu diễn cho văn bản đầu vào. Vector này có số chiều cố định và thường là 768 hoặc 1024 chiều tùy thuộc vào kích thước của mô hình. Vector này được sử dụng để thực hiện các tác vụ như phân loại, trích xuất thông tin, hay sinh văn bản.

Đây là một ví dụ về cách RoBERTa xử lý một câu văn bằng input và output tương ứng:

Input: "I love walking in the park on a sunny day."

Output:

* Embedding layer: các từ trong câu được mã hóa thành các vector số thực đại diện cho từ đó.
* Encoder layers: các vector từ được đưa qua nhiều lớp encoder để tính toán các biểu diễn phức tạp hơn cho câu đầu vào.
* Pooling layer: các biểu diễn vector của các từ trong câu được gom nhóm thành một vector đại diện cho toàn bộ câu.
* Classification layer: vector đại diện cho toàn bộ câu được đưa qua một lớp phân loại để phân loại câu theo mục đích tương ứng (ví dụ: phân loại cảm xúc, phân loại chủ đề,...).

Với ví dụ trên, đầu ra có thể là vector số thực biểu thị cho xác suất của câu đó thuộc vào một số lớp cảm xúc nhất định (ví dụ: positive, negative, neutral).

### 1.4. Ứng dụng

Một số ứng dụng của RoBERTa bao gồm:

1. **Phân loại văn bản**: RoBERTa được sử dụng để phân loại các văn bản trong nhiều lĩnh vực khác nhau, chẳng hạn như phân loại cảm xúc, phân loại tin tức, phân loại định hướng chính trị, phân loại ngôn ngữ, v.v.
2. **Trích xuất thông tin**: RoBERTa được sử dụng để trích xuất thông tin từ các văn bản, chẳng hạn như trích xuất tên, địa chỉ, ngày tháng, v.v. từ các tài liệu hợp đồng hoặc bản tin báo chí.
3. **Sinh câu**: RoBERTa được sử dụng để sinh câu tự động, giúp cho việc viết văn bản nhanh chóng và hiệu quả hơn.
4. **Dịch máy**: RoBERTa được sử dụng trong các hệ thống dịch máy để cải thiện chất lượng và độ chính xác của kết quả dịch.
5. **Xử lý ngôn ngữ tự nhiên**: RoBERTa được sử dụng để giải quyết các vấn đề trong xử lý ngôn ngữ tự nhiên, chẳng hạn như phân tích cú pháp, phân tích tình cảm, nhận dạng thực thể tên, v.v.

Một trong những ứng dụng thực tế của RoBERTa là trong việc xử lý ngôn ngữ tự nhiên và phân tích cảm xúc. Ví dụ, các công ty có thể sử dụng RoBERTa để phân tích cảm xúc của khách hàng trên các kênh truyền thông xã hội như Twitter hoặc Facebook. RoBERTa có thể phân tích và đánh giá nội dung từ những bài viết trên các trang mạng xã hội, từ đó giúp các công ty có thể hiểu được cảm nhận của khách hàng đối với sản phẩm hoặc dịch vụ của mình và đưa ra các cải tiến hoặc điều chỉnh phù hợp. Ngoài ra, RoBERTa cũng có thể được sử dụng trong các ứng dụng như dịch thuật tự động, nhận diện giọng nói, tóm tắt văn bản và phân tích tâm lý khách hàng.

### 1.5. Ví dụ

**Ví dụ 1:**

Để minh họa cho input và output của RoBERTa, chúng ta có thể sử dụng ví dụ sau:

Input: "Tôi là sinh viên Đại học Bách Khoa Hà Nội."

Output:

* Các token được tách ra: ["Tôi", "là", "sinh\_viên", "Đại\_học", "Bách\_Khoa", "Hà\_Nội", "."]
* Ma trận nhúng từ (embedding matrix) được tạo ra bằng cách sử dụng các vectơ nhúng từ (word embeddings) được học từ dữ liệu lớn trước khi huấn luyện. Mỗi token được biểu diễn bởi một vectơ 768 chiều.
* Đầu vào được chuẩn hóa bằng cách thêm các kí tự padding (chẳng hạn như kí tự 0) để có cùng độ dài. Ví dụ, nếu số lượng token tối đa trong một câu được đặt là 10, một câu có số lượng token ít hơn 10 sẽ được thêm các kí tự padding để đạt được độ dài bằng 10.
* Ma trận đầu vào (input matrix) được tạo ra từ các vectơ nhúng từ đã được chuẩn hóa và các kí tự padding. Ma trận này có kích thước là 10x768 nếu số lượng token tối đa là 10.
* Một lớp Transformer được áp dụng vào ma trận đầu vào để tính toán các đại diện ngữ nghĩa (semantic representations) của câu. Lớp Transformer gồm nhiều lớp con Transformer, mỗi lớp con có một số lượng head attention và một số lượng neuron ẩn. Các lớp Transformer này được lặp lại nhiều lần để tăng cường khả năng học và tạo ra các đại diện ngữ nghĩa có độ phức tạp cao hơn.
* Các đại diện ngữ nghĩa của các token trong câu được tạo ra bằng cách lấy giá trị của ma trận đầu ra của lớp Transformer tương ứng với vị trí của token đó trong câu. Mỗi token được biểu diễn bởi một vectơ 768 chiều.

Đầu ra của RoBERTa có thể được sử dụng để thực hiện các tác vụ NLP như phân loại, dịch máy, tóm tắt nội dung, và trích xuất thông tin, và nhiều ứng dụng khác.

**Ví dụ 2:**

Input: "I love playing football with my friends"

1. Tokenization: ['I', 'love', 'playing', 'football', 'with', 'my', 'friends']
2. Thêm token [CLS] vào đầu câu và [SEP] vào cuối câu: ['[CLS]', 'I', 'love', 'playing', 'football', 'with', 'my', 'friends', '[SEP]']
3. Chuyển đổi các token thành ID theo từ điển của RoBERTa: [0, 31414, 1218, 10599, 14568, 61, 144, 567, 2]
4. Thêm các token padding để đạt độ dài chuẩn của input (ví dụ: 10): [0, 31414, 1218, 10599, 14568, 61, 144, 567, 2, 0]
5. Đưa input đã qua xử lý vào mô hình RoBERTa
6. Lấy vector biểu diễn của token [CLS] làm output của mô hình

Output: vector biểu diễn của token [CLS] sau khi đưa input "I love playing football with my friends" qua mô hình RoBERTa.

## 2. Adam

Adam (Adaptive Moment Estimation) là một trong những thuật toán tối ưu hóa gradient descent được sử dụng rộng rãi trong deep learning. Nó là một thuật toán tối ưu hóa tự điều chỉnh, tức là nó tự động thay đổi learning rate (tốc độ học) cho mỗi tham số của mô hình, dựa trên thông tin gradient của chúng. Adam sử dụng hai bước để thực hiện tối ưu hóa: tính toán gradient và cập nhật trọng số của mô hình. Trong quá trình tính toán gradient, Adam tính trung bình của các gradient để cung cấp một ước lượng chính xác hơn cho hướng cập nhật. Trong khi cập nhật trọng số, Adam sử dụng kết hợp của trung bình gradient và trung bình bình phương gradient, kết hợp giữa tính chính xác và tính ổn định.

Adam được xem là một trong những thuật toán tối ưu hóa hiệu quả trong các bài toán deep learning, nhất là khi số lượng tham số lớn. Nó có thể giúp tăng tốc độ huấn luyện và đạt được kết quả tốt hơn trong thời gian ngắn hơn so với các thuật toán tối ưu hóa khác.

## 3. Optimization

### 3.1. Các bước thực hiện tối ưu hóa

1. **Implementation**: Mô hình RoBERTa được sử dụng trong fairseq (Ott et al., 2019).
2. **Maximum Length**: Một maximum length (độ dài tối đa) là 256 subword tokens được sử dụng để phân đoạn các câu trong quá trình huấn luyện mô hình. Với bộ dữ liệu được sử dụng, có tổng cộng 145 triệu câu và mỗi câu được chia thành các subword tokens. Do đó, sẽ có khoảng 13,8 triệu khối câu được tạo ra.
3. **Optimizer**: Trong quá trình huấn luyện mô hình, phương pháp tối ưu hóa Adam (Kingma và Ba, 2014) được sử dụng. Phương pháp này kết hợp giữa giá trị gradient trung bình và mô-men gradient để tối ưu hóa hàm mất mát.
4. **Batch Size**: Batch size là số lượng dữ liệu huấn luyện được sử dụng trong một lần cập nhật trọng số mô hình. Trong quá trình huấn luyện PhoBERTbase, batch size là 1024 và được chia đều trên 4 V100 GPUs, mỗi card có dung lượng 16GB. Trong khi đó, batch size cho PhoBERTlarge là 512.
5. **Learning Rate**: Learning rate là tham số quan trọng trong quá trình huấn luyện mô hình, đóng vai trò giúp định hướng và điều chỉnh quá trình tối ưu hóa trọng số mô hình. Trong quá trình huấn luyện PhoBERTbase, learning rate đạt đỉnh ở giá trị 0,0004. Trong khi đó, learning rate đạt đỉnh ở giá trị 0,0002 cho PhoBERTlarge.
6. **Epochs**: Một epoch là quá trình huấn luyện mô hình qua toàn bộ bộ dữ liệu huấn luyện. Trong quá trình huấn luyện PhoBERTbase và PhoBERTlarge, số epochs lần lượt là 40 và 5 tuần. Trong đó, learning rate được warm-up trong 2 epochs.

Tạo ra khoảng 540K bước huấn luyện cho PhoBERTbase và 1.08M bước huấn luyện cho PhoBERTlarge.

Cuối cùng, tiền huấn luyện PhoBERTbase trong 3 tuần và sau đó tiền huấn luyện PhoBERTlarge trong 5 tuần.

Những thông số trên được tính toán để đảm bảo quá trình huấn luyện mô hình được thực hiện một cách hiệu quả nhất.

### 3.2. Mục đích các bước trên

#### 3.2.1. Implementation

Sử dụng mã nguồn của RoBERTa và fairseq trong huấn luyện mô hình nhằm tối ưu hóa quá trình huấn luyện, tăng tốc độ tính toán, giảm độ phức tạp của việc lập trình và tăng khả năng tái sử dụng của mô hình.

RoBERTa và fairseq giúp cho việc huấn luyện và triển khai các mô hình NLP trở nên đơn giản và hiệu quả hơn. RoBERTa là một mô hình dựa trên kiến trúc của mô hình BERT nhưng được huấn luyện trên một tập dữ liệu lớn hơn, còn fairseq là một thư viện cho phép huấn luyện và triển khai các mô hình ngôn ngữ sử dụng các kiến trúc hiện đại như Transformer.

Việc sử dụng mã nguồn của RoBERTa và fairseq trong huấn luyện mô hình PhoBERT giúp tối ưu hóa quá trình huấn luyện bằng cách sử dụng các công cụ và thủ tục tiêu chuẩn trong các thư viện này. Điều này giúp cho quá trình huấn luyện trở nên đơn giản hơn, giảm thiểu khả năng sai sót trong lập trình và đảm bảo tính hiệu quả của mô hình được đào tạo.

#### 3.2.2. Maximum length

Đặt giới hạn chiều dài tối đa của câu là 256 subword tokens giúp giảm thiểu sự phức tạp tính toán và tiết kiệm bộ nhớ trong quá trình huấn luyện. Trong các mô hình dựa trên mạng nơ-ron, đặt giới hạn chiều dài tối đa của đầu vào có thể giúp cải thiện tốc độ huấn luyện và khả năng tổng quát hóa của mô hình.

Việc đặt giới hạn chiều dài trên có thể giúp tránh overfitting bằng cách đảm bảo rằng mô hình không học quá nhiều từ dữ liệu huấn luyện, đồng thời đảm bảo rằng mô hình có thể tổng quát hóa tốt trên các câu có chiều dài khác nhau trong tập dữ liệu mới.

#### 3.2.3. Optimizer (Adam)

Việc sử dụng Adam để tối ưu hóa quá trình huấn luyện giúp cải thiện tốc độ hội tụ của mô hình, tức là giảm thiểu thời gian cần thiết để mô hình đạt được kết quả tốt nhất.

#### 3.2.4. Batch Size

Sử dụng kích thước batch lớn và sử dụng nhiều GPU giúp tăng tốc quá trình huấn luyện.

#### 3.2.5. Learning Rate

Việc tăng thời gian huấn luyện mô hình cũng cải thiện chất lượng của mô hình, giúp nó đạt được kết quả tốt hơn khi được sử dụng trong các tác vụ xử lý ngôn ngữ tự nhiên.

#### 3.2.6. Epochs

Trong quá trình huấn luyện mô hình, epoch là một đơn vị đo lường được sử dụng để chỉ số lần mô hình được đưa qua toàn bộ tập dữ liệu huấn luyện. Nói cách khác, mỗi epoch là một vòng lặp qua toàn bộ tập dữ liệu huấn luyện. Trong trường hợp này, tác giả đã chọn huấn luyện mô hình trong 40 epoch.

Ngoài ra, tốc độ học được làm bắt đầu trong 2 epoch để giúp mô hình học tốt hơn từ đầu. Tốc độ học ở đây là một siêu tham số quan trọng trong quá trình huấn luyện mô hình deep learning. Nó quyết định tốc độ cập nhật trọng số mô hình dựa trên gradient được tính từ hàm mất mát trong quá trình huấn luyện. Tốc độ học quá cao có thể dẫn đến việc mô hình bị tràn số hoặc hội tụ không tốt. Trong khi đó, tốc độ học quá thấp sẽ khiến quá trình huấn luyện chậm và mô hình không hội tụ đến kết quả tốt.

Tổng số bước huấn luyện được tính dựa trên số lượng khối câu được tạo ra và kích thước lô (batch size) được sử dụng trong quá trình huấn luyện. Kích thước lô là số lượng ví dụ đầu vào được sử dụng để cập nhật trọng số mô hình trong mỗi lần đưa qua mạng. Trong trường hợp này, tác giả đã sử dụng kích thước lô là 1024 cho PhoBERTbase và 512 cho PhoBERTlarge, chia đều trên 4 GPU V100 (16GB mỗi GPU).

Tổng số bước huấn luyện cũng phụ thuộc vào số epoch và kích thước lô. Tác giả tính toán số bước huấn luyện cho mỗi phiên bản PhoBERT dựa trên công thức: số lượng khối câu được tạo ra x số epoch / kích thước lô. Kết quả là khoảng 540K bước huấn luyện cho PhoBERTbase và 1.08M bước huấn luyện cho PhoBERTlarge.

# **3 Experimental setup**

Fine-tuning trong PhoBERT bao gồm các bước sau:

1. Chuẩn bị dữ liệu: Chọn các tập dữ liệu phù hợp cho từng tác vụ (POS tagging, NER, NLI, ...), chia tập dữ liệu thành tập train, validation và test, xác định định dạng dữ liệu đầu vào và đầu ra cho từng tác vụ cụ thể.
2. Load pre-trained PhoBERT: Sử dụng pre-trained PhoBERT (đã được train trên large corpus) để tạo ra các embedding cho câu đầu vào.
3. Thêm lớp head: Tạo một hoặc nhiều lớp head để thực hiện tác vụ cụ thể, ví dụ như là POS tagging, NER, NLI, ...
4. Fine-tune mô hình: Sử dụng tập train để tinh chỉnh mô hình, cập nhật các trọng số của mô hình dựa trên hàm mất mát và thuật toán tối ưu hóa, ví dụ như AdamW.
5. Tinh chỉnh tham số và Đánh giá mô hình: Thực hiện việc tinh chỉnh các tham số mô hình như số lượng epoch, learning rate, batch size, dropout rate, số lượng lớp ẩn, số lượng heads, v.v. để cải thiện hiệu suất của mô hình cho từng tác vụ cụ thể. Sau mỗi epoch, đánh giá hiệu suất của mô hình trên tập validation. Nếu không có cải thiện nào trong 5 epoch liên tiếp, dừng sớm (early stopping). Sau khi tất cả các epoch được huấn luyện xong, chọn checkpoint tốt nhất và đánh giá trên tập test.
6. Đánh giá kết quả: Đánh giá kết quả bằng các chỉ số thích hợp cho từng tác vụ và so sánh với các mô hình khác.

## **3.1. Kiến trúc Transformer**

Transformer là một kiến trúc mạng nơ-ron đang được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên. Trong kiến trúc này, đầu vào và đầu ra đều là một chuỗi các token được biểu diễn dưới dạng các vector embedding.

Trong PhoBERT, Transformer được sử dụng để tinh chỉnh các embedding của các từ và phần tử ngữ cảnh (contextualized embedding), giúp nâng cao hiệu suất của PhoBERT trong các tác vụ như POS tagging, NER và NLI.

Cụ thể, PhoBERT sử dụng một lớp transformer để học biểu diễn ngôn ngữ tự nhiên của các từ trong câu. Lớp transformer này được sử dụng để tính toán các embedding contextualized của các từ trong câu, cho phép mô hình hiểu được các từ trong ngữ cảnh của chúng. Sau đó, các embedding này được sử dụng để phân loại văn bản (như trong trường hợp phân loại POS và NER), phân tích cú pháp (như trong trường hợp phân tích cây phụ thuộc), hoặc giải quyết các nhiệm vụ khác trong xử lý ngôn ngữ tự nhiên.

Tóm lại, mục đích của PhoBERT là sử dụng transformer trong quá trình fine-tuning để tăng cường khả năng xử lý ngôn ngữ tự nhiên của mô hình cho tiếng Việt.