**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

[**PHẦN GIỚI THIỆU** 1](#_Toc119620054)

[1. Đặt vấn đề 1](#_Toc119620055)

[2. Lịch sử giải quyết vấn đề 1](#_Toc119620056)

[3. Mục tiêu đề tài 2](#_Toc119620057)

[4. Đối tượng và phạm vi nghiên cứu 2](#_Toc119620058)

[5. Phương pháp nghiên cứu 3](#_Toc119620059)

[6. Kết quả đạt được 3](#_Toc119620060)

[7. Bố cục luận văn 4](#_Toc119620061)

[**PHẦN NỘI DUNG** 5](#_Toc119620062)

[CHƯƠNG 1. MÔ TẢ BÀI TOÁN 5](#_Toc119620063)

[1. Mô tả chi tiết bài toán 5](#_Toc119620064)

[2. Cơ sở lý thuyết 6](#_Toc119620065)

[2.1. Transformer 6](#_Toc119620066)

[2.2. Encoder 7](#_Toc119620067)

[2.2.i. Input embeddings 7](#_Toc119620068)

[2.2.ii. Multi-head Attention 8](#_Toc119620069)

[2.2.iii. Add & Normalize 8](#_Toc119620070)

[2.2.iv. Feed-forward network 9](#_Toc119620071)

[2.3. Decoder 9](#_Toc119620072)

[2.3.i. Masked Multi-head Attention 10](#_Toc119620073)

[2.4. Flask framework 10](#_Toc119620074)

[2.5. Thư viện Phunspell 10](#_Toc119620075)

[2.6. Thư viện Tensorflow 11](#_Toc119620076)

[CHƯƠNG 2. THU THẬP VÀ XỬ LÝ DỮ LIỆU 12](#_Toc119620077)

[1. Thu thập dữ liệu 12](#_Toc119620078)

[2. Tiền xử lý dữ liệu 12](#_Toc119620079)

[2.1. Tiền xử lý dữ liệu 12](#_Toc119620080)

[2.2. Xây dựng tập dữ liệu lỗi 13](#_Toc119620081)

[CHƯƠNG 3. THIẾT KẾ VÀ CÀI ĐẶT 14](#_Toc119620082)

[1. Quy trình sửa lỗi chính tả 14](#_Toc119620083)

[2. Mô hình Transformer 14](#_Toc119620084)

[3. Mô hình sửa lỗi tiếng Việt 15](#_Toc119620085)

[3.1. Xây dựng bộ tokenizer. 15](#_Toc119620086)

[3.2. Huấn luyện mô hình 16](#_Toc119620087)

[3.3. Sửa lỗi 16](#_Toc119620088)

danh mục hình

bảng danh sách từ viết tắt

abtract

tóm tắt

**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.

* 1. 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(thêm vào bảng từ viết tắt) 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 [4] – một thư viện phát hiện các từ sai và gợi ý các từ thay thế của ngôn ngữ Python. Hiện tại Phunspell [4] hỗ trợ hơn 70 ngôn ngữ khác nhau trong đó có tiếng Việt. Phunspell được phát triển dựa trên thư viện Hunspell [5] - 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.
  1. 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.

* 1. 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 [6] 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, …

* 1. 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.

* 1. 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.

* 1. 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**

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 [note]để tìm và sữa từ lỗi. Sử dụng thêm thư viện Phunspell [note] để giải quyết các từ sai không xuất hiện trong tập dữ liệu huấn luyện. Mô hình Transformer 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 sau (hình ??):

**Dữ liệu huấn luyện**

**Tiền xử lý dữ liệu**

**Vector hóa dữ liệu**

**Xây dựng mô hình**

**Huấn luyện mô hình**

Figure 1.Quá trình huấn luyện

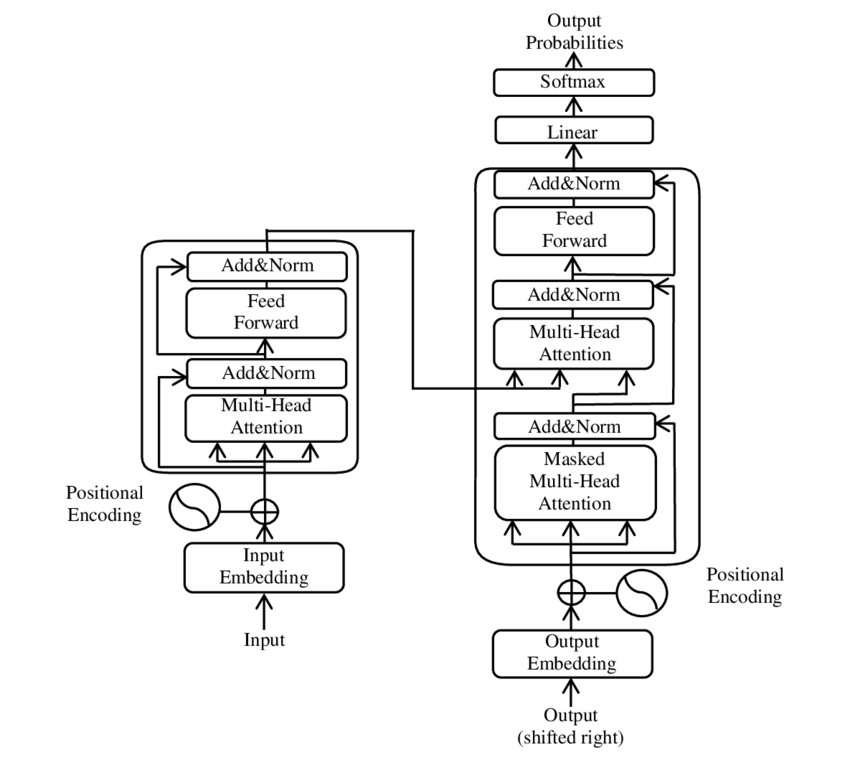
API được xây dựng bằng Flask sẽ đảm nhận việc nhận dữ liệu đầu vào, dữ liệu này sẽ được mã hóa trước khi đưa vào mô hình để sửa lỗi. Giải mã đầu ra của mô hình để có được câu hoàn chỉnh, sau đó sử dụng thư viện Phunspell để kiểm tra các từ mà mô hình không xử lý được và thay thế các từ đó dưới dạng các từ đã mã hóa. Tiếp tục đưa vào mô hình để xử lý các từ vừa được mã hóa và cuối cùng nhận được câu hoàn chỉnh sau khi giải mã.

* 1. Cơ sở lý thuyết
     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) [wiki [Transformer (mô hình học máy) – Wikipedia tiếng Việt](https://vi.wikipedia.org/wiki/Transformer_(m%C3%B4_h%C3%ACnh_h%E1%BB%8Dc_m%C3%A1y))].

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 [wiki [Transformer (mô hình học máy) – Wikipedia tiếng Việt](https://vi.wikipedia.org/wiki/Transformer_(m%C3%B4_h%C3%ACnh_h%E1%BB%8Dc_m%C3%A1y))].

Bài toán sử dụng kiến trúc tổng quan của mô hình Transformer bao gồm có mã hóa (encoder) và giải mã (decoder) (hình ??)



* + 1. 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.

* + - 1. Input embeddings

Input embeddings thực chất là các vector embeddings của các từ trong một câu đầu vào. Các embeddings 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.

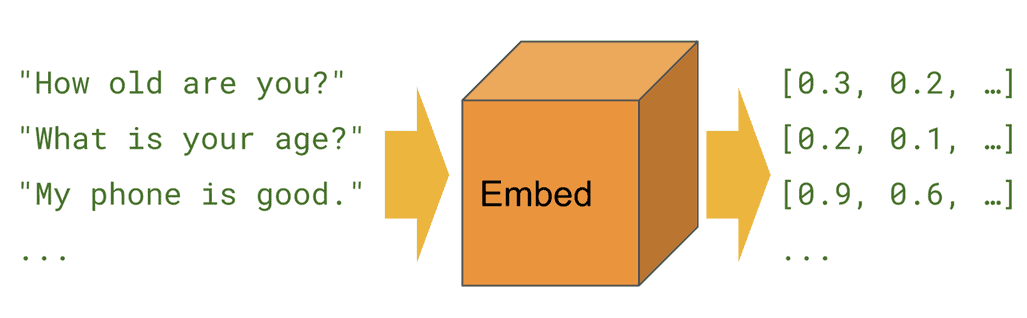
Việc vector hóa các một câu là việc cần thiết vì trong mô hình máy học, máy tính không có khả năng đọc được cái ký tự mà cần phải chuyển sang dạng số để máy tính có thể thực hiện việc tính toán.

Figure 2. Word embedding

* + - 1. 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 embeddings 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.

* + - 1. 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 Feedforward. 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.

* + - 1. 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 [[Mạng thần kinh truyền thẳng – Wikipedia tiếng Việt](https://vi.wikipedia.org/wiki/M%E1%BA%A1ng_th%E1%BA%A7n_kinh_truy%E1%BB%81n_th%E1%BA%B3ng)].

Hàm Feed-forward network như sau:

FNN() = max()

Trong đó , , , 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.

* + 1. 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 trong 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.

* + - 1. 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.

* + 1. Flask framework

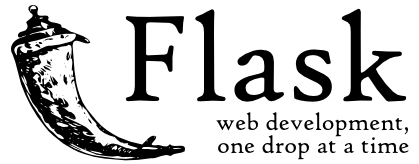
Flask là một micro framework dành cho xây dựng ứng dụng web của ngôn ngữ Python và không cần một công cụ hay thư viện cụ thể. Flask hỗ trợ dễ dàng việc tích hợp các chức năng sẵn có mà bên thứ ba cung cấp như xử lý biểu mẫu, lớp trừu tượng cho cơ sở dữ liệu,…[ [Flask (web framework) – Wikipedia tiếng Việt](https://en.wikipedia.org/wiki/Flask_(web_framework))].

Figure 3. Logo của thư viện Flask

* + 1. 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.

* + 1. Thư viện Tensorflow

Figure 4. Thư viện Phunspell trên github

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.

1. THU THẬP VÀ XỬ LÝ DỮ LIỆU
   1. Thu thập dữ liệu

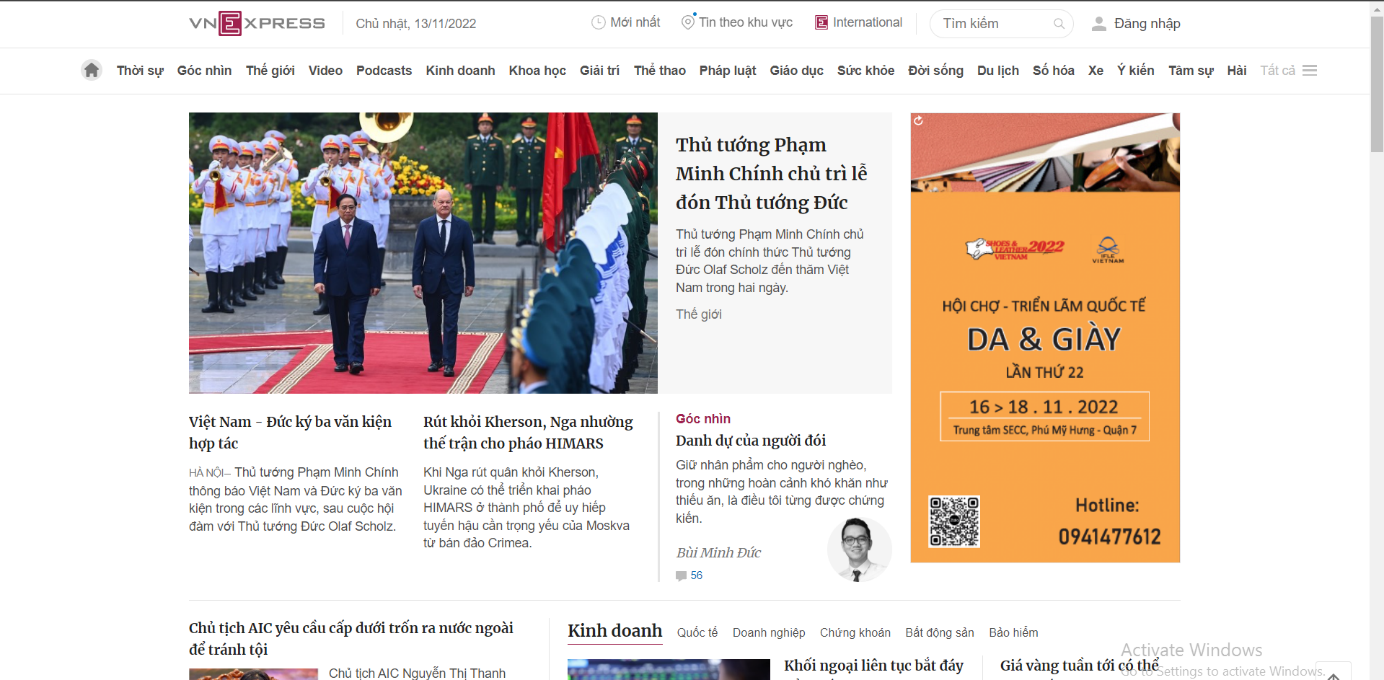
Dữ liệu được thu thập sẽ là nội dung của các bài báo trên trang web vnexpress.net với hơn 10 chủ đề khác nhau như: thời sự, góc nhìn, thế giới, khoa học, giải trí, thể thao, … đem lại sự đa dạng cho tập dữ liệu.

Figure 5. 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.

* 1. Tiền xử lý dữ liệu
     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
  + 1. 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.

1. 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, mấy chủ trả về phản hồi cho người dùng là câu sau khi đã sửa lỗi thành công. Quy trình được mô tả như hình (??)

Figure 6. Quy trình sửa lỗi chính tả

**Người dùng gửi yêu cầu**

**Máy chủ nhận yêu cầu**

**Vector hóa**

**Phát hiện và sửa lỗi**

**Máy chủ trả về phản hồi**

* 1. 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 Tranformers được áp dụng trong đề tài này [[Mô hình biến áp để hiểu ngôn ngữ  |  Text  |  TensorFlow](https://www.tensorflow.org/text/tutorials/transformer#multi-head_attention)]

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.
  1. Mô hình sửa lỗi tiếng Việt
     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 nhầm giúp đánh dấu điểm bắt đầu và kết thúc của vector đó.

* + 1. 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[Note] để tối ưu hóa mô hình, Sparse Categorical Crossentropy [Note] 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”

* + 1. 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.

* + 1. Sửa lỗi từ nằm ngoài từ điển

Do kích thước của tập dữ liệu là không quá lớn nên đôi khi sẽ xuất hiện các từ không nằm trong tập từ điển, vì vậy cần phải xử lý các từ này. Sử dụng thư viện chức năng gợi ý từ khi phát hiện từ sai của thư viện Phunspell để tiến hành xử lý trường hợp này.

Ví như từ “ching” không có trong tập từ điển thì lúc này thư viện Phunspell sẽ phát hiện đây là một từ lỗi là bắt đầu đưa ra danh sách gợi ý cho từ này như: “chồng”, “chông”, “chổng”, “chưng”, “trong”,… Tuy nhiên, ta không thể biết được từ chính xác dùng để thay thế cho từ sai kia là gì. Vì vậy cần chuyển những từ được gợi ý này về dạng những từ không có dấu. Khi đó ta sẽ còn 3 trường hợp để thay thế là “chong”, “chung”, “trong”. Dựa vào số lượng xuất hiện của 3 từ sau khi chuyển dạng thì từ “chong” có số lần xuất hiện nhiều nhất, sẽ được dùng để thay thế vào câu.

Sau đó tiếp tục cho qua mô hình để sửa lỗi lần nữa nhưng lần này sẽ chỉ sửa những từ được thêm vào ở trên thay vì chạy hết toàn bộ câu để sửa lỗi.

Trường hợp những từ mà Phunspell không xử lý được như “jdahswq” thì đây là những từ sai vô nghĩa nên sẽ được giữ lại và trả về cho người dùng.

1. KIỂM THỬ VÀ ĐÁNH GIÁ