# 

Contents

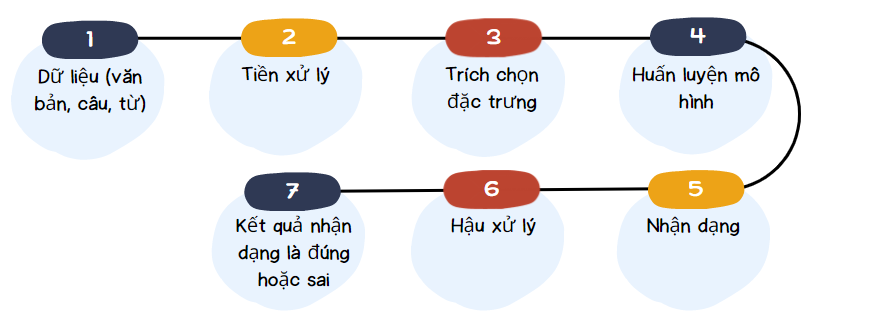
[Giới thiệu 2](#_Toc156326730)

[TÓM TẮT NỘI DUNG 3](#_Toc156326731)

[Chương i. Giới thiệu đề tài 4](#_Toc156326732)

[**1. Tổng quan đề tài** 4](#_Toc156326733)

[2. Cách tiếp cận giải quyết bài toán 5](#_Toc156326734)

[ 5](#_Toc156326735)

[3. Tổng quan về các phương pháp huấn luyện 6](#_Toc156326736)

[3.1. Logistic Regression 6](#_Toc156326737)

[3.2. RoBERTa 8](#_Toc156326738)

[3.3. Mô hình Mạng Nơ-ron Biên (BiRNN) 9](#_Toc156326739)

[4. Lý do vì sao chọn mô hình RoBERTa 10](#_Toc156326740)

[Chương II : Cơ sở lý thuyết về tiền xử lý dữ liệu và trích chọn đặc trưng 12](#_Toc156326741)

[1. Tổng quan về tiền xử lý dữ liệu 12](#_Toc156326742)

[2. Các công đoạn tiền xử lý 12](#_Toc156326743)

[2.1. Xây dựng từ điển (Vocabulary Construction) 13](#_Toc156326744)

[2.2. Tách từ (Tokenization) và mã hóa BPE (BPE Encoding) 14](#_Toc156326745)

[2.3. Chuyển dữ liệu về dạng số (Numeric Conversion) 15](#_Toc156326746)

[2.4. Chuẩn hoá kích thước dữ liệu(Normalize data size) 16](#_Toc156326747)

[Chương III: CƠ SỞ LÝ THUYẾT VỀ RoBERTa 19](#_Toc156326748)

[1. Giới thiệu 19](#_Toc156326749)

[2. Cấu trúc 19](#_Toc156326750)

[3. Cách hoạt động 20](#_Toc156326751)

[Chương IV. MÔ TẢ BÀI TOÁN 23](#_Toc156326752)

[1. Giới thiệu bài toán 23](#_Toc156326753)

[2. Dữ liệu 23](#_Toc156326754)

[2.1. Đọc và chia dữ liệu 24](#_Toc156326755)

[2.2. Tokenize nội dung và labels 25](#_Toc156326756)

[2.3. Load model RoBERTa 27](#_Toc156326757)

[2.4. Huấn luyện mô hình 27](#_Toc156326758)

[3. Đánh giá kết quả 31](#_Toc156326759)

[Chương V. kết luận và kiến nghị 33](#_Toc156326760)

[tài liệu tham khảo 35](#_Toc156326761)

# Giới thiệu

Trong quá trình viết văn bản tiếng Việt, lỗi ngữ pháp là một vấn đề thường gặp ở cả người Việt Nam hay người nước ngoài sử dụng tiếng Việt. Lỗi ngữ pháp có thể gây nhầm lẫn hoặc hiểu sai nội dung của văn bản. Tuy nhiên, việc phát hiện lỗi ngữ pháp tiếng Việt là một công việc đòi hỏi sự tinh tế và chính xác, đặc biệt là khi có hàng ngàn từ ngữ và cú pháp khác nhau trong ngôn ngữ này.

Vì vậy, việc phát triển một phần mềm phát hiện lỗi ngữ pháp của tiếng Việt là rất cần thiết. Điều này có thể giúp những người sử dụng tiếng Việt, bao gồm cả các nhà văn, biên tập viên, nhà báo và học sinh, tránh được những lỗi ngữ pháp cơ bản và nâng cao chất lượng văn bản của họ. Ngoài ra, phần mềm phát hiện lỗi ngữ pháp tiếng Việt còn có thể hỗ trợ giáo dục ngôn ngữ, giúp các học sinh và người học tiếng Việt nâng cao kỹ năng viết và sử dụng ngôn ngữ của mình một cách chính xác hơn.

# TÓM TẮT NỘI DUNG

Báo cáo khoa học này tập trung vào ứng dụng học máy trong nhận biết lỗi chính tả Tiếng Việt. Lỗi chính tả trong văn bản Tiếng Việt gây hiểu lầm và giảm tính chuyên nghiệp. Nghiên cứu này nhằm áp dụng phương pháp học máy để nhận biết và sửa lỗi chính tả Tiếng Việt.

Báo cáo trình bày các phương pháp và thuật toán học máy được sử dụng, cùng với quá trình thu thập và xử lý dữ liệu Tiếng Việt chứa lỗi chính tả. Một mô hình học máy được xây dựng để nhận biết lỗi chính tả trong văn bản.

Nghiên cứu này hướng đến việc cải thiện chất lượng văn bản Tiếng Việt và hỗ trợ viết và giao tiếp chính xác hơn trong nhiều lĩnh vực. Đóng góp của nghiên cứu này là mở rộng ứng dụng học máy trong nhận biết lỗi chính tả Tiếng Việt và tạo điều kiện cho phát triển tiếp theo trong việc xử lý các lỗi chính tả phức tạp và áp dụng vào các ngôn ngữ khác.

Tóm lại, ứng dụng học máy trong nhận biết lỗi chính tả Tiếng Việt mang lại tiềm năng và lợi ích đáng kể trong cải thiện chất lượng văn bản và giao tiếp chính xác. Nghiên cứu này đóng góp vào phát triển của lĩnh vực này và mở ra các hướng nghiên cứu và ứng dụng tiếp theo.

# Chương i. Giới thiệu đề tài

**1. Tổng quan đề tài**

Hiện tại, nghiên cứu về ứng dụng học máy trong nhận biết lỗi chính tả đã đạt được nhiều tiến bộ đáng kể. Các phương pháp dựa trên học máy đã được áp dụng để xây dựng các hệ thống tự động phát hiện và sửa lỗi chính tả trong văn bản.

Một số kỹ thuật phổ biến được sử dụng làm cơ sở cho các ứng dụng học máy trong nhận biết lỗi chính tả bao gồm:

i. Mô hình ngôn ngữ: Các mô hình ngôn ngữ sử dụng học máy để dự đoán xác suất của các từ hoặc câu trong một văn bản. Các mô hình này có thể được sử dụng để phát hiện các từ không phù hợp hoặc câu không hợp lý trong văn bản, gợi ý các sửa đổi để khắc phục lỗi chính tả.

ii. Mô hình dịch máy: Các mô hình dịch máy sử dụng học máy để dịch văn bản từ một ngôn ngữ sang ngôn ngữ khác. Các mô hình này có thể phát hiện và sửa lỗi chính tả khi chúng không phù hợp với ngữ cảnh hoặc ngữ pháp của ngôn ngữ đích.

iii. Mạng nơ-ron hồi quy: Mạng nơ-ron hồi quy (RNN) và các biến thể của nó, chẳng hạn như LSTM (Long Short-Term Memory), đã được sử dụng rộng rãi để xử lý vấn đề nhận biết lỗi chính tả. RNN có khả năng xử lý thông tin tuần tự, giúp nắm bắt ngữ cảnh và mối quan hệ giữa các từ trong văn bản.

iv. Mô hình Transformer: Mô hình Transformer, đặc biệt là biến thể của nó như BERT (Bidirectional Encoder Representations from Transformers), đã mang lại những cải tiến đáng kể trong lĩnh vực nhận biết lỗi chính tả. Các mô hình này có khả năng xử lý ngữ cảnh phức tạp và nhận biết các lỗi chính tả dựa trên các quan hệ ngữ pháp và ý nghĩa của từ trong câu.

Tuy nhiên, việc nhận biết lỗi chính tả là một vấn đề phức tạp, bởi vì nhiều lỗi chính tả phụ thuộc vào ngữ cảnh và có thể không dễ dàng nhận diện. Ngoài ra, các ngôn ngữ và văn bản có sự đa dạng cao cũng tạo ra thách thức trong việc xây dựng các hệ thống nhận biết lỗi chính tả chính xác và đáng tin cậy.

Tóm lại, mặc dù đã có nhiều tiến bộ trong lĩnh vực này, nhưng việc nhận biết lỗi chính tả vẫn là một vấn đề nghiên cứu tiếp tục và đòi hỏi sự phát triển và cải tiến của các phương pháp học máy.

## 2. Cách tiếp cận giải quyết bài toán

## 

Ứng dụng học máy trong việc nhận diện chính tả đúng/sai có thể được thực hiện qua các bước sau:

1. Dữ liệu: Đây là bước đầu tiên trong quá trình. Dữ liệu ban đầu được thu thập từ các nguồn khác nhau, là các văn bản từ bài báo trên mạng, đời sống, … . Dữ liệu này được tổ chức và lưu trữ trong các định dạng phù hợp để tiếp tục các bước tiền xử lý.

2. Tiền xử lý: Bước tiền xử lý là quá trình làm sạch và chuẩn bị dữ liệu trước khi đưa vào mô hình học máy. Điều này bao gồm các công việc như loại bỏ dữ liệu nhiễu, xử lý dữ liệu thiếu, chuẩn hóa và chuẩn hóa dữ liệu, thực hiện các phép biến đổi và tạo các biểu diễn phù hợp cho dữ liệu.

3. Trích chọn đặc trưng: Bước này liên quan đến việc chọn ra những đặc trưng quan trọng từ dữ liệu tiền xử lý để mô hình có thể học từ đó. Trích chọn đặc trưng có thể sử dụng các phương pháp như kỹ thuật mã hóa, biến đổi dữ liệu, trích xuất thông tin quan trọng và giảm chiều dữ liệu.

4. Huấn luyện mô hình: Sau khi có dữ liệu tiền xử lý và các đặc trưng được trích chọn, quá trình huấn luyện mô hình được thực hiện. Mô hình học máy sẽ được huấn luyện trên dữ liệu huấn luyện đã được gán nhãn để học các mẫu và quy luật trong dữ liệu.

5. Nhận dạng: Sau khi mô hình đã được huấn luyện, quá trình nhận dạng sẽ được thực hiện. Dữ liệu mới sẽ được đưa vào mô hình để dự đoán và phân loại. Quá trình này sẽ cho ra các kết quả dự đoán dựa trên mô hình đã được huấn luyện.

6. Hậu xử lý: Kết quả nhận dạng từ mô hình có thể cần được xử lý để loại bỏ nhiễu, sắp xếp lại hoặc tinh chỉnh cải tiến mô hình để đạt được kết quả tốt nhất.

7. Kết quả nhận dạng: Cuối cùng, kết quả từ quá trình nhận dạng sẽ được đánh giá để xem liệu chúng có chính xác hay không. Các phương pháp đánh giá khác nhau có thể được sử dụng, bao gồm tính chính xác, độ đo F1, ma trận nhầm lẫn và các phương pháp khác để đánh giá hiệu suất của mô hình.

Tóm lại, quá trình từ dữ liệu đến kết quả nhận dạng thông qua mô hình học máy là một chuỗi các bước tiền xử lý, trích chọn đặc trưng, huấn luyện, nhận dạng, hậu xử lý và đánh giá. Mỗi bước đóng vai trò quan trọng trong việc xử lý và trích xuất thông tin từ dữ liệu để đưa ra kết quả nhận dạng chính xác.

## 3. Tổng quan về các phương pháp huấn luyện

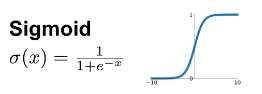
### 3.1. Logistic Regression

Logistic Regression là một phương pháp quan trọng và phổ biến trong lĩnh vực Machine Learning và Statistical Modeling. Phương pháp này được phát triển bởi David Cox vào năm 1958 và sau đó được nâng cấp bởi Donald Rubin vào năm 1972.

Ban đầu, Logistic Regression được sử dụng trong các bài toán dự đoán xác suất và phân loại trong các nghiên cứu y học và y học thực nghiệm. Nó đã chứng minh được tính linh hoạt và hiệu quả trong việc xử lý các bài toán phân loại hai lớp và đa lớp.

Hàm mô hình Logistic Regression hoạt động dựa trên một hàm sigmoid (hay hàm logistic) để ánh xạ đầu vào tuyến tính thành xác suất.

Đầu tiên, để hiểu cách hoạt động của Logistic Regression, chúng ta cần biết về hàm sigmoid. Hàm sigmoid có dạng:



Trong đó, x là tổng trọng số của các biến đầu vào kết hợp với các tham số (weights) tương ứng. Hàm sigmoid có giá trị nằm trong khoảng (0, 1) và được sử dụng để ánh xạ giá trị đầu vào thành xác suất.

Trong Logistic Regression, chúng ta sử dụng hàm sigmoid để dự đoán xác suất xảy ra của một sự kiện (ví dụ: lỗi sai chính tả) dựa trên các biến đầu vào. Xác suất này được ký hiệu là P(y=1|z) và có thể được tính bằng:

P(y=1|z) = sigmoid(x)

Trong đó, y là biến phụ thuộc (đầu ra) và z là các biến đầu vào.

Các tham số (weights) trong Logistic Regression được ước lượng bằng cách tối ưu hóa một hàm mất mát (loss function) như hàm cross-entropy loss. Mục tiêu là điều chỉnh các tham số sao cho mô hình có khả năng dự đoán xác suất chính xác nhất.

Quá trình tối ưu hóa được thực hiện thông qua các phương pháp như Gradient Descent hoặc các biến thể của nó. Mục tiêu là điều chỉnh các tham số để giảm thiểu sai số giữa xác suất dự đoán và giá trị thực tế của biến phụ thuộc.

Khi quá trình huấn luyện hoàn thành và các tham số được ước lượng, ta có thể sử dụng mô hình Logistic Regression để dự đoán xác suất xảy ra của sự kiện mới dựa trên các giá trị đầu vào mới.

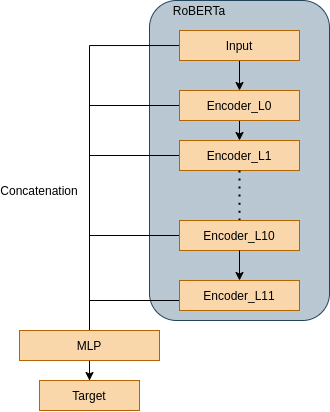
Tóm lại, Logistic Regression là một phương pháp sử dụng hàm sigmoid để dự đoán xác suất xảy ra của một sự kiện dựa trên các biến đầu vào. Quá trình tối ưu hóa được thực hiện để điều chỉnh các tham số sao cho mô hình có khả năng dự đoán chính xác nhất.

### 3.2. RoBERTa

Mô hình RoBERTa (Robustly Optimized BERT approach) là một biến thể của mô hình BERT (Bidirectional Encoder Representations from Transformers) được tối ưu hóa để đạt hiệu suất tốt hơn trong các nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm bài toán nhận diện lỗi chính tả đúng/sai.

RoBERTa sử dụng kiến trúc mã hóa theo hướng chuyển tiếp (transformer-based encoder) để mã hóa ngữ cảnh của các câu văn và tạo ra biểu diễn nhiều chiều (embeddings) cho từng từ trong câu. Mô hình sử dụng một tập hợp các lớp mã hóa transformer (transformer encoder layers) để tìm hiểu mối quan hệ ngữ nghĩa và cú pháp giữa các từ trong câu.

Quá trình hoạt động của mô hình RoBERTa có các bước chính như sau:

Đầu vào: Câu văn đầu vào được chuyển đổi thành biểu diễn số hóa (numerical representation) bằng cách ánh xạ từng từ trong câu vào các vector số (word embeddings).

Mã hóa ngữ cảnh: Các lớp mã hóa transformer được áp dụng lên các biểu diễn từ để tạo ra biểu diễn ngữ cảnh cho từng từ trong câu. Quá trình này cho phép mô hình học cú pháp và ngữ nghĩa của câu văn.

Biểu diễn đại diện: Các biểu diễn ngữ cảnh của các từ trong câu văn được tổng hợp lại thành một biểu diễn đại diện cho cả câu. Điều này cho phép mô hình nhận biết thông tin ngữ nghĩa và cú pháp của toàn bộ câu văn.

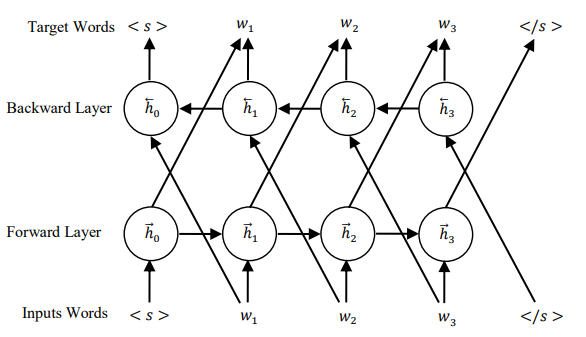
Đầu ra: Biểu diễn đại diện của câu văn được đưa vào một lớp kết nối đầy đủ (fully connected layer) và áp dụng hàm kích hoạt softmax để dự đoán xác suất cho mỗi lớp đích, trong trường hợp này là "lỗi chính tả đúng" và "lỗi chính tả sai".

Mô hình RoBERTa được huấn luyện thông qua quá trình tối ưu hóa các tham số dựa trên một hàm mất mát (loss function) như cross-entropy loss, trong đó mục tiêu là điều chỉnh các tham số để giảm thiểu sai số giữa xác suất dự đoán và nhãn thực tế của câu văn.

Tổng quan, mô hình RoBERTa sử dụng kiến trúc mã hóa transformer để mã hóa ngữ cảnh của câu văn và dự đoán xác suất cho các lớp đích. Quá trình huấn luyện mô hình được thực hiện thông qua tối ưu hóa các tham số để tối thiểu hóa sai số giữa xác suất dự đoán và nhãn thực tế. Mô hình RoBERTa đã cho thấy hiệu suất tốt trong các nhiệm vụ xử lý ngôn ngữ tự nhiên, bao gồm nhận diện lỗi chính tả đúng/sai.

### 3.3. Mô hình Mạng Nơ-ron Biên (BiRNN)

Mô hình Mạng Nơ-ron Biên (BiRNN) là một kiến trúc mạng nơ-ron hồi quy (RNN) được sử dụng trong bài toán nhận diện lỗi chính tả đúng sai. Đặc trưng chính của BiRNN là khả năng "nhìn ngược" vào ngữ cảnh của câu, giúp mô hình có cái nhìn tổng quan về các từ trước và sau từ đang xem xét.

Cách thức hoạt động của mô hình BiRNN trong bài toán nhận diện lỗi chính tả đúng sai như sau:

1. Đầu vào: Các câu văn bản được chia thành các từ (tokenization) và được biểu diễn dưới dạng các vector từ (word embeddings). Mỗi từ trong câu được biểu diễn bằng một vector số thực có kích thước cố định.
2. Mô hình BiRNN: Mô hình BiRNN gồm hai phần, một phần là RNN theo hướng thuận (forward RNN) và một phần là RNN theo hướng ngược (backward RNN). Cả hai RNN này có cấu trúc tương tự và chia sẻ các trọng số.
3. Forward RNN: Forward RNN nhận các vector từ đầu vào và thực hiện lan truyền thuận qua các từ theo thứ tự từ đầu đến cuối câu. Mỗi lần lan truyền, forward RNN cập nhật trạng thái ẩn và tính toán đầu ra.
4. Backward RNN: Backward RNN nhận các vector từ đầu vào và thực hiện lan truyền ngược qua các từ theo thứ tự từ cuối câu đến đầu câu. Mỗi lần lan truyền, backward RNN cập nhật trạng thái ẩn và tính toán đầu ra.
5. Kết hợp đầu ra: Kết quả từ forward RNN và backward RNN được kết hợp để tạo thành đầu ra cuối cùng của mô hình BiRNN. Đầu ra này thường được sử dụng để dự đoán nhãn của từng từ trong câu, ví dụ như nhãn "đúng" hoặc "sai" cho từng từ chứa lỗi chính tả.
6. Huấn luyện: Mô hình BiRNN được huấn luyện bằng cách cung cấp các cặp câu và nhãn tương ứng (có lỗi chính tả hoặc không lỗi chính tả). Quá trình huấn luyện sẽ điều chỉnh các trọng số của mô hình để tối thiểu hóa sai số giữa nhãn dự đoán và nhãn thực tế.
7. Dự đoán: Sau khi mô hình BiRNN được huấn luyện, nó có thể được sử dụng để dự đoán nhãn của các câu mới, tức là xác định lỗi chính tả đúng sai trong các từ.

Mô hình BiRNN có khả năng nhìn thấy cả ngữ cảnh trước và sau của từ, giúp nắm bắt được sự phụ thuộc giữa các từ trong câu. Điều này làm cho BiRNN phù hợp cho bài toán nhận diện lỗi chính tả, vì việc nhận diện lỗi chính tả thường phụ thuộc vào ngữ cảnh xung quanh từ đó.

## 4. Lý do vì sao chọn mô hình RoBERTa

Hiệu suất đã được chứng minh: Mô hình Roberta đã được chứng minh là một trong những mô hình ngôn ngữ hiệu quả và mạnh mẽ trong nhiều tác vụ xử lý ngôn ngữ tự nhiên, bao gồm việc nhận diện lỗi chính tả. Nó đã đạt được kết quả tốt trong các nhiệm vụ như xếp hạng câu, phân loại văn bản và dịch máy, và có khả năng học được biểu diễn ngôn ngữ phong phú.

Xử lý ngôn ngữ tự nhiên tiếng Việt: Mô hình Roberta có khả năng xử lý ngôn ngữ tự nhiên tiếng Việt. Việc sử dụng mô hình đã được huấn luyện trước trên dữ liệu tiếng Việt và biểu diễn ngôn ngữ riêng cho tiếng Việt sẽ giúp nó có khả năng nhận diện lỗi chính tả hiệu quả trong văn bản tiếng Việt.

Tính linh hoạt và khả năng tái sử dụng: Mô hình Roberta là một mô hình pre-trained, có khả năng tái sử dụng và dễ dàng tích hợp vào các hệ thống khác nhau. Việc sử dụng mô hình đã được huấn luyện trước như Roberta cho phép ta tiết kiệm thời gian và công sức trong việc xây dựng một mô hình từ đầu.

Hỗ trợ từ cộng đồng và tài liệu phong phú: Mô hình Roberta là một mô hình phổ biến và được sử dụng rộng rãi trong cộng đồng nghiên cứu và công nghiệp. Có sẵn tài liệu phong phú, mã nguồn mở và ví dụ về cách sử dụng mô hình Roberta, giúp chúng ta tận dụng được những kiến thức và kinh nghiệm đã có để triển khai một hệ thống nhận diện lỗi chính tả hiệu quả.

# Chương II : Cơ sở lý thuyết về tiền xử lý dữ liệu và trích chọn đặc trưng

## 1. Tổng quan về tiền xử lý dữ liệu

Trong ứng dụng học máy trong kiểm tra lỗi chính tả Tiếng Việt, tiền xử lý dữ liệu đóng vai trò quan trọng để chuẩn bị dữ liệu đầu vào cho quá trình huấn luyện mô hình.

Quá trình tiền xử lý dữ liệu giúp chuẩn bị dữ liệu văn bản phù hợp cho việc huấn luyện mô hình học máy trong bài toán kiểm tra lỗi chính tả Tiếng Việt. Nó giúp cải thiện khả năng đại diện và xử lý dữ liệu của mô hình, từ đó cải thiện độ chính xác và hiệu suất của quá trình kiểm tra lỗi chính tả.

Trong bài toán sắp tới, chúng tôi sử dụng mô hình RoBERTa để giải quyết vấn đề của mình. Vây nên ở phần này chúng ta sẽ tìm cách xử lý dữ liệu cho phù hợp với mô hình RoBERa. Chi tiết các công đoạn tiền xử lý dữ liệu sẽ được trình bày chi tiết hơn ở phần sau

## 2. Các công đoạn tiền xử lý

Giai đoạn tiền xử lý văn bản là một phần quan trọng và có ảnh hưởng trực tiếp đến độ chính xác của quá trình nhận dạng lỗi chính tả. Tuy nhiên, nó cũng có thể làm tăng thời gian xử lý của hệ thống. Dưới đây là một số bước chính trong quá trình tiền xử lý dữ liệu:

Bài toán sắp tới chúng tôi áp dụng giải thuật BPE để xử lý bài toán nên ở đây chúng ta sẽ tìm hiểu chi tiết hơn về nó :

Các bước tiền xử lý dữ liệu khi sử dụng BPE (Byte Pair Encoding) để mã hóa văn bản thành một biểu diễn mới bao gồm các bước sau:

1. Xây dựng từ điển (Vocabulary Construction): Xây dựng từ điển (dictionary) từ tất cả các từ xuất hiện trong dữ liệu văn bản. Mỗi từ sẽ được đặt vào một entry trong từ điển. Tìm các cặp ký tự (byte pairs) phổ biến nhất trong các từ và xây dựng một bộ từ điển của các cặp ký tự này. Các cặp ký tự này sẽ được sử dụng để thay thế các từ trong dữ liệu văn bản.
2. Tách từ (Tokenization) và mã hóa BPE (BPE Encoding): Áp dụng quy tắc BPE để mã hóa dữ liệu văn bản. Mỗi từ sẽ được tách thành các cặp ký tự liên tiếp và các cặp ký tự này sẽ được thay thế bằng các token tương ứng trong bộ từ điển BPE.
3. Chuyển đổi thành dạng số (Numeric Conversion): Sau khi áp dụng BPE, các token được biểu diễn dưới dạng chuỗi các ký tự. Để đưa chúng vào mô hình học máy, cần thực hiện bước chuyển đổi thành dạng số. Có thể sử dụng một bộ từ điển (dictionary) để ánh xạ các token thành các số nguyên duy nhất hoặc biểu diễn chúng dưới dạng các đặc trưng số học.
4. Chuẩn hóa kích thước dữ liệu(Normalize data size): Để đảm bảo đầu vào cho mô hình có cùng kích thước, các câu văn có thể được thêm padding (điền thêm) hoặc cắt bớt (truncation) để có độ dài đồng nhất, đồng thời thêm các chỉ số đầu cuối (start/end tokens), padding và truncate. Điều này giúp mô hình nhận được đầu vào có kích thước đồng nhất và xử lý hiệu quả.

Các bước trên đại diện cho quá trình tiền xử lý dữ liệu khi áp dụng BPE. Bước chuyển đổi thành dạng số có thể khác nhau tùy thuộc vào yêu cầu cụ thể của bài toán và mô hình học máy được sử dụng.

### 2.1. Xây dựng từ điển (Vocabulary Construction)

Quá trình xây dựng từ điển trong BPE (Byte Pair Encoding) là một phần quan trọng trong việc áp dụng phương pháp này để mã hóa văn bản. Tổng quan về quá trình xây dựng từ điển trong BPE như sau:

1. Tách từ (Tokenization): Dữ liệu văn bản ban đầu được tách thành các từ riêng biệt. Các từ này sẽ được coi là các "từ" cơ bản trong quá trình xây dựng từ điển.
2. Tính tần suất xuất hiện: Đếm số lần xuất hiện của mỗi từ trong dữ liệu văn bản. Tần suất

này sẽ được sử dụng để xác định các cặp ký tự phổ biến.

1. Xây dựng từ điển ban đầu: Khởi tạo từ điển ban đầu bằng cách đặt mỗi từ là một entry trong từ điển. Mỗi entry sẽ chứa từ đó và tần suất xuất hiện của từ đó trong dữ liệu.
2. Xác định cặp ký tự phổ biến: Trong quá trình xây dựng từ điển, lặp lại các bước sau cho một số lần xác định trước hoặc cho đến khi đạt được số lượng cặp ký tự phổ biến mong muốn:

* Tìm cặp ký tự phổ biến nhất trong từ điển dựa trên tần suất xuất hiện.
* Ghép cặp ký tự thành một ký tự mới.
* Cập nhật từ điển bằng cách thay thế các cặp ký tự phổ biến bằng ký tự mới và cập nhật tần suất xuất hiện của ký tự mới.

1. Chuyển đổi thành dạng số: Các mã số tương ứng với các cặp ký tự BPE được gán cho từng từ trong văn bản. Điều này tạo ra một biểu diễn số hóa của văn bản ban đầu, trong đó mỗi từ được biểu diễn bằng một mã số.
2. Hoàn thành quá trình xây dựng từ điển: Quá trình xác định cặp ký tự phổ biến và cập nhật từ điển sẽ tiếp tục cho đến khi số lượng cặp ký tự phổ biến đạt đủ hoặc đạt được một điều kiện dừng khác.

Kết quả cuối cùng của quá trình xây dựng từ điển là một bộ từ điển BPE gồm các cặp ký tự phổ biến và ký tự mới tương ứng. Bộ từ điển này sẽ được sử dụng trong quá trình mã hóa và giải mã văn bản sử dụng BPE.



*Ví dụ về 1 bộ từ điển sau khi đã được xây dựng (file dict.txt)*

### 2.2. Tách từ (Tokenization) và mã hóa BPE (BPE Encoding)

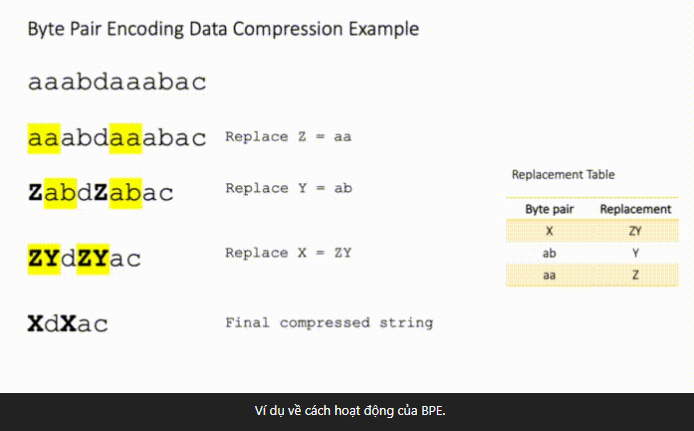
BPE (Byte-Pair Encoding) là một thuật toán phổ biến trong xử lý ngôn ngữ tự nhiên và tiền xử lý dữ liệu văn bản. Nó được sử dụng để tách các từ thành các subword units.

Quá trình BPE bắt đầu bằng việc chia toàn bộ từ vựng ban đầu thành các ký tự riêng lẻ. Sau đó, thuật toán sẽ lặp lại hai bước cho đến khi đạt được số lượng subword units mong muốn:

Tính toán tần suất xuất hiện của các cặp subword units liền kề. Cặp có tần suất xuất hiện cao nhất sẽ được chọn để tạo thành một subword mới.

Thay thế các cặp subword units được chọn bằng subword mới, và cập nhật từ vựng. Quá trình này tiếp tục cho đến khi đạt được số lượng subword units mong muốn.

Kết quả là toàn bộ từ vựng sẽ được chia thành các subword units, bao gồm các từ nguyên gốc, các từ ghép và các subword nhỏ hơn. Việc này giúp giảm độ phức tạp của từ vựng và giữ được thông tin cấu trúc từ ngữ, đồng thời hỗ trợ việc xử lý các từ mới hoặc chưa biết trước.



BPE giúp giảm kích thước từ điển, giảm độ phức tạp của dữ liệu và cung cấp một phương pháp biểu diễn từ vựng tốt hơn. Điều này hữu ích trong các tác vụ xử lý ngôn ngữ tự nhiên như dịch máy, gợi ý từ khóa và nhiều tác vụ khác.

### 2.3. Chuyển dữ liệu về dạng số (Numeric Conversion)

Quá trình này giúp chuyển đổi từ văn bản sang các token tương ứng trong bộ từ điển BPE, dựa trên các quy tắc đã được thiết lập trước.

Mã hóa BPE (Byte Pair Encoding) là một phương pháp mã hóa dữ liệu văn bản thành các token. Quá trình bắt đầu bằng cách tách mỗi từ trong văn bản thành các cặp ký tự liên tiếp. Ví dụ, từ "apple" sẽ được chia thành các cặp ký tự "ap", "pp", "pl", "le". Sau đó, các cặp ký tự này sẽ được thay thế bằng các token tương ứng trong bộ từ điển BPE.

Bộ từ điển BPE chứa các từ và token tương ứng và đã được xây dựng ở trên. Trong quá trình mã hóa, các cặp ký tự được thay thế bằng các token tương ứng trong bộ từ điển. Ví dụ, cặp ký tự "và" có thể được thay thế bằng token “41673512 và "," có thể được thay thế bằng token “166046266” như đã được định nghĩa trong từ điển và cứ tiếp tục như vậy.

Quá trình mã hóa BPE giúp giảm kích thước của từ vựng và đồng thời giữ lại thông tin quan trọng trong dữ liệu văn bản. Thay vì biểu diễn mỗi từ riêng lẻ, nó biểu diễn các cặp ký tự liên tiếp và thay thế chúng bằng các token tương ứng. Điều này có thể giúp tăng tốc độ xử lý và giảm độ phức tạp trong quá trình xử lý dữ liệu văn bản.

Tóm lại, mã hóa BPE là quá trình chuyển đổi dữ liệu văn bản thành các token tương ứng trong bộ từ điển BPE bằng cách tách từ thành các cặp ký tự liên tiếp và thay thế chúng. Quá trình này có thể cải thiện hiệu suất xử lý dữ liệu và giảm kích thước của từ vựng.

### 2.4. Chuẩn hoá kích thước dữ liệu(Normalize data size)

Khi xử lý dữ liệu trong xử lý ngôn ngữ tự nhiên, chúng ta có thể thực hiện nhiều bước để chuẩn hóa kích thước dữ liệu, bao gồm việc thêm các chỉ số đầu cuối (start/end tokens), padding và truncate. Quá trình này giúp đảm bảo rằng tất cả các mẫu dữ liệu có cùng kích thước, đồng thời đánh dấu sự bắt đầu và kết thúc của văn bản.

Cụ thể, quy trình xử lý dữ liệu có thể được thực hiện như sau:

1. Thêm các chỉ số đầu cuối (start/end tokens):

• Chúng ta thêm một chỉ số đầu (start token) vào đầu mỗi mẫu dữ liệu để đánh dấu sự bắt đầu của văn bản.

• Chúng ta cũng thêm một chỉ số cuối (end token) vào cuối mỗi mẫu dữ liệu để đánh dấu sự kết thúc của văn bản.

• Các chỉ số đầu cuối này giúp mô hình học máy hiểu được phạm vi của văn bản và hỗ trợ trong việc dự đoán.

2. Padding:

• Khi sử dụng padding, chúng ta điều chỉnh kích thước của các mẫu dữ liệu để chúng có cùng độ dài.

• Để thực hiện padding, ta thêm các giá trị padding vào cuối mỗi mẫu dữ liệu.

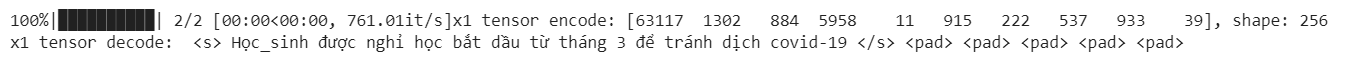
• Một giá trị padding thường được chọn, ví dụ như số 0, được sử dụng để biểu thị giá trị không mang ý nghĩa trong dữ liệu.

3. Truncate:

• Truncate là quá trình cắt bỏ các phần tử dư thừa của dữ liệu để đạt được kích thước giới hạn.

• Khi một mẫu dữ liệu có độ dài lớn hơn kích thước đã định trước, ta cắt bỏ các phần tử thừa để đạt kích thước mong muốn.

• Điều này có thể xảy ra khi một mô hình chỉ chấp nhận một kích thước cố định cho đầu vào.



*Ví dụ về quá trình chuẩn hóa kích thước dữ liệu*

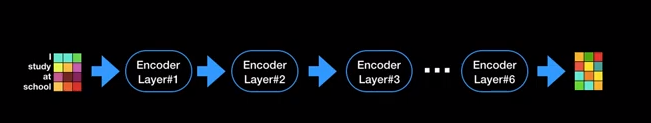
Kết hợp các bước trên, chúng ta có thể xử lý dữ liệu bằng cách thêm các chỉ số đầu cuối, thực hiện padding và truncate để đạt được kích thước chuẩn cho mỗi mẫu dữ liệu. Quá trình này đảm bảo rằng dữ liệu có cùng kích thước và chuẩn hóa, giúp tạo điều kiện cho việc huấn luyện và sử dụng mô hình học máy.

* 1. Trích xuất đặc trưng

Trích xuất đặc trưng (feature extraction) là quá trình rút trích thông tin quan trọng từ dữ liệu đầu vào và biến đổi nó thành một tập hợp các đặc trưng có ý nghĩa. Trích xuất đặc trưng là một bước quan trọng trong xử lý dữ liệu và được áp dụng rộng rãi trong nhiều lĩnh vực, bao gồm xử lý ngôn ngữ tự nhiên, thị giác máy tính, âm thanh và nhiều lĩnh vực khác.

Trong mô hình học sâu như RoBERTa, trích xuất đặc trưng thường được thực hiện bằng cách sử dụng các lớp mã hóa (encoder layers) để biểu diễn thông tin quan trọng trong dữ liệu đầu vào. Các lớp mã hóa như lớp Self-Attention và mạng thần kinh tiếp tục trong kiến trúc Transformer của RoBERTa giúp học các đặc trưng phức tạp từ dữ liệu ngôn ngữ. Quá trình này cho phép RoBERTa học được sự tương quan giữa các từ và cấu trúc ngữ pháp trong văn bản.

Khi trích xuất đặc trưng từ RoBERTa, chúng ta sử dụng các lớp mã hóa để biểu diễn thông tin ngữ cảnh và ý nghĩa của từng từ và câu trong dữ liệu. Các biểu diễn ngữ cảnh này có thể được sử dụng để thực hiện các tác vụ như phân loại văn bản, dịch máy, phân tích ý kiến và nhiều tác vụ khác trong xử lý ngôn ngữ tự nhiên.



*Kiến trúc gồm nhiều layers tại encoder của model BERT. Mô hình huấn luyện từ RoBERTa cho phép ta trích suất các đặc trưng từ những layers của encoder. Có thể là layer cuối hoặc toàn bộ các layers.*

Tóm lại, trích xuất đặc trưng trong RoBERTa là quá trình sử dụng các lớp mã hóa để biểu diễn thông tin quan trọng trong dữ liệu ngôn ngữ và tạo ra các đặc trưng có ý nghĩa để sử dụng trong các tác vụ xử lý ngôn ngữ tự nhiên.

* 1. Kết luận chương

Trong quá trình xử lý dữ liệu, chúng ta đã thực hiện một loạt các công việc để chuẩn bị dữ liệu cho mô hình RoBERTa. Chúng ta đã áp dụng giải thuật Byte Pair Encoding (BPE) để xử lý dữ liệu ngôn ngữ tự nhiên. BPE là một phương pháp nén từ vựng mạnh mẽ, đã được sử dụng rộng rãi trong xử lý ngôn ngữ tự nhiên và dịch máy.

Bằng cách áp dụng BPE, chúng ta đã thực hiện việc nén từ vựng của văn bản. Quá trình này đã giúp giảm số lượng từ vựng và làm giảm kích thước dữ liệu, đồng thời tạo ra các đại diện nén cho các từ phổ biến và các từ ghép trong ngôn ngữ.

BPE cung cấp lợi ích đáng kể cho việc xử lý ngôn ngữ tự nhiên. Đầu tiên, nó giúp giảm không gian lưu trữ và tăng tốc độ xử lý dữ liệu. Thứ hai, nó làm giảm hiện tượng OOV (Out-Of-Vocabulary) bằng cách tạo ra các từ mới thông qua ghép các byte phổ biến. Điều này giúp mô hình hiểu được các từ mới mà nó chưa bao giờ gặp phải trong quá trình huấn luyện.

Trong chương tiền xử lý dữ liệu, việc áp dụng BPE đã cung cấp một cách tiếp cận hiệu quả để xử lý dữ liệu ngôn ngữ tự nhiên. Điều này đã đóng góp vào việc cải thiện hiệu suất và độ chính xác của mô hình học máy khi áp dụng vào bài toán xử lý ngôn ngữ tự nhiên.

# Chương III: CƠ SỞ LÝ THUYẾT VỀ RoBERTa

## 1. Giới thiệu

RoBERTa (Robustly optimized BERT approach) là một mô hình ngôn ngữ dựa trên kiến trúc Transformer, được phát triển bởi Facebook AI Research (FAIR) vào năm 2019. Mô hình RoBERTa là một phiên bản được tinh chỉnh và cải tiến của mô hình BERT (Bidirectional Encoder Representations from Transformers).

Mô hình RoBERTa được huấn luyện trên một lượng lớn dữ liệu văn bản từ nguồn Internet, gồm khoảng 160GB dữ liệu tiếng Anh và dữ liệu từ 50GB dữ liệu tiếng khác. Điều này giúp RoBERTa hiểu ngôn ngữ phong phú và có khả năng đại diện cho nhiều khía cạnh ngôn ngữ.

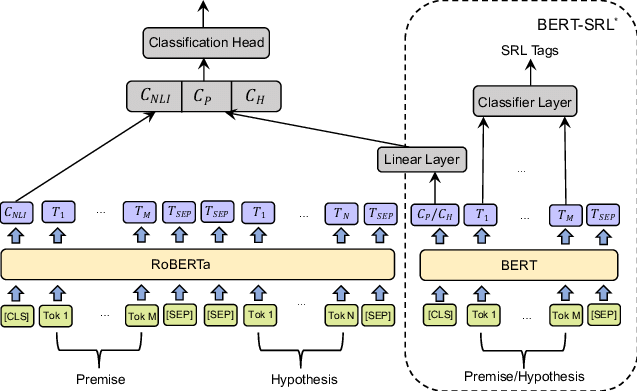
Trong quá trình huấn luyện, RoBERTa sử dụng một số phương pháp tối ưu hóa để cải thiện hiệu suất của mô hình. Đầu tiên, RoBERTa loại bỏ việc dự đoán từ đánh dấu (masked language modeling) như trong BERT và thay vào đó, nó sử dụng mã nhúng từ vựng ngẫu nhiên. Điều này giúp mô hình học được biểu diễn ngôn ngữ tổng quát hơn và tránh sự phụ thuộc quá mức vào thông tin bị ẩn trong quá trình huấn luyện.

Mô hình RoBERTa cũng sử dụng việc xếp chồng các tác vụ (multi-task learning) và tăng cường dữ liệu ngẫu nhiên (random data augmentation) để cải thiện hiệu suất. Việc xếp chồng các tác vụ có nghĩa là mô hình được huấn luyện trên nhiều tác vụ ngôn ngữ khác nhau, bao gồm dự đoán từ tiếp theo, phân loại câu, dự đoán lớp masked và dự đoán liên kết câu. Điều này giúp mô hình học được các đặc trưng ngôn ngữ đa dạng và có khả năng tổng quát hóa tốt hơn.

Mô hình RoBERTa cũng giữ lại cấu trúc Transformer của BERT với một bộ mã hóa sử dụng các lớp self-attention và mạng nơ-ron truyền thẳng. Cấu trúc này cho phép mô hình hiểu mối quan hệ giữa các từ trong câu và tạo ra biểu diễn ngôn ngữ sâu sắc và giàu ý nghĩa.

## 2. Cấu trúc

RoBERTa (Robustly Optimized BERT Pretraining Approach) là một mô hình ngôn ngữ dựa trên kiến trúc Transformer. Nó được xây dựng dựa trên kiến trúc của BERT (Bidirectional Encoder Representations from Transformers), nhưng có một số cải tiến và tối ưu hóa.

Cấu trúc của RoBERTa bao gồm nhiều lớp Transformer được xếp chồng lên nhau. Mỗi lớp Transformer gồm hai phần chính: Encoder và Decoder. Tuy nhiên, trong trường hợp của RoBERTa, chỉ sử dụng phần Encoder. Mỗi lớp Encoder trong RoBERTa bao gồm nhiều lớp con, bao gồm Self-Attention và Feed-Forward Neural Network.

Self-Attention là một phần quan trọng trong kiến trúc Transformer. Nó cho phép mô hình RoBERTa tập trung vào các phần quan trọng của câu đầu vào trong quá trình mã hóa thông tin. Trong quá trình này, mỗi từ trong câu đầu vào được biểu diễn bằng một vector. Các vector này được sử dụng để tính toán các trọng số attention, giúp mô hình "chú ý" đến các từ quan trọng và xác định mức độ quan trọng của chúng trong việc xử lý câu.

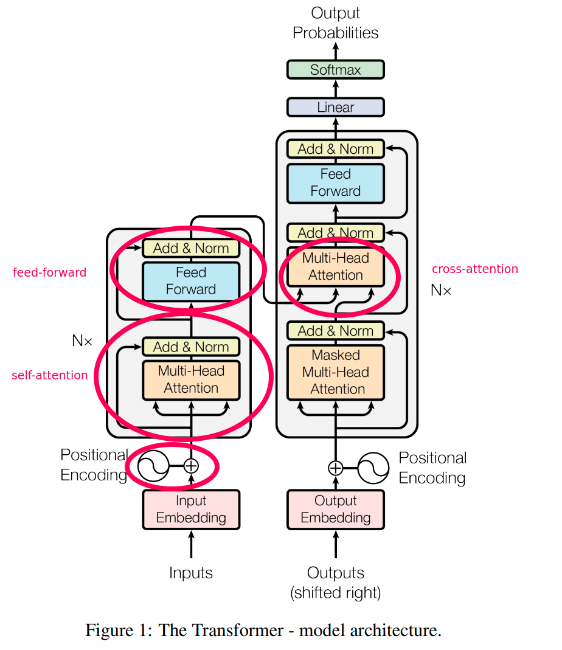
Feed-Forward Neural Network là một lớp mạng thần kinh truyền thẳng, nó được sử dụng để biến đổi đầu ra của lớp Self-Attention. Lớp này bao gồm các lớp tuyến tính và hàm kích hoạt phi tuyến tính để trích xuất thông tin và tạo ra biểu diễn nâng cao cho các từ trong câu.

Mô hình RoBERTa được huấn luyện trên một lượng lớn dữ liệu văn bản từ nhiều nguồn khác nhau, đảm bảo nắm bắt được các đặc trưng ngôn ngữ phổ biến. Sau khi được huấn luyện, RoBERTa có khả năng hiểu và biểu diễn ngôn ngữ tự nhiên.

## 3. Cách hoạt động

RoBERTa (Robustly Optimized BERT Pretraining Approach) là một biến thể của mô hình BERT (Bidirectional Encoder Representations from Transformers) được cải tiến và tối ưu hóa. RoBERTa sử dụng kiến trúc transformer để xử lý ngôn ngữ tự nhiên.

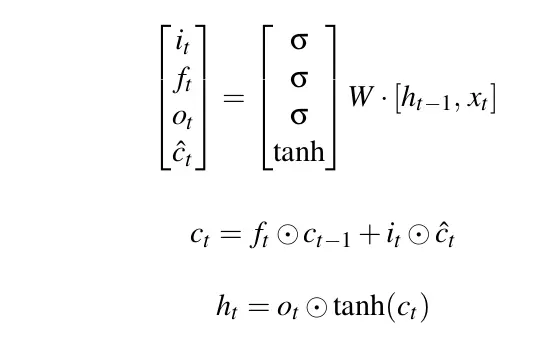
Cách hoạt động của RoBERTa gồm các bước sau:

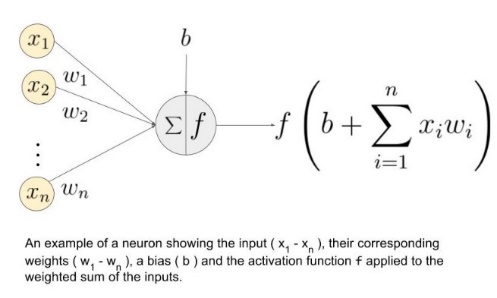
1. Tokenization: Văn bản đầu vào được chia thành các từ (token) riêng lẻ. Mỗi từ được mã hóa thành một vector số.

2. Embedding: Mỗi từ được mã hóa thành một vector số thông qua một bộ mã hóa từ (word embedding). Các vector này biểu diễn ngữ nghĩa và cú pháp của từ.

3. Positional Encoding: Để bảo toàn thông tin về vị trí từ trong câu, RoBERTa sử dụng mã hóa vị trí (positional encoding) để thêm thông tin về vị trí vào vector từ.

4. Encoder Layers: RoBERTa sử dụng nhiều lớp mã hóa (encoder layers) để xử lý đầu vào. Mỗi lớp mã hóa bao gồm hai thành phần chính: self-attention và feed-forward neural network.

• Self-Attention: Mỗi lớp mã hóa trong RoBERTa sử dụng self-attention để tập trung vào các phần quan trọng của câu. Trong quá trình self-attention, mỗi từ được biểu diễn bằng ba vector: vector truy vấn (query), vector khóa (key), và vector giá trị (value). Sử dụng ma trận trọng số, RoBERTa tính toán điểm tương đồng giữa các cặp vector này để xác định mức độ quan trọng của từng từ đối với các từ khác trong câu. Kết quả là một ma trận attention, mô tả mức độ quan trọng của các từ đối với nhau.

• Feed-Forward Neural Network: Sau quá trình self-attention, mỗi từ được kết hợp với các thông tin quan trọng từ các từ khác trong câu. Tiếp theo, mỗi từ được đưa qua một mạng neural network truyền thẳng (feed-forward neural network) để xử lý thông tin và tạo ra biểu diễn mới cho từ đó.

5. Cross-attention, còn được gọi là multi-head attention cho phép mô hình tập trung vào các phần khác nhau của các đầu vào khác nhau và tạo ra kết quả dựa trên sự tương tác giữa chúng. Trong cross-attention, chúng ta có một bộ ba đầu vào gồm: một vector trạng thái truy vấn (query), một tập hợp các vector trạng thái khóa (keys) và một tập hợp các vector trạng thái giá trị (values). Sử dụng vector trạng thái truy vấn để truy vấn thông tin từ các vector trạng thái khóa và sau đó sử dụng kết quả để tính toán trọng số cho các vector trạng thái giá trị. Kết quả cuối cùng được tính toán bằng cách lấy tổng có trọng số của các vector trạng thái giá trị.

6. Linear (tuyến tính) là một phép toán tuyến tính được áp dụng trên đầu ra của một lớp trước đó. Thông thường, linear layer được sử dụng để biến đổi không gian đặc trưng của dữ liệu. Nó bao gồm các phép nhân ma trận và phép cộng ma trận để ánh xạ đầu vào sang một không gian đặc trưng mới. Linear layer có thể tìm ra các quan hệ tuyến tính giữa các đặc trưng và thực hiện các phép biến đổi phức tạp trên dữ liệu.

7. "Add and Norm" là phương pháp chuẩn hóa đầu ra của lớp tự chú ý trong mạng transformer. Nó bao gồm hai bước:

• Cộng kết quả của tự chú ý với đầu vào ban đầu để kết hợp thông tin từ các lớp trước và thông tin mới.

• Chuẩn hóa kết quả để duy trì phân phối và tránh vấn đề gradient.

"Add and Norm" cải thiện hiệu suất và ổn định quá trình huấn luyện của mạng transformer.

8. Softmax là một hàm kích hoạt thường được sử dụng trong mạng nơ-ron để chuyển đổi đầu ra của một lớp thành một phân phối xác suất

Tóm lại, RoBERTa sử dụng kiến trúc transformer với self-attention và feed-forward neural network để xử lý ngôn ngữ tự nhiên. Qua các lớp mã hóa, RoBERTa có khả năng hiểu và biểu diễn các mối quan hệ phức tạp giữa các từ trong câu, giúp nó thực hiện tốt trong nhiều tác vụ ngôn ngữ khác nhau.

# Chương IV. MÔ TẢ BÀI TOÁN

## 1. Giới thiệu bài toán

Chúng ta sẽ tiến hành huấn luyện một mô hình phoBERT để phân loại văn bản thành hai loại "đúng" và "sai ngữ pháp"

PhoBERT là một phiên bản của mô hình Roberta được điều chỉnh và huấn luyện đặc biệt cho tiếng Việt. Nó được phát triển bởi các nhà nghiên cứu từ Đại học Công nghệ Tokyo và VINAI Research. PhoBERT sử dụng kiến trúc Transformer và đã được huấn luyện trên một lượng lớn dữ liệu tiếng Việt từ các nguồn như báo chí, Internet và đời sống hàng ngày. Mô hình này có thể được sử dụng để xử lý nhiều tác vụ ngôn ngữ tiếng Việt, bao gồm cả phân loại văn bản, dịch máy, và nhận diện thực thể.

Bài toán phân loại văn bản thành hai loại "đúng" và "sai ngữ pháp" được thực hiện nhằm đáp ứng các mục tiêu sau:

• Đánh giá và cải thiện chất lượng ngôn ngữ: Phân loại ngữ pháp giúp đánh giá và sửa lỗi ngôn ngữ trong văn bản, từ đó cải thiện chất lượng và sự chính xác của nó.

• Hỗ trợ học ngôn ngữ: Bài toán này hỗ trợ việc học ngôn ngữ bằng cách kiểm tra và đánh giá ngữ pháp của câu văn bản, từ đó nâng cao kiến thức và kỹ năng viết văn.

• Tích hợp vào ứng dụng xử lý ngôn ngữ tự nhiên: Mô hình phân loại ngữ pháp có thể tích hợp vào các ứng dụng xử lý ngôn ngữ tự nhiên để kiểm tra và cải thiện chất lượng ngôn ngữ tự động.

## 2. Dữ liệu

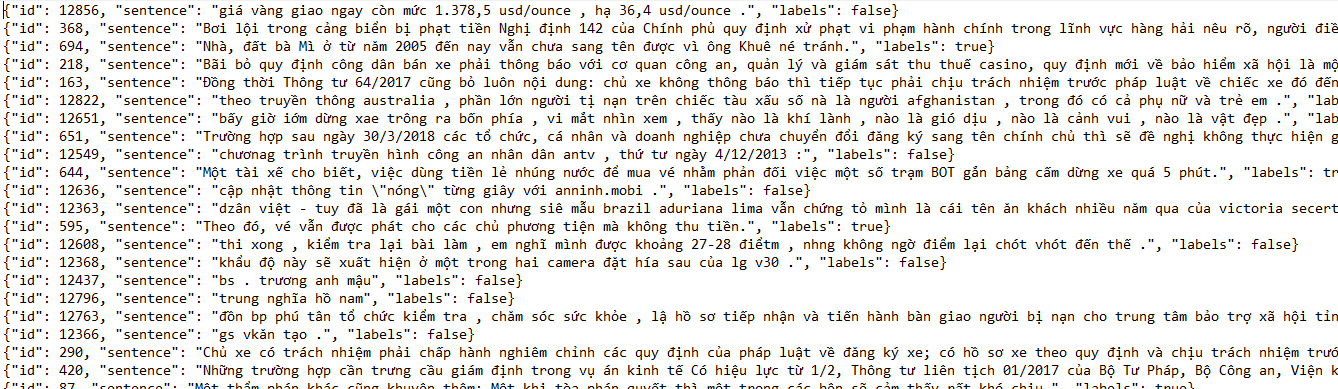
Dữ liệu được sử dụng trong bài toán này gồm 15,000 câu văn bản, bao gồm 5,000 câu được phân loại là "đúng" và 10,000 câu được phân loại là "sai ngữ pháp". Dữ liệu này được thu thập từ các nguồn như bài báo, internet và cuộc sống hàng ngày.



Việc lấy dữ liệu từ các nguồn đa dạng như vậy giúp đảm bảo tính phổ biến và đại diện của các lỗi ngữ pháp trong dữ liệu. Điều này giúp mô hình huấn luyện trở nên đa dạng và có khả năng xử lý các lỗi ngữ pháp phổ biến trong thực tế.

Dữ liệu này sẽ được sử dụng để huấn luyện mô hình Roberta với mục tiêu phân loại chính xác các câu văn bản là "đúng" hoặc "sai ngữ pháp".

Dữ liệu được lưu vào file ‘data.json’ với định dạng như dưới :

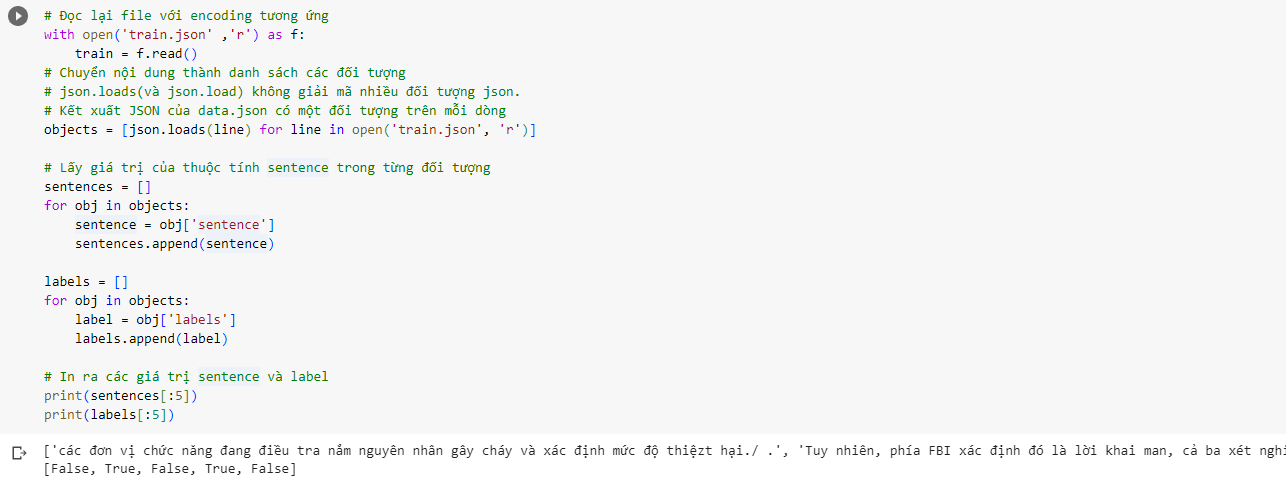


### 2.1. Đọc và chia dữ liệu

Sau khi đã download dữ liệu về, chúng ta sẽ đọc file json và lưu các bài báo vào những list chứa nội dung và nhãn tương ứng theo 2 folders train và test.



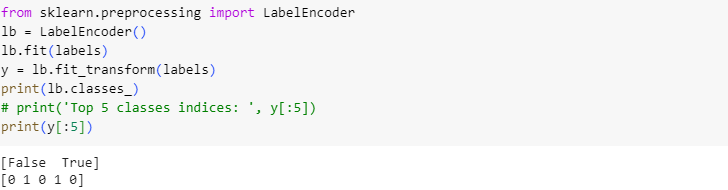
Khởi tạo các mảng dữ liệu để lưu trữ riêng biệt thông tin của các sentences và các labels



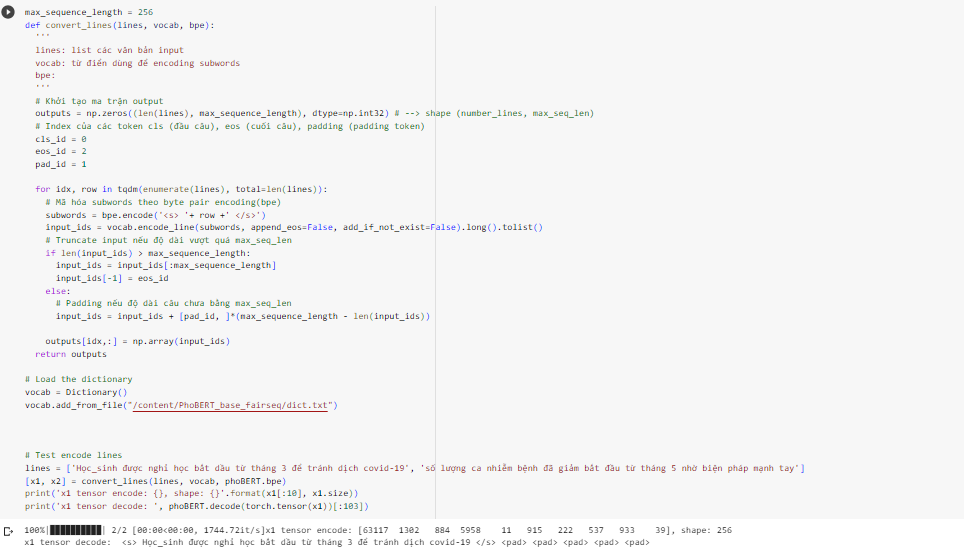
### 2.2. Tokenize nội dung và labels

Để mã hoá các labels chúng ta gọi phương thức fit trên LabelEncoder để tìm hiểu và học cách mã hóa các nhãn từ danh sách labels. Quá trình này sẽ xác định các giá trị nhãn duy nhất và xây dựng bản đồ tương ứng giữa các giá trị nhãn và các giá trị số.

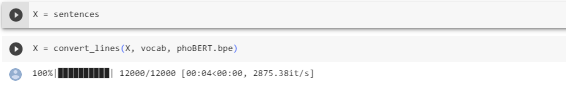
Gọi phương thức fit\_transform trên LabelEncoder để thực hiện mã hóa các nhãn trong biến labels. Quá trình này sẽ chuyển đổi các nhãn thành các giá trị số tương ứng dựa trên bản đồ mã hóa đã học từ bước trước.



Tiếp theo ta sẽ tokenize các câu văn sang chuỗi index và padding câu văn về cũng một độ dài.



Như vậy ta thấy rằng các câu văn đã được encode về token index. Từ token index có thể decode ngược trở lại thành câu input sau khi đã thêm các token đặc biệt đánh dấu vị trí bắt dầu: <s>, kết thúc: </s> câu và các vị trí nằm ngoài câu: <pad>. Ta sẽ token toàn bộ câu input sang index như sau:



Như vậy 2 biến X và Y hiện tại đã lần lượt được lưu các giá trị đã được mã hoá của các câu và các nhãn.

### 2.3. Load model RoBERTa



Chúng ta cần sử dụng mô hình Pre-trained phoBERT để phân loại văn bản vào 2 classes khác nhau. Bước này được thực hiện bằng cách tải mô hình PhoBERT đã được huấn luyện trước và đăng ký một đầu phân loại mới. Sau đó, chuỗi văn bản được mã hóa thành các chỉ số token và đưa vào mô hình để dự đoán xác suất của mỗi class. Kết quả là các giá trị log của xác suất được dự đoán cho mỗi class.

Sau đó chúng ta thực hiện việc chọn ra class có xác suất dự đoán cao nhất từ ma trận xác suất đã tính được. Bằng cách sử dụng hàm argmax từ thư viện numpy, ta chọn ra chỉ số của class có xác suất cao nhất trong mỗi hàng của ma trận logprobs. Kết quả được in ra màn hình là nhãn dự đoán 1(đúng) hoặc 0(sai).

### 2.4. Huấn luyện mô hình

#### 2.4.1. Xây dựng hàm đánh giá mô hình theo 2 metric là accuracy và f1\_score.





Hàm evaluate được sử dụng để tính toán và trả về độ chính xác (acc) và điểm F1 từ các giá trị dự đoán và nhãn thực tế.

Hàm validate được dung để tính toán độ chính xác và điểm F1 ở tệp validation

Mô hình được đánh giá bằng cách chuyển sang chế độ eval().

Với mỗi batch dữ liệu từ valid\_loader, mô hình được sử dụng để dự đoán giá trị đầu ra.

Các giá trị đầu ra được chuyển thành xác suất bằng hàm torch.exp().

Độ chính xác và điểm F1 được tính toán từ các giá trị dự đoán và nhãn thực tế.

Trung bình của độ chính xác và điểm F1 được tính và trả về.

#### 2.4.2. Khởi tạo DataLoader



Đoạn code sử dụng phương pháp Stratified K-fold cross-validation để chia dữ liệu thành 5 fold (phân chia theo tỷ lệ tương tự cho mỗi lớp).

Vòng lặp for duyệt qua từng fold để huấn luyện và đánh giá mô hình. Biến FOLD được sử dụng để chỉ định fold được huấn luyện (nếu không phải fold này, sẽ bỏ qua).

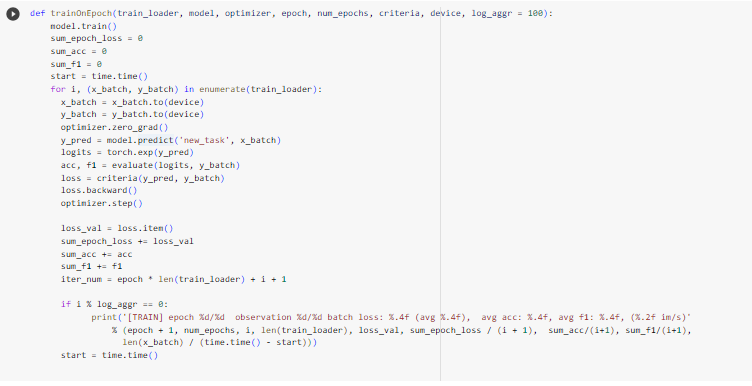
Tạo ra các tập dữ liệu train và validation từ các chỉ mục tương ứng với fold hiện tại.

Sử dụng lớp DataLoader để tạo các vòng lặp huấn luyện và đánh giá (train\_loader và valid\_loader) từ các tập dữ liệu train và validation tương ứng.

Batch size (BATCH\_SIZE) được sử dụng để xác định số lượng mẫu trong mỗi batch trong quá trình huấn luyện và đánh giá.

Quá trình huấn luyện và đánh giá sẽ được thực hiện cho fold được chỉ định và các thông tin về quá trình huấn luyện sẽ được in ra màn hình.

#### 2.4.3. Hàm huấn luyện mô hình trên từng epoch



Mô hình được chuyển sang chế độ train() để bắt đầu quá trình huấn luyện.

Đối với mỗi batch từ train\_loader, dữ liệu đầu vào và nhãn được chuyển sang thiết bị (device) để tính toán trên GPU nếu có.

Bộ tối ưu hóa (optimizer) được đặt về 0 với optimizer.zero\_grad() để xóa các gradient tích tụ từ vòng lặp trước đó.

Mô hình được sử dụng để dự đoán giá trị đầu ra.

Các giá trị đầu ra được chuyển thành xác suất bằng hàm torch.exp().

Độ chính xác và điểm F1 được tính toán từ các giá trị dự đoán và nhãn thực tế.

Mất mát (loss) được tính toán bằng hàm criteria().

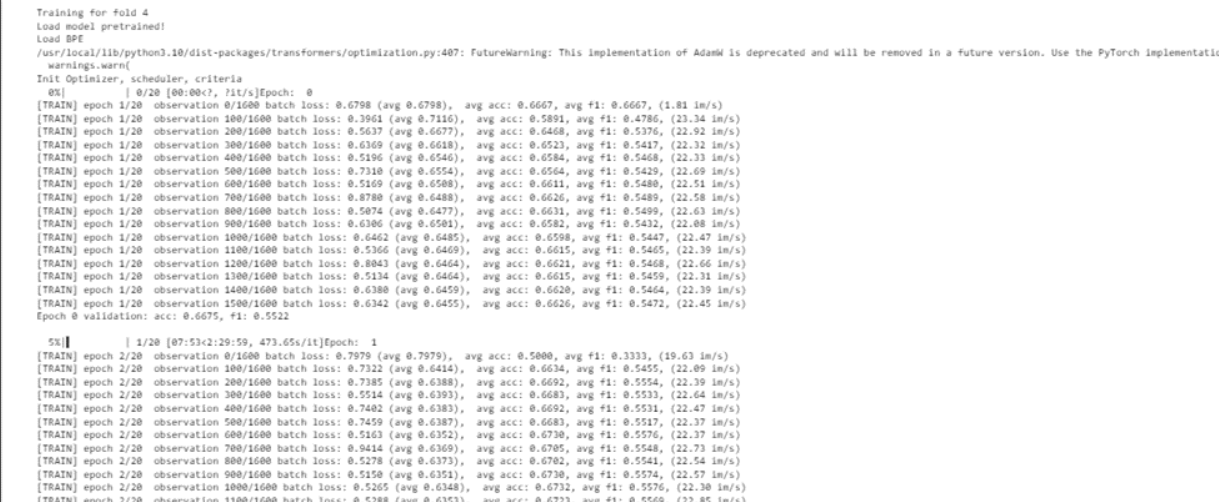
Đạo hàm của mất mát được tính toán thông qua loss.backward().

Bộ tối ưu hóa được cập nhật thông qua optimizer.step().

Các giá trị mất mát, độ chính xác và điểm F1 được cộng dồn để tính tổng cho epoch hiện tại.

* Số lượng quan sát và batch hiện tại được tính để in thông tin tiến trình.
* Thời gian thực hiện batch tính toán được ghi nhận để tính tốc độ tính toán.
* Thông tin về mất mát, độ chính xác, điểm F1 và tốc độ tính toán được in ra màn hình theo từng batch.

Sau khi thực hiện lần lượt các bước trên mô hình sẽ chuyển sang trạng thái huấn luyện.



## 3. Đánh giá kết quả

Chúng ta sẽ sử dụng các độ đo như accuracy và F1-score để đánh giá mô hình phân loại.

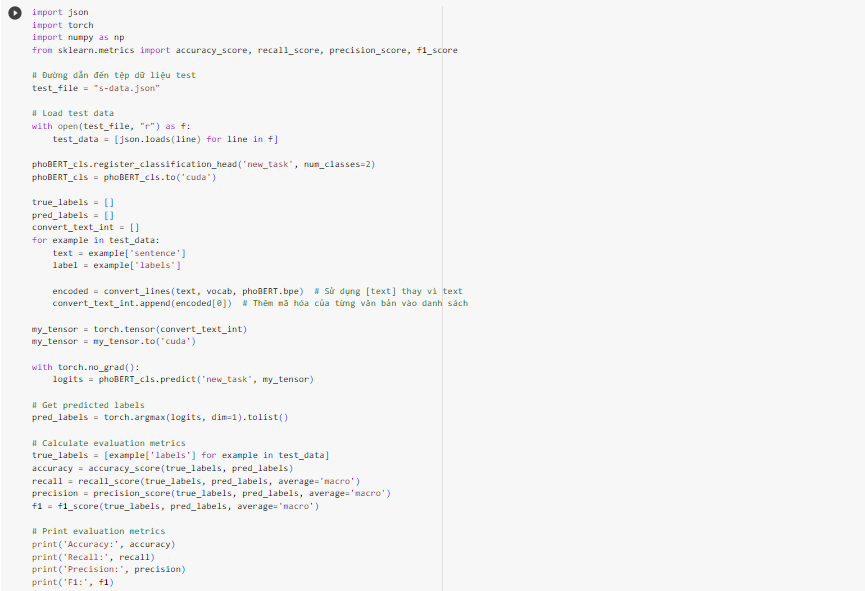
Accuracy (độ chính xác) được tính bằng tỷ lệ giữa số lượng dự đoán đúng và tổng số mẫu trong tập dữ liệu. Nó cho thấy khả năng của mô hình trong việc phân loại chính xác các mẫu.

F1-score là một độ đo kết hợp giữa precision và recall.

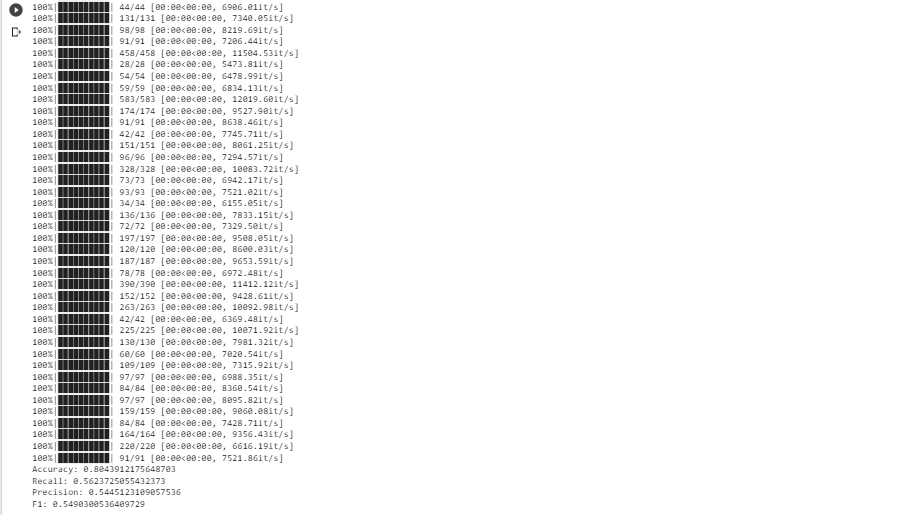
* Precision (độ chính xác dương tính) là tỷ lệ giữa số lượng dự đoán đúng positive và tổng số dự đoán positive.
* Recall (độ phủ) là tỷ lệ giữa số lượng dự đoán đúng positive và tổng số thực tế positive.

F1-score là một trung bình điều hòa của precision và recall, nhằm đánh giá hiệu suất của mô hình trên cả hai lớp.

Trong quá trình đánh giá, chúng ta sẽ sử dụng tập dữ liệu kiểm tra hoặc tập dữ liệu test đã được tách ở trên để đánh giá mô hình. Mô hình sẽ được đưa vào chế độ đánh giá (evaluation mode) để tắt các lớp dropout và không tính gradient trong quá trình đánh giá.



Kết quả đánh giá sẽ được báo cáo dưới dạng các giá trị số như accuracy và F1-score, cùng với các thông số khác liên quan đến quá trình huấn luyện và đánh giá, như số lượng mẫu, batch loss và thời gian thực hiện



Kết quả được thực nghiệm trên tập test



Dựa trên các giá trị trên, có thể kết luận rằng mô hình Roberta dùng để nhận diện lỗi chính tả chưa cho kết quả tốt. Mặc dù accuracy đạt mức cao, nhưng recall, precision và f1-score cho thấy mô hình chưa có hiệu suất tốt trong việc nhận diện và phân loại các lỗi chính tả. Cần tiếp tục nghiên cứu và cải thiện mô hình để đạt được kết quả tốt hơn trong tương lai.

# Chương V. kết luận và kiến nghị

Trong bài báo cáo này, chúng tôi đã tiến hành nghiên cứu về ứng dụng học máy trong kiểm tra lỗi chính tả tiếng Việt và đánh giá mô hình Roberta dùng để nhận diện lỗi chính tả. Từ các kết quả và đánh giá, chúng tôi rút ra một số kết luận quan trọng và đưa ra các kiến nghị phát triển trong tương lai.

* Kết luận:

Mô hình Roberta đã đạt được kết quả khá tích cực với tỷ lệ dự đoán chính xác (accuracy) lên tới 80%. Điều này cho thấy mô hình có khả năng nhận diện và phân loại lỗi chính tả một cách hiệu quả. Tuy nhiên, các chỉ số recall, precision và f1-score cho thấy mô hình còn có một số hạn chế trong việc nhận diện và phân loại chính xác các lỗi chính tả.

* Các kiến nghị:

Mở rộng và cải thiện tập dữ liệu huấn luyện: Để đạt hiệu suất tốt hơn, cần thu thập và tạo ra một tập dữ liệu huấn luyện lớn và đa dạng, phản ánh đầy đủ các lỗi chính tả thực tế trong tiếng Việt. Tập dữ liệu này cần được xử lý và gán nhãn chính xác để đảm bảo chất lượng huấn luyện mô hình.

Nâng cao quá trình tiền xử lý và biểu diễn ngôn ngữ: Việc xử lý văn bản tiếng Việt, bao gồm các quy tắc phân đoạn, tách từ và chuẩn hóa dấu tiếng Việt, có thể được cải thiện để cung cấp thông tin đầy đủ và chính xác hơn cho mô hình. Ngoài ra, việc sử dụng biểu diễn ngôn ngữ phong phú như BERT hoặc mô hình ngôn ngữ tiếng Việt riêng có thể cải thiện khả năng nhận diện lỗi chính tả.

Kết hợp các phương pháp học máy khác nhau: Sử dụng các phương pháp học máy kết hợp như mô hình biến đổi, mạng nơ-ron hồi quy (RNN) hoặc mô hình dịch máy có thể tăng cường hiệu suất của mô hình Roberta trong việc nhận diện lỗi chính tả.

Đánh giá và cải thiện các thông số đánh giá: Cần tiếp tục đánh giá và cải thiện các chỉ số recall, precision và f1-score của mô hình để đảm bảo hiệu suất chính xác và toàn diện trong việc nhận diện lỗi chính tả.

# tài liệu tham khảo

1. IBM - Logistic Regression:

Link: https://www.ibm.com/topics/logistic-regression#:~:text=Resources-,What%20is%20logistic%20regression%3F,given%20dataset%20of%20independent%20variables

Tác giả: IBM

Mô tả: Trang web này cung cấp thông tin về hồi quy logistic, một phương pháp thống kê dùng để dự đoán biến phụ thuộc dựa trên một tập dữ liệu các biến độc lập. Bạn có thể tìm hiểu về định nghĩa, cách thức hoạt động và ứng dụng của hồi quy logistic trong trang web này.

1. Natural-Language-Processing-Specialization (GitHub Repository by amanjeetsahu):

Link: https://github.com/amanjeetsahu/Natural-Language-Processing-Specialization

Tác giả: amanjeetsahu

Mô tả: Kho lưu trữ này có thể chứa các tài liệu và mã nguồn liên quan đến Chuyên ngành Xử lý Ngôn ngữ Tự nhiên (Natural Language Processing). Vì không có thông tin chi tiết nào được cung cấp, vui lòng xem các tài liệu và mã nguồn trong kho lưu trữ để tìm hiểu thêm.

1. Transformer-based Language Models (Factored AI):

Link: https://factored.ai/transformer-based-language-models/

Tác giả: Factored AI

Mô tả: Trang web này cung cấp thông tin về mô hình ngôn ngữ dựa trên Transformer, một mô hình học sâu mạnh mẽ trong lĩnh vực xử lý ngôn ngữ tự nhiên. Bạn có thể tìm hiểu về cơ sở lý thuyết và ứng dụng của mô hình này trong việc xử lý ngôn ngữ tự nhiên.

1. Hugging Face - NLP Course Chapter 6.5 (Hugging Face):

Link: https://huggingface.co/learn/nlp-course/chapter6/5?fw=pt

Tác giả: Hugging Face

Mô tả: Đây là một phần trong khóa học về xử lý ngôn ngữ tự nhiên (NLP) được tổ chức bởi Hugging Face. Trang web cung cấp tài liệu và mã nguồn về chương 6.5, nơi bạn có thể tìm hiểu về các khía cạnh liên quan đến Mô hình Ngôn ngữ Dựa trên Transformer và các ứng dụng của nó trong NLP.