BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI

XÂY DỰNG WEBSITE TRẮC NGHIỆM – PHÂN HỆ: ỨNG DỤNG MÔ HÌNH TRANSFORMER VÀO SỬA LỖI CHÍNH TẢ TIẾNG VIỆT

Sinh viên thực hiện: **Dương Trung Hiền**
Mã số sinh viên: **B1812267**
Khóa: **44**

Cần Thơ, 12/2022

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

LUẬN VĂN TỐT NGHIỆP ĐẠI HỌC
NGÀNH KHOA HỌC MÁY TÍNH

ĐỀ TÀI

XÂY DỰNG WEBSITE TRẮC NGHIỆM – PHÂN HỆ: ỨNG DỤNG MÔ HÌNH TRANSFORMER VÀO SỬA LỖI CHÍNH TẢ TIẾNG VIỆT

Sinh viên thực hiện: **Dương Trung Hiền**
Mã số sinh viên: **B1812267**
Khóa: **44**

Cần Thơ, 12/2022

NHẬN XÉT CỦA GIÁO VIÊN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Cần Thơ, ngày tháng năm 2022

(GVHD ký và ghi rõ họ tên)

LỜI CẢM ƠN

Trong quá trình thực hiện luận văn em đã có được sự giúp đỡ rất nhiều từ thầy Lưu Tiến Đạo. Em xin gửi lời chân thành cảm ơn đến thầy – người đã hướng dẫn tận tình trong quá trình thực hiện luận văn tốt nghiệp này. Nhờ sự giúp đỡ của thầy mà luận văn tốt nghiệp có thể hoàn thành một cách tốt nhất.

Em cũng xin được lời cảm ơn chân thành này đến những thầy cô giảng viên của trường Đại học Cần Thơ, đặc biệt là thầy cô thuộc Khoa CNTT & TT, những người đã truyền đạt những kiến thức và kỹ năng hữu ích trong suốt quá trình học tập.

Em cũng xin được gửi lời cảm ơn đến chân thành đến gia đình và bạn bè đã ủng hộ, tiếp lửa cho em trong suốt quá trình thực hiện niên luận.

Tuy cố nhiều cố gắng trong suốt quá trình thực hiện nhưng không thể tránh khỏi sai sót trong quá trình thực hiện. Em rất mong nhận được đóng góp ý kiến quý báo của quý thầy cô để luận văn tốt nghiệp có thể hoàn thiện hơn.

Cần Thơ, ngày tháng năm 2022

Người viết

Dương Trung Hiền

MỤC LỤC

DANH MỤC BẢNG

DANH MỤC HÌNH

BẢNG DANH SÁCH TỪ VIẾT TẮT

Tên viết tắt	Tên đầy đủ
LSTM	Long Short Term Memory
API	Application Programming Interface
AI	Artificial Intelligence
NLP	Natural Language Processing
RNN	Recurrent Neural Network
FNN	Feed Forward Network
CPU	Central Processing Unit
GPU	Graphics Processing Unit
HTML	Hypertext Markup Language
ReLU	Rectified Linear Unit

ABTRACT

Language is a communication system that makes it easier for people to communicate and exchange information. Each country will have its languages, and Vietnam is no exception. People can exchange data through the two most common forms of speaking and reading. With reading form, people need to write in the text by language, then others can read. However, in writing, people can make mistakes by writing words that don't exist in the dictionary. These words are misspelled words. These misspelled words can make people don't understand the information correctly.

With the evolution of technologies, many tools help to find misspelled words. However, most of these tools only help find misspelled words, and users must replace them themselves. And when texts are too large, and users make many mistakes, it will take a lot of time to fix these. So this thesis was born to help do this automatically to help solve the above problem.

The thesis uses the basic Transformer model supported by the Tensorflow library for training. The model has an accuracy of up to 92,8 percent.

TÓM TẮT

Ngôn ngữ là một hệ thống giao tiếp giúp con người giao tiếp và trao đổi thông tin với nhau một cách dễ dàng hơn. Mỗi quốc gia thường sẽ có một loại quốc ngữ riêng cho quốc gia đó và Việt Nam cũng có riêng cho mình một ngôn ngữ riêng. Con người có thể trao đổi thông tin lẫn nhau bằng 2 hình thức phổ biến nhất là nói và đọc. Đối với hình thức đọc thì con người cần phải thực hiện việc viết thành văn bản bằng ngôn ngữ để người khác có thể đọc được. Tuy nhiên, trong quá trình thực hiện viết văn bản thì thường sẽ mắc các lỗi viết các từ không tồn tại trong từ điển. Các từ này được gọi là từ sai chính tả. Các từ sai chính tả này sẽ dẫn đến việc người đọc không hiểu đúng được thông tin muốn truyền tải.

Với sự phát triển của công nghệ, đã có rất nhiều công cụ giúp phát hiện lỗi chính tả. Tuy nhiên, hầu hết các công cụ này chỉ giúp phát hiện từ sai và bắt buộc người dùng phải tự mình sửa lỗi. Và nếu văn bản quá lớn và mắc quá nhiều lỗi sẽ tốn rất nhiều thời gian để sửa lỗi. Vì vậy đề tài này ra đời giúp thực hiện việc này một cách tự động giúp giải quyết vấn đề trên.

Đề tài sử dụng mô hình Transformer cơ bản được thư viện Tensorflow hỗ trợ để thực hiện huấn luyện. Mô hình cho ra độ chính xác cao lên đến 92,8%.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Viết sai chính tả trong lúc soạn thảo văn bản là một vấn đề luôn xuất hiện ở nhiều môi trường làm việc khác nhau. Việc viết sai chính tả có thể bắt nguồn từ nhiều lý do nhỏ nhặt như: lỗi phần mềm Unicode, đánh máy nhanh, lỗi do người đánh máy,… Với vấn đề này thì từ lâu những phần mềm soạn thảo văn bản như Word đã cung cấp khả năng kiểm tra từ viết sai chính tả, tuy nhiên nó chỉ giới hạn ở việc kiểm tra chính tả và gợi ý từ để thay thế và ta cần phải thay thế từ sai đó một cách thủ công. Đôi khi làm việc thủ công vẫn có thể xảy ra sai sót, như vậy sẽ làm ảnh hưởng rất nhiều nếu như đó là một tài liệu quan trọng. Từ yêu cầu trên đề tài "Ứng dụng mô hình Transformer vào sửa lỗi chính tả tiếng việt" được ra đời với khả năng tự động thay thế các từ sai chính tả giúp cải thiện quá trình làm việc một cách tốt hơn.

2. Lịch sử giải quyết vấn đề

Chủ đề sửa lỗi chính tả không phải là một chủ đề mới, đã có rất nhiều các nghiên cứu, giải pháp, hướng đi dành cho chủ đề này. Dưới đây là một vài nghiên cứu có liên quan:

- "VSEC: Transformer-based Model for Vietnamese Spelling Correction" [1] do nhóm tác giả Dinh-Truong Do, Ha Thanh Nguyen, Thang Ngoc Bui và Hieu Dinh Vo thuộc trường Đại học Công nghệ, Đại học Quốc gia Hà Nội và Viện Khoa học và Công nghệ Tiên tiến Nhật Bản nghiên cứu và thực hiện. Nghiên cứu đã sử dụng mô hình Transformer vào việc sửa chính tả dành cho tiếng Việt với độ chính xác trên tập kiểm tra đạt đến hơn 76%.

- "Deep Learning Approach for Vietnamese Consonant Misspell Correction" [2] do nhóm tác giả Ha Thanh Nguyen, Tran Binh Dang và Le Minh Nguyen thuộc Viện Khoa học và Công nghệ Tiên tiến Nhật Bản nghiên cứu và thực hiện. Sử dụng kiến trúc LSTM và mô hình N-Gram vào sửa lỗi các từ sai có phát âm tương tự nhau trong tiếng Việt.

- "Using Large N-gram for Vietnamese Spell Checking" [3] do nhóm tác giả Nguyen Thi Xuan Huong, Tran Thai Dang, The Tung Nguyen và Anh Cuong

Le thuộc trường Đại học Công nghệ, Đại học Quốc gia Hà Nội nghiên cứu và thực hiện. Sử dụng mô hình N-Gram với Uni-Gram, Bi-Gram và Tri-Gram vào huấn luyện và so sánh độ hiệu quả của cả 3 mô hình trên với nhau.

Ngoài các nghiên cứu thì còn có một số công cụ như:

- Thư viện Phunspell – một thư viện phát hiện các từ sai và gợi ý các từ thay thế của ngôn ngữ Python. Hiện tại Phunspell hỗ trợ hơn 70 ngôn ngữ khác nhau trong đó có tiếng Việt. Phunspell được phát triển dựa trên thư viện Hunspell - một thư viện hiện đang được sử dụng trong một số phần mềm như: Libre Offfice, Open Office, Firefox và Thunder Bird.
- API https://viettelgroup.ai/nlp/api/v1/spell-checking được phát triển bởi Viettel Group AI. Có khả năng nhận diện từ sai và gợi ý các từ thay thế, có thể được sử dụng một cách tiện lợi và dễ dàng hơn vì là một API.

3. Mục tiêu đề tài

Mục tiêu của đề tài "Ứng dụng mô hình Transformer vào sửa lỗi chính tả tiếng Việt" là xây dựng được mô hình máy hình có khả năng tìm và thay thế các từ sai chính tả. Xây dựng thành một API để dễ dàng truy cập và sử dụng, cụ thể là một phần trong việc xây dựng website thi trắc nghiệm.

API sẽ nhận đầu vào là một câu và câu đó sẽ được mã hóa trước khi đưa vào mô hình Transformer để tiến hành tìm kiếm phát hiện các từ sai và thay thế nó. Đầu ra của mô hình là dữ liệu đã được mã hóa, vì thế sẽ cần phải giải mã để thành một câu hoàn chỉnh. Câu này sẽ là kết quả để trả về trong dữ liệu phản hồi của API khi người dùng gọi đến.

4. Đối tượng và phạm vi nghiên cứu

- Đối tượng:

Đối với API, nghiên cứu framework Flask và triển khai xây dựng API hoàn chỉnh.

Đối với mô hình, nghiên cứu về kiến trúc của mô hình Transformer như tầng mã hóa (Encoder Layer), tầng giải mã (Decoder Layer), … Bên cạnh đó tìm hiểu về thư viện Tensorflow để xây dựng, triển khai mô hình, tạo mô hình mã hóa cho câu đầu vào

và giải mã cho câu đầu ra. Thư viện Phunspell để phát hiện từ sai dành cho trường hợp các từ đó không nằm trong tập dữ liệu huấn luyện, sử dụng khả năng gợi ý từ của thư viện để thay thế cho từ đó. Thư viện Numpy để thực hiện tính toán trên ma trận. Thư viện Pandas dùng để đọc, ghi các dữ liệu phục vụ cho quá trình huấn luyện và kiểm tra.

- Phạm vi nghiên cứu:

Đối với API, phạm vi nghiên cứu sẽ là các phương thức gửi yêu cầu lên máy chủ, cách thức truyền dữ liệu thông qua các phương thức, nhận dữ liệu, trả dữ liệu về cho người dùng.

Đối với mô hình, phạm vi nghiên cứu sẽ là dữ liệu đến từ các bài báo trên các website đọc báo trực tuyến trên mạng, cụ thể là https://vnexpress.net. Các lỗi từ sai thường mắc phải trong tiếng Việt như: từ không dấu, từ viết tắt, lỗi Unicode, …

5. Phương pháp nghiên cứu

Về lý thuyết, tìm hiểu về kiến trúc của mô hình Transformer, các tầng mã hóa và giải mã của mô hình Transformer, mô hình chuẩn hóa từ thành các vector.

Về thực hành, ưu tiên sử dụng các thư viện, framework của ngôn ngữ Python như: Flask để xây dựng API, thư viện Phunspell để phát hiện từ sai, thư viện Tensorflow để xây dựng mô hình và xử lý tập dữ liệu. Huấn luyện mô hình với đầu vào là câu đúng chính tả và câu sai chính tả.

6. Kết quả đạt được

Xây dựng được mô hình Transformer với các tầng cơ bản. Mô hình có khả năng phát hiện và sửa từ bị sai chính tả. Sử dụng thư viện Phunspell để tiếp tục xử lý các từ sai không nằm trong tập dữ liệu huấn luyện. Xây dựng API sửa lỗi chính tả và tích hợp vào trong website thi trắc nghiệm.

7. Bố cục luận văn

Phần giới thiệu:

- Giới thiệu tổng quát về đề tài.

Phần nội dung:

- Chương 1. Mô tả bài toán và cơ sở lý thuyết
- Chương 2. Thu thập và xử lý dữ liệu
- Chương 3. Thiết kế và cài đặt
- Chương 4. Kiểm thử và đánh giá

Phần kết luận:

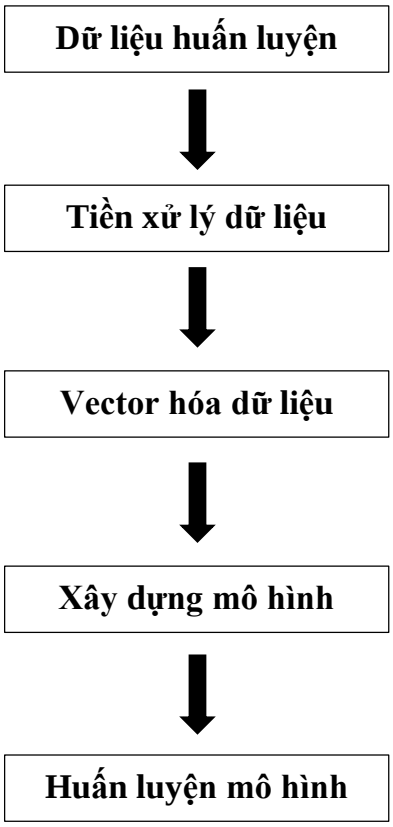
- Kết quả đạt được, hạn chế của mô hình và hướng phát triển mô hình trong tương lai.

PHẦN NỘI DUNG

CHƯƠNG 1. MÔ TẢ BÀI TOÁN

1. Mô tả chi tiết bài toán

Bài toán sửa lỗi chính tả trong văn bản tiếng Việt chính là áp dụng bài toán thêm dấu tiếng Việt sử dụng mô hình Transformer để tìm và sữa từ lỗi. Mô hình Transformer [4] sử dụng 2 tầng mã hóa và giải mã để đưa vào huấn luyện và sửa lỗi từ. Dữ liệu mã hóa là câu tiếng Việt sau khi đã gỡ bỏ hoàn toàn dấu câu và dữ liệu giải mã là câu tiếng Việt có dấu hoàn chỉnh. Dựa trên dữ liệu mã hóa và giải mã mô hình học được khả năng thêm dấu cho câu. Quá trình huấn luyện được biểu diễn như hình 1.



Hình 1. Quá trình huấn luyện

API được xây dựng bằng Flask sẽ đảm nhận việc nhận dữ liệu đầu vào, dữ liệu này sẽ được mã hóa trước khi đưa vào mô hình để sửa lỗi. Giải mã đầu ra của mô hình để có được câu hoàn chỉnh, khi câu được giải mã thành công Flask sẽ gửi lại phản hồi cho người dùng là câu đã được sửa hoàn chỉnh.

Bài toán sửa lỗi tiếng Việt có nét tương đồng giống với bài toán dịch máy trong máy học [5]. Khi mà mục tiêu của bài toán dịch máy là chuyển đổi ngôn ngữ A sang ngôn ngữ B thì đối với bài toán sửa lỗi tiếng Việt lại là chuyển đổi ngôn ngữ bị sai chính tả sang ngôn ngữ đúng chính tả.

Các lỗi chính tả trong tiếng Việt rất đa dạng như các từ sau: “chuong”, “chowf”, “ko”,… Đề tài chỉ tập trung vào xử lý lỗi chính tả thiếu dấu câu.

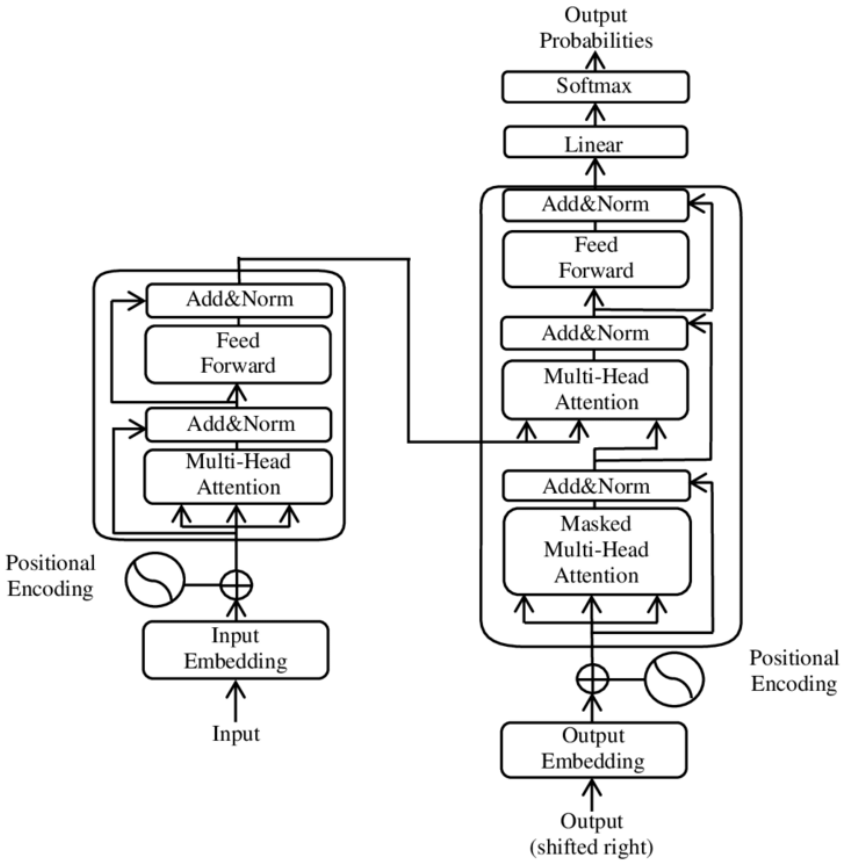
2. Cơ sở lý thuyết

2.1. Transformer

Transformer là một mô hình học sâu được giới thiệu năm 2017, được dùng chủ yếu ở lĩnh vực xử lý ngôn ngữ tự nhiên (NLP).

Giống như các mạng thần kinh hồi quy (recurrent neural network – RNN), các Transformer được thiết kế để xử lý dữ liệu tuần tự, chẳng hạn như ngôn ngữ tự nhiên, cho các tác vụ như dịch mấy thống kê hay tóm tắt tự động. Tuy nhiên, khác với RNN, các Transformer không yêu cầu dữ liệu tuần tự được xử lý theo thứ tự. Ví dụ, nếu dữ liệu đầu vào là một câu ngôn ngữ tự nhiên, Transformer không cần phải xử lý phần đầu câu trước phần cuối câu. Do tính năng này, Transformer cho phép nhiều phép tính toán song song và vì vậy giảm thời gian huấn luyện [6].

Bài toán sử dụng kiến trúc tổng quan của mô hình Transformer bao gồm có mã hóa (encoder) nằm bên trái và giải mã (decoder) nằm bên phải được mô tả ở hình 2.



Hình 2. Kiến trúc mô hình Transformer

2.2. Encoder

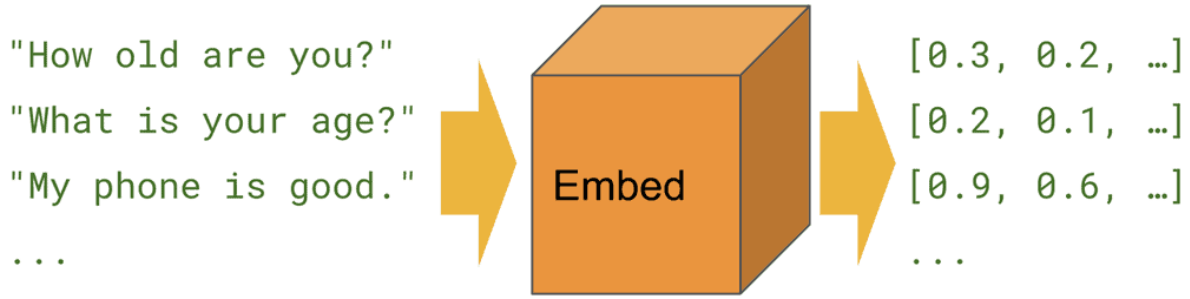
Một khối encoder (encoder block) được tạo thành từ nhiều nhiều tầng encoder (encoder layer) lại với nhau, mỗi một tầng encoder sẽ được tạo thành từ 2 thành phần: Multi-head Attention và Feedforward Network. Một khối encoder sẽ nhận đầu vào là input embeddings.

2.2.i. Input embedding

Input embedding thực chất là các vector embedding của các từ trong một câu đầu vào. Các embedding này được tạo ra bằng cách kết hợp vector word embedding và positional embedding. Trong đó:

- Vector word embedding là một vector dùng để biểu diễn dữ liệu là các từ dưới dạng các số thực.
- Vector positional embedding là một vector dùng để biểu diễn thứ tự của một từ trong câu, cho biết thông tin về vị trí và khoảng cách giữa các từ.

Việc vector hóa các một câu là việc cần thiết vì trong mô hình máy học, máy tính không có khả năng đọc được cái ký tự mà cần phải chuyển sang dạng số để máy tính có thể thực hiện việc tính toán.



Hình 3. Word embedding

2.2.ii. Multi-head attention

Multi-head attention là sự kết hợp của nhiều tầng self-attention lại với nhau, thông thường có 8 tầng. Nhiệm vụ của các tầng self-attention này là thực hiện phép tính nhân từng vector embedding với 3 ma trận trọng số là Wq, Wk, Wv, kết quả của phép tính là 3 vector q, k và v. Trong đó vector q (queries) và vector k (keys) dùng để tính trọng số khuếch đại thông tin của từng từ trong câu. Vector v (values) là vector biểu diễn cho các từ trong câu đầu vào. Các vector trong số này sẽ được cập nhật trong suốt quá trình huấn luyện.

Trọng số ma trận q, k và v là hoàn toàn khác nhau đối với mỗi tầng self-attention. Toàn bộ các tầng self-attention sẽ được tính toán song song với nhau, vector biểu diễn v của mỗi tầng sẽ được nối lại trước khi nhân với ma trận trọng số Wo để cho ra một vector duy nhất, vector này sẽ là đầu vào cho bước tiếp theo của khối encoder là Add & Normalize.

2.2.iii. Add & Normalize

Như trong mô hình cấu trúc của Transformer, ta có thể thấy được quá trình này được thực hiện sau quá trình Multi-head attention và Feed-forward. Nhận đầu vào chính là các vector đầu ra của một trong 2 bước trên, sau đó sử dụng kỹ thuật dropout (bỏ học) với tỉ lệ nhất định và đem cộng với vector đầu vào. Tiếp theo là Normalize vector đó

nhằm mục đích bổ sung thông tin cho vector tránh việc bị mất mát thông tin gốc quá nhiều khi thực hiện quá trình dropout trên.

2.2.iv. Feed-forward network

Còn được gọi là mạng thông tin truyền thẳng, thông tin di chuyển chỉ một chiều hướng đến từ các nút đầu vào, thông qua các nút ẩn (nếu có) và đi đến các nút đầu ra. Không có chu trình (chu kỳ) hoặc vòng lặp trong mạng [7][8].

Hàm Feed-forward network như sau:

$$\text{FNN}(x, W_1, W_2, b_1, b_2) = \max(0, xW_1 + b_1)\, W_2 + b_1$$

Trong đó W_1, W_2, b_1, b_1 là các tham số có thể học được.

Nhiệm vụ của quá trình này đơn giản chỉ là khai thác những thông tin ẩn mà ở quá trình Multi-head attention không khai thác được.

2.3. Decoder

Kiến trúc của khối decoder cũng gần như là tương tự so với khối encoder. Sự khác biệt là ở khối decoder sẽ có tầng Masked Multi-head attention thực hiện tính toán trước khi đi qua các tầng giống như của khối encoder. Trước khi tiến đến tầng Multi-head attention thì tầng decoder cần thêm cả đầu ra của khối encoder để làm đầu vào cho tầng này bên cạnh đầu ra của tầng Masked Multi-head attention. Đầu ra của tầng decoder sẽ được đưa vào hàm tuyến tính (Linear) để thực hiện việc biến đổi vector đầu vào này thành các vector có số chiều bằng với số từ có trong bộ từ điển của tập dữ liệu. Sau đó vector này được đưa vào hàm Softmax để biến thành phân phối xác xuất. Dựa trên phân phối xác xuất mà biết được từ nào được chọn.

Nhiệm vụ của khối decoder đơn giản chỉ là giải mã những gì mà khối encoder đã mã hóa. Quá trình decode của khối decoder gần như tương tự so với quá trình encoder, khi mà phần sau của khối decoder là giống với kiến trúc của khối encoder. Ở tầng Multi-head attention của khối decoder thì thay vì nhận đầu vào là vector input thì tầng này sẽ lấy ma trận trọng số Wq từ tầng Masked Multi-head attention trước đó kết hợp với 2 ma trận Wk và Wv từ đầu ra của tầng encoder làm đầu vào.

2.3.i. Masked Multi-head attention

Nhiệm vụ của tầng này gần như là tương tự so với tầng Multi-head Attention. Tuy nhiên, thay vì thực hiện giống như tầng Multi-head attention thực hiện việc tính toán trên toàn bộ vector thì Masked Multi-head attention sẽ thực hiện việc tính toán trên một phần của vector. Cụ thể là khi thực hiện tính toán đến từ thứ *i* trong câu thì phần còn lại phía sau sẽ được ẩn đi khi thực hiện tính toán.

2.4. Flask framework

Flask là một micro framework dành cho xây dựng ứng dụng web của ngôn ngữ Python và không cần một công cụ hay thư viện cụ thể. Flask hỗ trợ dễ dàng việc tích hợp các chức năng sẵn có mà bên thứ ba cung cấp như xử lý biểu mẫu, lớp trừu tượng cho cơ sở dữ liệu,…[9].

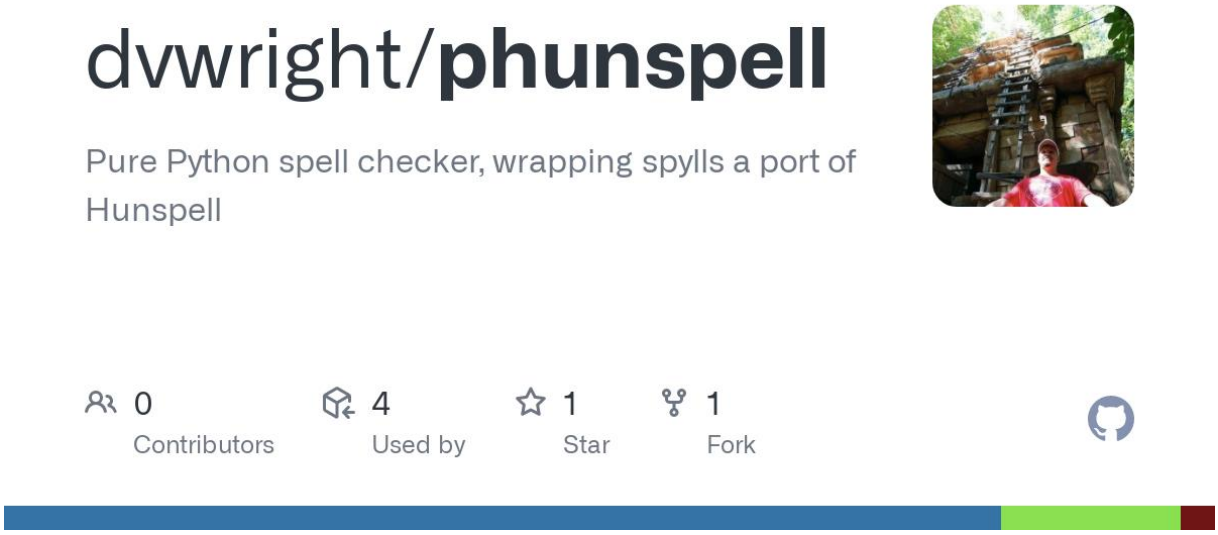
Hình 4. Logo Flask framework

2.5. Thư viện Phunspell

Thư viện Phunspell là một thư viện giúp kiểm tra chính tả dành cho ngôn ngữ Python. Phunspell sử dụng spylls, một cổng của bộ công cụ kiểm tra chính tả Hunspell.

Phunspell cung cấp bộ từ điển của tất cả các ngôn ngữ mà được phần mềm LibreOffice hỗ trợ. Hiện tại đã hỗ trợ lên tới hơn 70 ngôn ngữ, bao gồm cả tiếng Việt.

Phunspell cung cấp khả năng phát hiện một từ là đúng hay sai chính tả, hoặc là phát hiện nhiều từ trong một câu. Bên cạnh việc phát hiện thư viện cũng có khả năng đưa ra danh sách các từ có thể thay thế dành cho từ sai đó.



Hình 5. Thư viện Phunspell

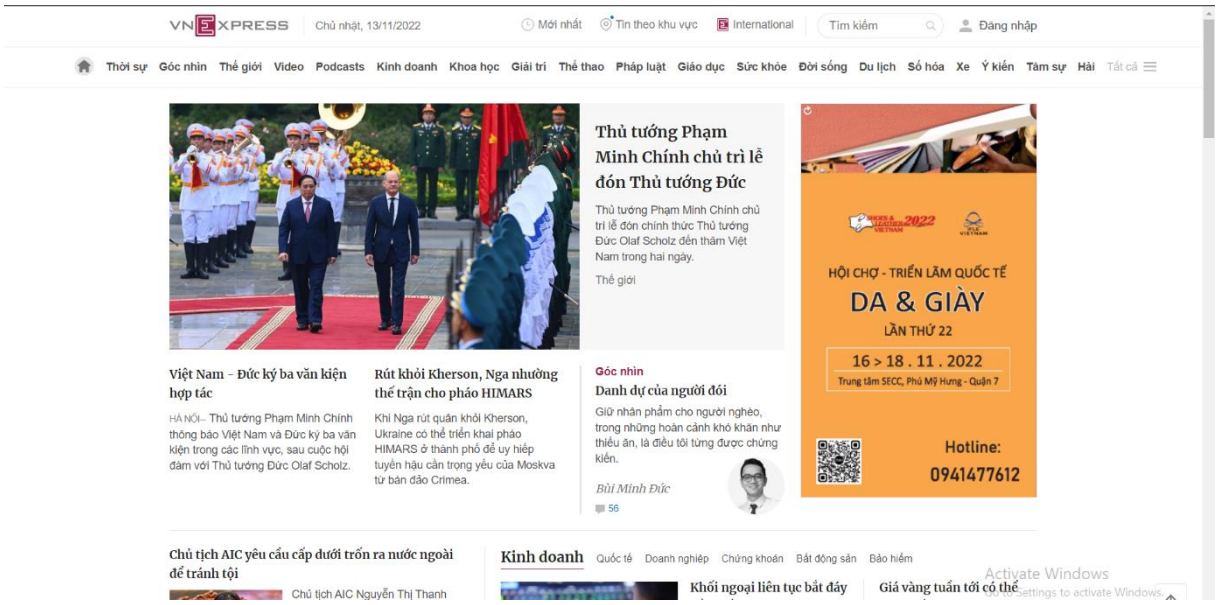
2.6. Thư viện Tensorflow

Là một thư viện mã nguồn mở dành cho máy học được phát triển bởi Google Brain, hỗ trợ các tính toán thường được sử dụng trong máy học và học sâu giúp việc giải quyết các vấn đề liên quan dễ dàng hơn. Tensorflow cung cấp khả năng chạy song song trên nhiều CPU hay GPU cùng một lúc bên cạnh việc cung cấp các hàm tính toán sẵn có phục vụ cho máy học và học sâu giúp cho việc giải quyết các bài toán máy học nhanh chóng và dễ dàng hơn.

CHƯƠNG 2. THU THẬP VÀ XỬ LÝ DỮ LIỆU

1. Thu thập dữ liệu

Dữ liệu được thu thập sẽ là nội dung của các bài báo trên trang web vnexpress.net với hơn 10 chủ đề khác nhau như: thời sự, góc nhìn, thế giới, khoa học, giải trí, thể thao, … đem lại sự đa dạng cho tập dữ liệu.



Hình 6. Giao diện trang báo vnpexress.net

Tập dữ liệu được thu thập từ trang báo điện tử vnpexress.net đảm bảo vệ sự chính xác về chính tả cho nội dung của các bài báo. Đây là một nguồn dữ liệu đáng tin cậy cho việc huấn luyện mô hình.

Sử dụng thư viện BeautifulSoup4 của ngôn ngữ python để tiến hành thu thập dữ liệu, thư viện này giúp chuyển đổi dữ liệu HTML giúp dễ dàng trích xuất được nội dung của các bài báo.

Tập dữ liệu thu thập được chứa hơn 160.000 câu tiếng Việt, độ dài trung bình của các câu trong tập dữ liệu là 54 từ.

2. Tiền xử lý dữ liệu

2.1. Tiền xử lý dữ liệu

Dữ liệu sẽ được xử lý qua các bước như

- Xóa các kí tự đặt biệt

- Xóa các mã HTML
- Xóa các kí tự xuống dòng (\n)
- Lược bỏ câu không chứa bất kỳ từ tiếng Việt nào

2.2. Xây dựng tập dữ liệu lỗi

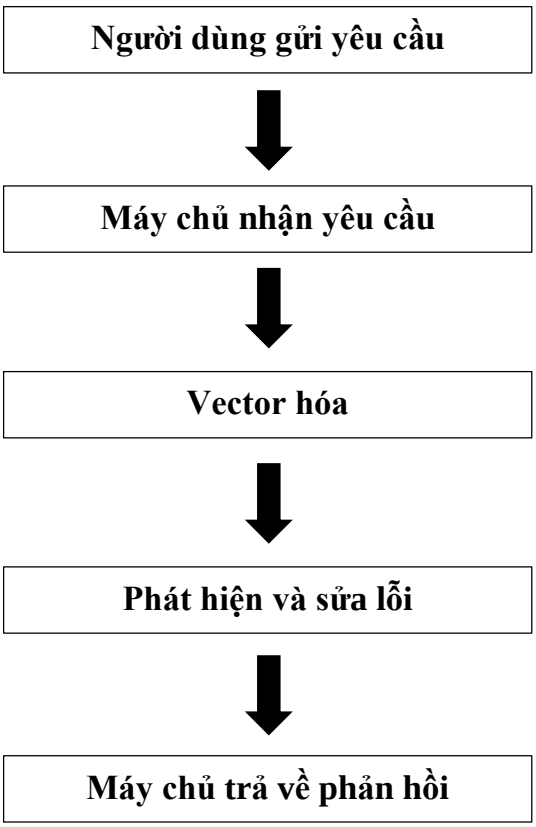
Để có thể huấn luyện mô hình, ta cần xây dựng tập dữ liệu lỗi để mô hình có thể học được các sửa lỗi dựa trên tập dữ liệu đúng và tập dữ liệu lỗi.

Vì đây là bài toán thêm dấu câu tiếng Việt dùng mô hình Transformer, dữ liệu lỗi sẽ là các câu không chứa dấu câu, ví dụ: day la mot cau tieng viet.

CHƯƠNG 3. THIẾT KẾ VÀ CÀI ĐẶT

1. Quy trình sửa lỗi chính tả

Quy trình sửa lỗi chính tả bắt đầu khi người dùng gửi yêu cầu lên máy chủ với câu mà người dùng cần sửa lỗi. Máy chủ nhận được yêu cầu của người dùng, bắt đầu vector hóa câu của người dùng gửi lên, vector thu được đưa vào mô hình để thực hiện việc phát hiện và sửa lỗi. Sau khi hoàn thành việc sửa lỗi, máy chủ trả về phản hồi cho người dùng là câu sau khi đã sửa lỗi thành công. Quy trình được mô tả như hình 7.



Hình 7. Quy trình sửa lỗi chính tả

2. Mô hình Transformer

Mô hình Transformer được tạo thành từ 2 khối là encoder và decoder, vì vậy cần phải tiến hành xây dựng 2 khối này trước. Ở trang chủ của thư viện Tensorflow cung cấp sẵn cách triển khai mô hình Transformer được áp dụng trong đề tài này [10]

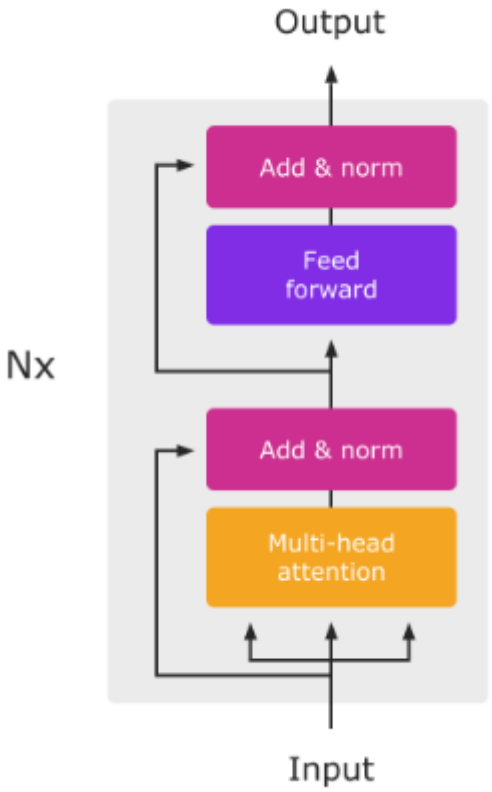
Ngoài ra, một vài tham số ta cần nắm trước khi thực hiện xây dựng mô hình như:

- Num_layers: số lượng khối Attention sub-layers của tầng encoder và decoder.

- D_model: kích thước của vector embedding.
- Num_heads: số lượng head có trong một attention layers.
- Dff: số lượng tối đa của các token được cho phép của văn bản đầu vào.
- Dropout_rate: tỷ lệ dropout.
- Input_vocab_size: số lượng từ vựng trong tập dữ liệu đúng.
- Output_vocab_size: số lượng từ vựng trong tập dữ liệu lỗi.

2.1. Encoder

Encoder là khối giúp ta biến vector đầu vào đã được trích xuất dữ liệu thành 3 vector trọng số Wq, Wk, Wv. Quá trình thực hiện tính toán 3 vector trọng số là nhiệm vụ của tầng encoder.



Hình 8. Kiến trúc Encoder

2.1.i. Multi-head Attention

Xây dựng lớp Multi-head attention kế thừa hàm khởi tạo của lớp Multi-head attention của thư viện Tensorflow. Các biến khởi tạo của lớp này bao gồm có

num_heads, d_model, các hàm tính toán trọng số Wq, Wk, Wv được tạo bởi phương thức keras.layers.Dense của Tensorflow. Khi gọi đến lớp này sẽ thực hiện tính toán trọng số cho 3 ma trận q, k và v.

2.1.ii. Add & Normalize

Tầng này sẽ được gọi để tính toán sau khi dữ liệu đã trải qua tầng Multi-head attention và Feed-forward.

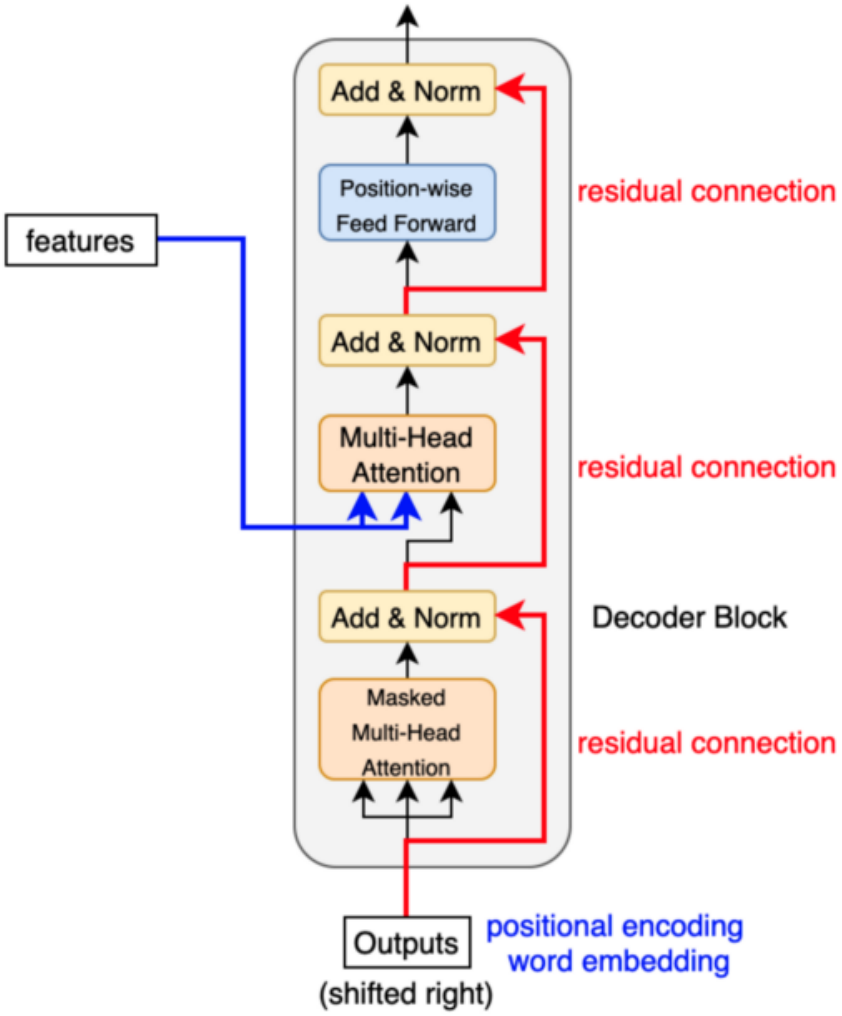
Xây dựng layernorm bằng phương thức keras.layers.LayerNormalization của Tensorflow, dropout bằng phương thức keras.layers.Dropout với tỉ lệ dropout mong muốn. Cần phải có 2 cặp layernorm và dropout riêng biệt để tính toán đầu ra của 2 tầng Multi-head attention và Feed-forward.

2.1.iii. Feed-forward

Xây dựng hàm Feed-forward bằng phương thức keras.Sequential với một mảng là 2 tầng tính toán keras.layers.Dense, trong đó tầng đầu tiên nhận tham số dff sử dụng hàm kích hoạt ReLU, tầng thứ 2 sử dụng tham số d_model với hàm kích hoạt mặc định.

2.2. Decoder

Quá trình này gần giống với encoder tuy nhiên có khác biệt đôi chút khi có thêm 1 tầng nhiều hơn so với encoder và vector đầu vào là vector của các câu chuẩn đã được trích xuất đặc trưng thay vì là các vector lỗi đã được trích xuất đặc trưng.



Hình 9. Kiến trúc Decoder

2.2.i. Masked Mutli-head attention

Tầng này cũng giống như tầng Multi-head attention vì cả 2 tầng đều thực hiện tính toán tương tự nhau vì vậy việc xây dựng cũng sẽ tương tự nhau. Sự khác biệt là khi Masked thực hiện tính toán sẽ che đi phần phía sau của từ đang được tính toán.

Đầu ra của quá trình Decoder sẽ cần phải qua tầng Linear và Softmax. Xây dựng 2 tầng này bằng phương thức keras.layers.Dense với hàm kích hoạt tương ứng cho mỗi tầng.

3. Mô hình sửa lỗi tiếng Việt

3.1. Xây dựng bộ tokenizer.

Giống như hầu hết các bài toán xử lý ngôn ngữ tự nhiên khác khi mà việc trích xuất đặc trưng để huấn luyện hay còn gọi là vector hóa văn bản là việc cần phải có. Lý

do để cần phải tiến hành trích xuất đặc trưng là vì các mô hình máy học thực chất cũng chỉ là các phép toán, mà những phép toán thì không thể thực hiện việc tính toán như lấy 2 kí tự nhân với nhau. Ví dụ ta lấy từ "tôi" và thực hiện phép nhân với từ "là" thì điều này là không thực hiện được. Đó là lý do vì sao ta cần thực hiện quá trình trích xuất đặc trưng, khi mà quá trình này giúp ta biến đổi các câu văn bản thuần túy thành các vector dưới dạng số để các phép toán có thể thực hiện được. Như ví dụ trên khi ta thực hiện vector hóa thì từ "tôi" sẽ được trích xuất thành số 1 và từ "là" trích xuất thành số 2, khi đó ta đã có thể thực hiện phép toán với số 1 và 2.

Để có thể thực hiện việc vector hóa văn bản này thì ta cần phải xây dựng bộ tokenizer dành cho tập dữ liệu dùng để huấn luyện. Tensorflow cung cấp một module giúp chúng ta xây dựng một cách dễ dàng. Cài đặt module này với câu lệnh: pip install tensorflow-datasets.

Ta sử dụng phương thức build_from_corpus nằm trong lớp SubwordTextEncoder của module tensorflow-datasets để thực hiện xây dựng bộ tokenizer cho cả 2 tập dữ liệu chuẩn và lỗi.

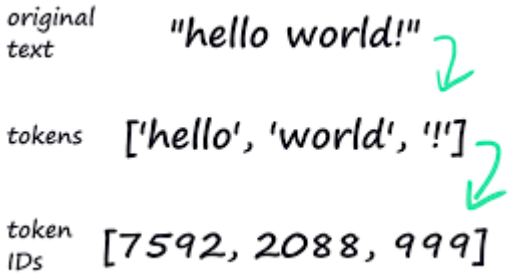
Bộ tokenizer sẽ có 2 phương thức là encode và decode, đối với encode sẽ giúp biến đổi các câu thành các vector.

"tôi thích máy học" => [1, 10, 32, 221]

Và ngược lại phương thức decode sẽ giúp biến đổi vector thành câu hoàn chỉnh.

[1, 10, 32, 221] => "tôi thích máy học".

Để có thể huấn luyện mô hình, ta sẽ cần thêm ở đầu và cuối mỗi vector các padding nhằm giúp đánh dấu điểm bắt đầu và kết thúc của vector đó.



Hình 10. Quá trình tokenizer

3.2. Huấn luyện mô hình

Quá trình huấn luyện mô hình thực chất chỉ là việc đi tìm các trọng số làm sao cho các trọng số này khi thực hiện các tính toán với vector đầu vào cho kết quả giống với vector đầu ra. Trong đó các vector đầu ra là các câu đúng chính tả từ tập dữ liệu chuẩn, còn vector đầu vào là các câu trong tập dữ liệu lỗi được xây dựng ở bước tiền xử lý dữ liệu.

Khởi tạo mô hình với các tham số được giới thiệu ở phần Mô hình Transformer ở trên. Sử dụng hàm tối ưu Adam để tối ưu hóa mô hình, Sparse Categorical Crossentropy để thực hiện tính toán lỗi và độ chính xác cho mô hình.

Khi huấn luyện mô hình thành công, sử dụng phương thức save_weights để lưu lại các trọng số đã huấn luyện được vào tập tin "model_weight.h5".

Quá trình này huấn luyện sẽ thực hiện tính toán với các vector thu được sau khi trích xuất đặc trưng. Tập các vector này sẽ được chia nhỏ thành các batch, mô hình sẽ thực hiện tính toán trọng số với từng batch một cho đến hết. Khi toàn bộ các batch đã được tính toán xong thì quá trình huấn luyện đã kết thúc và điều này tương đương với 1 epoch đã được hoàn thành. Mô hình có thể lại tiếp tục thực hiện một epoch mới cho đến khi thu được mô hình tốt nhất. Số lượng epoch và số lượng của một batch sẽ phụ thuộc vào việc cài đặt tham số cho mô hình.

3.3. Sửa lỗi

Để có thể sửa lỗi, ta cần phải tải các trọng số đã huấn luyện được từ tập tin "model_weight.h5" bằng phương thức load_weights.

Quá trình sửa lỗi sẽ diễn ra như sau:

- Vector hóa câu cần sửa lỗi
- Khởi tạo vector output bằng padding bắt đầu câu
- Cho 2 vector vào mô hình

- Mô hình tính toán tìm từ tiếp theo, ta nối từ đó vào cuối câu output rồi tiếp tục với output mới.

Quá trình này sẽ diễn ra cho đến khi đạt đến độ dài của câu cần sửa lỗi hoặc khi tìm thấy từ tiếp theo là padding kết thúc câu.

3.4. Giải mã

Sau khi trải qua quá trình sửa lỗi sẽ nhận được một vector, vector này sẽ cần được mã hóa để có thể nhận lại được một câu hoàn chỉnh trước khi được trả về cho người dùng.

Đây chính là quá trình ngược lại của bước tokenizer khi đây là bước trích xuất đặc trưng từ câu thành vector để có thể tính toán trong mô hình máy học, thì quá trình giải mã sẽ là quá trình biến vector số học thành một chuỗi các kí tự ghép thành một câu hoàn chỉnh để người dùng có thể đọc được thay vì việc trả về một vector chỉ là những con số mà người dùng không biết nó mang ý nghĩa gì.

Sử dụng bộ tokenizer của tập dữ liệu chuẩn với phương thức decode nhận vào là vector và sẽ trả về là một câu tương ứng cho vector đó.

CHƯƠNG 4. KIỂM THỬ VÀ ĐÁNH GIÁ

1. Môi trường kiểm thử

Môi trường kiểm thử sẽ được chạy trên phiên bản python 3.7.12 (64 bit) với các gói và thư viện cùng phiên bản của nó được liệt kê dưới bảng sau:

Bảng 1. Thư viện để cài đặt môi trường

Tên	Phiên bản	Câu lệnh cài đặt
Tensorflow	2.6.4	pip install tensorflow==2.6.4
Tensorflow_datasets	4.3.0	pip install tensorflow-datasets==4.3.0
Flask	Lastest	pip install Flask
Pandas	1.3.5	pip install pandas==1.3.5
Numpy	1.21.6	pip install numpy==1.21.6

Máy chủ cung cấp API chạy bằng Flask sẽ được chạy ở máy cục bộ ở cổng 8000. Với địa chỉ https://localhost:8000. Trong đó API để gọi khi muốn sửa lỗi là /spellcheck sử dụng phương thức POST, nhận dữ liệu với content-type là application/json.

2. Kiểm thử

Quá trình kiểm thử sẽ bao gồm việc kiểm tra độ chính xác của mô hình và so sánh độ hiệu quả khi sử dụng mô hình so với sử dụng khả năng gợi ý từ của thư viện Phunspell.

Kiểm thử mô hình với các tham số như sau:

Bảng 2. Tham số dùng để huấn luyện mô hình 1

Num_layers	4
D_model	128
Dff	512
Num_heads	8
Input_vocab_size	8358
Output_vocab_size	8180

Dropout_rate	0.1
Batch_size	24

Bảng 3. Tham số đung để huấn luyện mô hình 2

Num_layers	4
D_model	128
Dff	512
Num_heads	8
Input_vocab_size	8358
Output_vocab_size	8180
Dropout_rate	0.1
Batch_size	24

Bảng 4. Tham số dùng để huấn luyện mô hình 3

Num_layers	4
D_model	128
Dff	512
Num_heads	8
Input_vocab_size	8358
Output_vocab_size	8180
Dropout_rate	0.1
Batch_size	24

Sử dụng thư viện Phunspell, lấy từ đầu tiên được gợi ý để thay thế cho từ bị sai.

3. Đánh giá

3.1. Công thức đánh giá

Công thức dùng để đánh giá mô hình như sau:

$$Accuracy = \frac{\sum_{0}^{n-1} \frac{tt_i}{t_i}}{n}$$

Trong đó:

- N là tổng số câu có trong tập dữ liệu.
- tt_i là số từ được sửa đúng trong câu thứ i.
- t_i là tổng số từ có trong câu thứ i.

3.2. Kết quả thu được

Độ chính xác thu được khi sử dụng các mô hình và thư viện Phunspell được biểu diễn ở bảng sau:

Mô hình	**Độ chính xác**
Mô hình tham số 1	92,8 %
Mô hình tham số 2	89,4 %
Mô hình tham số 3	80,6 %
Phunspell	

PHẦN KẾT LUẬN

1. Kết quả đạt được

"Ứng dụng mô hình Transformer vào sửa lỗi chính tả tiếng Việt" đã được xây dựng thành công và đạt được các kết quả như sau:

- Xây dựng thành công mô hình Transformer, mô hình sửa được những từ sai.
- Xây dựng thành công API nhận và trả dữ liệu hoàn chỉnh.
- Mô hình có độ chính xác cao lên đến 92,8%.
- Mô hình dễ sử dụng với sự hỗ trợ hoàn toàn đến từ thư viện Tensorflow.

2. Ưu và nhược điểm

Ưu điểm:

- API dễ dàng đơn giản, dễ sử dụng khi sử dụng Flask – một framework dễ triển khai cho các API đơn giản.
- Trọng số của mô hình nhỏ gọn, dễ dàng tải lại mô hình.
- Mô hình có độ chính xác cao lên đến 92,8%
- Xây dựng API giúp người dùng dễ dàng tiếp cận đến mô hình hoàn chỉnh.

Nhược điểm:

- Thời gian dự đoán một câu tương đối lâu, trung bình mất đến 3 giây để dự đoán xong một câu.
- Mô hình chỉ có thể dự đoán cho các câu có độ dài tối đa là 38 từ, nếu muốn mô hình có thể dự đoán các câu dài hơn cần phải huấn luyện lại mô hình, điều này đồng nghĩa với thời gian sửa lỗi có thể sẽ tăng lên.
- Mô hình chỉ có thể sửa lỗi chính tả ở dạng các từ không có dấu, nếu gặp các lỗi chính tả khác sẽ không sửa lỗi được.

3. Hướng phát triển

- Giúp mô hình có khả năng sửa mọi lỗi chính tả thường gặp hơn thay vì chỉ sửa lỗi một dạng duy nhất bằng cách giả lập thêm các dạng lỗi ở tập dữ liệu lỗi hoặc kết hợp với thư viện Phunspell.
- Tăng tốc khả năng sửa lỗi giúp mô hình dự đoán nhanh hơn.

TÀI LIỆU THAM KHẢO

[1] Do, Dinh-Truong, et al. "VSEC: Transformer-Based Model for Vietnamese Spelling Correction." Pacific Rim International Conference on Artificial Intelligence. Springer, Cham, 2021.

[2] Nguyen, Ha Thanh, Tran Binh Dang, and Le Minh Nguyen. "Deep learning approach for vietnamese consonant misspell correction." International Conference of the Pacific Association for Computational Linguistics. Springer, Singapore, 2019.

[3] Thi Xuan Huong, Nguyen, Tran-Thai Dang, and Anh-Cuong Le. "Using large N-gram for vietnamese spell checking." Knowledge and Systems Engineering. Springer, Cham, 2015. 617-627.

[4] Feng, Ben, Dayiheng Liu, and Yanan Sun. "Evolving transformer architecture for neural machine translation." Proceedings of the Genetic and Evolutionary Computation Conference Companion. 2021.

[5] Somers, Harold. "Machine translation: History, development, and limitations." (2011).

[6] Vaswani, Ashish, et al. "Attention is all you need." Advances in neural information processing systems 30 (2017).

[7] Zell, Andreas. Simulation neuronaler netze. Vol. 1. No. 5.3. Bonn: Addison-Wesley, 1994.

[8] Schmidhuber, Jürgen. "Deep learning in neural networks: An overview." Neural networks 61 (2015): 85-117.

[9] Grinberg, Miguel. Flask web development: developing web applications with python. " O'Reilly Media, Inc.", 2018.

[10] "*Neural machine translation with a Transformer and Keras | Text | TensorFlow*". Neural machine translation with a Transformer and Keras. 1 Nov. 2022, https://www.tensorflow.org/text/tutorials/transformer.