

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Nâng cao chất lượng báo cáo khoa học dựa trên sửa
lỗi chính tả

ĐỖ THỊ HỒNG THẢO

thao.dth176878@sis.hust.edu.vn

Ngành: Công nghệ thông tin

Giảng viên hướng dẫn: PGS.TS Lê Thanh Hương

Chữ ký GVHD

Khoa: Khoa học máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 08/2022

LỜI CAM KẾT

Họ và tên sinh viên: Đỗ Thị Hồng Thảo
Điện thoại liên lạc: 0944588230
Email: thao.dth176878@sis.hust.edu.vn
Lớp: AS
Hệ đào tạo: Công nghệ thông tin Việt Nhật

Tôi – *Đỗ Thị Hồng Thảo* – cam kết Đồ án Tốt nghiệp (ĐATN) là công trình nghiên cứu của bản thân tôi dưới sự hướng dẫn của *PGS.TS Lê Thanh Hương*. Các kết quả nêu trong ĐATN là trung thực, là thành quả của riêng tôi, không sao chép theo bất kỳ công trình nào khác. Tất cả những tham khảo trong ĐATN – bao gồm hình ảnh, bảng biểu, số liệu, và các câu trích dẫn – đều được ghi rõ ràng và đầy đủ nguồn gốc trong danh mục tài liệu tham khảo. Tôi xin hoàn toàn chịu trách nhiệm với dù chỉ một sao chép vi phạm quy chế của nhà trường.

Hà Nội, ngày 8 tháng 8 năm 2022

Tác giả ĐATN

Thảo

Đỗ Thị Hồng Thảo

LỜI CẢM ƠN

Đồ án này được hoàn thành tại trường Đại học Bách Khoa Hà Nội dưới sự hướng dẫn của PGS.TS Lê Thanh Hương.

Lời đầu tiên, em xin chân thành cảm ơn PGS.TS Lê Thanh Hương, người đã luôn đồng hành, định hướng và giúp đỡ cho em trong suốt quá trình nghiên cứu và thực hiện đồ án này.

Bên cạnh đó, em xin gửi lời cảm ơn chân thành đến các giảng viên đã và đang giảng dạy tại trường Đại học Bách Khoa Hà Nội, đặc biệt là những giảng viên trong suốt 5 năm qua đã dạy bảo và truyền đạt cho em rất nhiều kiến thức và kinh nghiệm để em có thể hoàn thiện bản thân mình hơn.

Cuối cùng, em xin được gửi lời cảm ơn đến gia đình và bạn bè đã đồng hành và khích lệ em rất nhiều trong quá trình nghiên cứu thực hiện đồ án này.

Trong quá trình xây dựng và hoàn thiện báo cáo cũng như đồ án tốt nghiệp, em sẽ không tránh khỏi những sai sót, vì thế em rất mong các thầy cô và các bạn đọc góp ý để em có thể hoàn thiện hơn nữa ĐATN này.

Em xin chân thành cảm ơn!

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong thời đại mà công nghệ trở nên phổ biến, xã hội ngày càng phát triển, ở tất cả mọi lĩnh vực người ta đều hướng đến những điều chỉnh chu và tốt đẹp nhất. Mà thứ cần chỉnh chu nhất có lẽ chính là các văn bản, bài báo. Dù một văn bản trình bày đẹp, bô cục rõ ràng nhưng sai chính tả thì sẽ làm cho người đọc khó nắm bắt được ý trong văn bản. Đặc biệt, đối với những báo cáo khoa học hay bài báo khoa học thì việc sai chính tả khiến cho bài báo hay báo cáo đó trở nên khó hiểu và giảm bớt tính chuyên nghiệp.

Vậy làm sao để mọi người có thể phát hiện và sửa những lỗi sai chính tả trong các văn bản - đặc biệt là trong các bài báo, báo cáo khoa học - một cách nhanh chóng và chính xác. Xuất phát từ vấn đề trên, nhiệm vụ của đồ án tốt nghiệp (ĐATN) này là phát triển mô đun phát hiện lỗi sai và mô đun sửa lỗi sai. Bên cạnh đó ĐATN còn phát triển và xây dựng một hệ thống web sẽ tích hợp với hệ thống Coopy của trường Công Nghệ Thông Tin và Truyền Thông thuộc trường Đại học Bách Khoa Hà Nội. Hai mô đun trên sẽ được tích hợp trong hệ thống web phát hiện và sửa lỗi chính tả giúp cho sinh viên có thể kiểm tra chính tả và sửa lỗi chính tả trên những báo cáo khoa học. Đồng thời, cải thiện lại giao diện của mô đun kiểm tra sao chép trên hệ thống Coopy của trường.

Để làm được điều đó, trước tiên ĐATN tiếp cận bài toán phát hiện lỗi sai, tiếp theo ĐATN sẽ tiếp cận bài toán sửa lỗi sai. Trong quá trình làm ĐATN, em đã nghiên cứu và thực nghiệm hai mô đun: mô đun phát hiện lỗi sai sử dụng transformer và mô đun sửa lỗi sai sử dụng kết hợp giữa 3 phương pháp là mô hình N-gram, mô hình huấn luyện trước BartPho và tập luật . Trong đó mô đun phát hiện lỗi sai đạt 97.30% trên bộ dữ liệu phoBert và bộ dữ liệu được trích xuất ra từ các file đồ án, còn mô đun sửa lỗi sai đạt 93.55% trên bộ dữ liệu phoBert.

Đồng thời, ĐATN sẽ phát triển hệ thống website phát hiện và chỉnh sửa lỗi chính tả sử dụng công nghệ ReactJS, Flask, Firebase và cải thiện giao diện của phần mềm kiểm tra trùng lặp trên hệ thống Coopy sử dụng ReactJS.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Tổng quan về bài toán phát hiện lỗi chính tả và sửa lỗi chính tả	1
1.3 Các giải pháp hiện tại và hạn chế	2
1.4 Mục tiêu và định hướng giải pháp	3
1.4.1 Mục tiêu.....	3
1.4.2 Định hướng giải pháp	3
1.5 Đóng góp của đồ án	4
1.6 Bố cục đồ án	4
CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT	6
2.1 Ngữ cảnh của bài toán.....	6
2.1.1 Tiếng (âm tiết)	6
2.1.2 Từ	7
2.2 Các kết quả nghiên cứu liên quan	8
2.2.1 Mô đun phát hiện lỗi chính tả	8
2.2.2 Mô đun gợi ý sửa lỗi chính tả	8
2.3 Mô hình huấn luyện BART	8
2.4 Mô hình BartPho	9
2.5 Mô hình Transformer	10
2.5.1 Encoder và Decoder	10
2.5.2 Các tiến trình self-attention và encoder-decoder attention	11
2.6 Định lý Bayes	11
2.7 Thuật toán N-gram	12
2.7.1 Mô hình ngôn ngữ.....	12

2.7.2 Mô hình ngôn ngữ n-gram	12
-------------------------------------	----

CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT..... 14

3.1 Tổng quan giải pháp.....	14
3.2 Mô đun phát hiện lỗi sử dụng transformer.....	16
3.3 Mô đun gợi ý sửa lỗi	16
3.3.1 Mô hình N-gram sử dụng trong sửa lỗi chính tả	18
3.3.2 Mô hình huấn luyện trước BartPho	18
3.3.3 Tập luật sử dụng cho bài toán sửa lỗi chính tả	19
3.3.4 Phương pháp đo độ tương đồng giữa hai từ	20

CHƯƠNG 4. ĐÁNH GIÁ THỰC NGHIỆM..... 21

4.1 Các tham số đánh giá	21
4.2 Phương pháp thí nghiệm.....	21
4.2.1 Thực hiện chuẩn bị dữ liệu	21
4.2.2 Chuẩn bị môi trường.....	22
4.2.3 Tiền xử lý dữ liệu	22
4.2.4 Huấn luyện mô hình Transformer cho mô đun phát hiện lỗi chính tả	23
4.2.5 Huấn luyện mô hình với N-gram cho mô đun gợi ý sửa lỗi	23

4.3 Kết quả của mô hình Transformer cho mô đun phát hiện lỗi chính tả	23
---	----

4.4 Kết quả của mô đun sửa lỗi chính tả	24
---	----

CHƯƠNG 5. PHÁT TRIỂN HỆ THỐNG TRANG WEB THU THẬP VÀ ĐÁNH GIÁ SẢN PHẨM..... 26

5.1 Cải thiện giao diện đạo văn của hệ thống Coopy	26
5.1.1 Khảo sát giao diện hiện tại	26
5.1.2 Cải thiện giao diện	27

5.2 Thiết kế và triển khai hệ thống tích hợp mô đun kiểm tra và sửa lỗi chính tả với hệ thống Coopy	29
5.2.1 Phân tích yêu cầu	29
5.2.2 Tổng quan chức năng	30
5.2.3 Đặc tả chức năng	30
5.2.4 Yêu cầu phi chức năng	34
5.3 Công nghệ sử dụng	34
5.3.1 Frontend	34
5.3.2 Backend	37
5.4 Thiết kế kiến trúc	37
5.5 Thiết kế chi tiết giao diện	38
5.6 Thiết kế chi tiết mô đun kiểm tra bằng file (Check document module)	39
5.7 Thiết kế API	40
5.8 Xây dựng hệ thống	41
5.8.1 Thư viện và công cụ sử dụng	41
5.8.2 Kết quả đạt được	42
5.9 Đóng gói và triển khai	46
CHƯƠNG 6. KẾT LUẬN	47
6.1 Kết luận	47
6.2 Hướng phát triển trong tương lai	48
TÀI LIỆU THAM KHẢO	49
PHỤ LỤC	49

DANH MỤC HÌNH VẼ

Hình 2.1	Cấu tạo của âm tiết	6
Hình 2.2	Cấu tạo của từ	7
Hình 2.3	Sơ đồ kiến trúc transformer kết hợp với attention	10
Hình 3.1	Quy trình thực hiện kiểm tra lỗi chính tả và gợi ý sửa lỗi chính tả	15
Hình 3.2	Model phát hiện lỗi sử dụng transformer [5]	16
Hình 3.3	Mô đun gợi ý sửa lỗi	17
Hình 3.4	Sử dụng mô hình BartPho	19
Hình 4.1	Công thức Precision và Recall	21
Hình 4.2	Độ chính xác của mô hình transformer	23
Hình 4.3	Độ chính xác của mô hình N-gram cho bài toán gợi ý sửa lỗi chính tả	24
Hình 4.4	Độ chính xác của mô hình huấn luyện trước BartPho cho bài toán gợi ý sửa lỗi chính tả	24
Hình 4.5	Độ chính xác của phương pháp sử dụng tập luật cho bài toán gợi ý sửa lỗi chính tả	24
Hình 4.6	Độ chính xác sau khi kết hợp ba phương pháp là mô hình n-gram, mô hình huấn luyện trước BartPho và tập luật	24
Hình 5.1	Giao diện hiện tại của hệ thống Coopy	26
Hình 5.2	Giao diện chính sau khi cải thiện	27
Hình 5.3	Giao diện mục tài liệu trên mạng	28
Hình 5.4	Giao diện mục tài liệu loại bỏ	28
Hình 5.5	Giao diện sau khi cải thiện của hệ thống Coopy	29
Hình 5.6	Biểu đồ use case tổng quan của hệ thống	30
Hình 5.7	Vòng đời của ReactJS	35
Hình 5.8	Kiến trúc chung của hệ thống	38
Hình 5.9	Thiết kế mockup giao diện	38
Hình 5.10	Thiết kế tính năng tải lên file để kiểm tra chính tả	39
Hình 5.11	Thiết kế tính năng tải file xuống sau khi sửa lỗi chính tả	40
Hình 5.12	Giao diện trang chủ	42
Hình 5.13	Giao diện chức năng kiểm tra lỗi chính tả bằng input	43
Hình 5.14	Giao diện hiển thị kết quả của chức năng kiểm tra lỗi chính tả bằng cách nhập văn bản từ bàn phím	44
Hình 5.15	Giao diện chức năng kiểm tra lỗi chính tả bằng file	45

Hình 5.16 Giao diện file sau khi đã được kiểm tra lỗi sai 45

DANH MỤC BẢNG BIỂU

Bảng 5.1	Danh sách use case	30
Bảng 5.2	Đặc tả use case "Kiểm tra lỗi chính tả bằng file"	31
Bảng 5.3	Đặc tả use case "Kiểm tra lỗi chính tả bằng văn bản người dùng nhập"	32
Bảng 5.4	Đặc tả use case "Sửa lỗi chính tả trên file"	32
Bảng 5.5	Đặc tả use case "Sửa lỗi chính tả trên văn bản người dùng nhập"	33
Bảng 5.6	Đặc tả use case "Tải file đã được chỉnh sửa"	34
Bảng 5.7	Danh sách API của hệ thống	41
Bảng 5.8	Danh sách công cụ và thư viện sử dụng	41

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Chương này tập trung giới thiệu về những vấn đề thực tế dẫn tới việc chọn đề tài, tổng quan về bài toán phát hiện và sửa lỗi chính tả. Tiếp theo đưa ra mục tiêu và phạm vi của đồ án, định hướng giải pháp, đóng góp và bối cảnh trình bày của đồ án.

1.1 Đặt vấn đề

Báo cáo nghiên cứu khoa học là mẫu báo cáo quan trọng đối với nhiều sinh viên đang thực hiện nghiên cứu khoa học. Bài báo cáo này sẽ đánh giá chất lượng và kết quả của bài nghiên cứu khoa học sau quá trình thực hiện nên cần chú trọng và làm cẩn thận. Nhưng hiện tại báo cáo khoa học đang gặp 2 vấn đề là sao chép không có trích dẫn và sai lỗi chính tả.

Đối với vấn đề báo cáo khoa học sao chép không có trích dẫn đã được các nhà khoa học của trường Công Nghệ Thông Tin và Truyền Thông thuộc trường Đại học Bách Khoa Hà Nội giải quyết bằng việc phát triển hệ thống COOPY áp dụng các công nghệ tiên tiến về AI (xử lý ngôn ngữ tự nhiên, học máy / học sâu). Đây là sản phẩm phần mềm bao gồm lỗi công nghệ trí tuệ nhân tạo, đã được ứng dụng tại trường Công Nghệ Thông Tin và Truyền Thông và một số trường khác thuộc trường Đại học Bách Khoa Hà Nội tạo hiệu quả rất tích cực trong việc từng bước nâng cao chất lượng tài liệu khoa học.

Hệ thống đã giải quyết được vấn đề kiểm tra việc báo cáo khoa học không có trích dẫn nhưng chưa có chức năng phát hiện và sửa lỗi chính tả. Ngoài ra, phần giao diện của chức năng kiểm tra sao chép còn đơn giản, chưa thân thiện với người dùng.

Chính vì vậy, nên trong ĐATN này em muốn cải thiện giao diện của chức năng kiểm tra sao chép trở nên thân thiện với người dùng hơn, xây dựng mô đun kiểm tra lỗi chính tả và mô đun sửa lỗi chính tả để nâng cao chất lượng báo cáo khoa học và cuối cùng là xây dựng hệ thống sử dụng hai mô đun để có thể tích hợp được vào hệ thống Coopy kiểm tra sao chép.

Hai mô đun kiểm tra lỗi chính tả và mô đun sửa lỗi chính tả sẽ được giải quyết thông qua bài toán phát hiện lỗi chính tả và sửa lỗi chính tả.

1.2 Tổng quan về bài toán phát hiện lỗi chính tả và sửa lỗi chính tả

Bài toán phát hiện và sửa lỗi sai chính tả (SC) sẽ gồm hai bài toán nhỏ là phát hiện lỗi sai và bài toán sửa lỗi chính tả.

Bài toán phát hiện lỗi sai chính tả là một bài toán rất được quan tâm trong xử lý ngôn ngữ tự nhiên. Bài toán có đầu vào là một văn bản hay một đoạn văn, đầu ra sẽ là những từ bị sai chính tả.

Bên cạnh bài toán phát hiện lỗi sai, thì bài toán sửa lỗi chính tả cũng là một bài toán đang được quan tâm nhiều trong lĩnh vực xử lý ngôn ngữ tự nhiên. Bài toán có đầu vào là những từ bị sai chính tả và đầu ra là những từ đúng chính tả có thể thay thế cho từ đã bị sai ở đầu vào.

Trong thời đại mà công nghệ ngày càng phát triển thì những bài báo hay báo cáo khoa học cũng ngày càng nhiều. Nhưng việc đảm bảo một bài báo cáo hay một bài báo không xảy ra lỗi chính tả là điều khó có thể kiểm soát được. Vì những lỗi chính tả thông thường có thể là do lỗi đánh máy hay do lỗi từ vựng địa phương gây ra. Chính vì vậy nhu cầu kiểm tra lỗi chính tả cho các bài báo hay báo cáo khoa học ngày càng trở nên cần thiết. Nhưng khi đã biết lỗi sai chính tả ở đâu thì người dùng hay nhóm nghiên cứu khoa học còn muốn sửa lỗi sai chính tả đó bằng cách gợi ý từ đúng hay sửa trực tiếp các từ sai chính tả đó. Để giải quyết hai vấn đề đó người dùng cần một hệ thống có thể tự động phát hiện lỗi sai và sửa lỗi sai trên các văn bản khoa học một cách nhanh chóng và chính xác.

Hiện nay, có một số trang web chỉ dừng ở mức phát hiện lỗi sai cho người dùng nhưng lại chưa đưa ra gợi ý sửa lỗi sai dẫn tới người dùng sẽ không biết cách để sửa những lỗi sai đó.

Đặc biệt, bài toán phát hiện lỗi sai và sửa lỗi sai đang được quan tâm ở rất nhiều lĩnh vực khác nhau, đặc biệt là các lĩnh vực liên quan tới việc phải viết văn bản chuyên môn.

1.3 Các giải pháp hiện tại và hạn chế

Đối với bài toán phát hiện và sửa lỗi sai chính tả đã có giải pháp tiếp cận là sử dụng mô hình Seq2Seq (sequence-to-sequence) hay sử dụng mô hình ngôn ngữ N-gram kết hợp với khoảng cách Levenshtein.

Nhưng đối với giải pháp Seq2Seq lại có hạn chế là sửa lỗi luôn ngay sau khi phát hiện lỗi dẫn tới có thể sẽ bị sửa sai ý của người dùng.

Còn đối với giải pháp sử dụng mô hình N-gram kết hợp với khoảng cách Levenshtein lại có hạn chế là chỉ sử dụng từ điển nên không có ngữ cảnh cho câu dẫn tới có thể sẽ tìm và sửa lỗi bị sai ngữ cảnh của câu văn.

Đối với riêng bài toán phát hiện lỗi sai thì ngoài sử dụng mô hình Seq2Seq, N-gram thì còn các cách tiếp cận khác như sử dụng RNN cùng LSTM. Nhưng do phải xử lý câu đầu vào một cách tuần tự nên nhược điểm của RNN và LSTM là tốc

độ xử lý chậm và hạn chế trong việc biểu diễn sự phụ thuộc xa giữa các từ trong một câu.

Đối với bài toán sửa lỗi sai chính tả chúng ta có nhiều giải pháp tiếp cận, nhưng thuận tiện nhất là đưa ra gợi ý thì hiện tại có giải pháp là dùng fill mask. Trong đó, Fill mask là nhiệm vụ che một số từ trong câu và dự đoán từ nào sẽ thay thế các mặt nạ đó. Ngoài ra mô hình PhoBERT là mô hình huấn luyện trước sử dụng kiến trúc của BERT nên có thể tạo ra các biểu diễn từ, từ quá trình ẩn các vị trí token một cách ngẫu nhiên trong câu input và dự báo chính chính từ đó ở output dựa trên bối cảnh là các từ xung quanh. Như vậy khi đã biết các từ xung quanh, chúng ta hoàn toàn có thể dự báo được từ phù hợp nhất với vị trí đã được masking.

Vấn đề của PhoBERT là mô hình chỉ cho phép gợi ý 1 vị trí mask trong câu. Tức là 1 câu chỉ có thể gợi ý cho 1 vị trí. Chính vì vậy, đối với những câu sai nhiều hơn 2 lỗi chính tả, tức là cần gợi ý cho từ 2 vị trí trở lên thì các mô hình này chưa đáp ứng được.

1.4 Mục tiêu và định hướng giải pháp

1.4.1 Mục tiêu

Từ các vấn đề đặt ra ở phần 1.2 và những mong muốn trong phần 1.1 thì mục tiêu của ĐATN là:

- Cải thiện giao diện của hệ thống phát hiện sao chép Coopy của trường Công Nghệ Thông Tin và Truyền Thông.
- Xây dựng và triển khai mô đun phát hiện lỗi sai chính tả.
- Xây dựng và triển khai mô đun sửa lỗi sai chính tả.
- Xây dựng và phát triển hệ thống phát hiện và sửa lỗi chính tả.

Sau khi xây dựng và thực nghiệm hai mô đun phát hiện và sửa lỗi chính tả, em sẽ tiến hành phát triển một hệ thống sử dụng hai mô đun để nhằm tích hợp với hệ thống phát hiện sao chép của trường Công Nghệ Thông Tin và Truyền Thông. Nhằm giúp cho sinh viên có thể một bài báo cáo chỉnh chu hơn.

1.4.2 Định hướng giải pháp

Từ các mục tiêu đã nêu trong phần 1.4.1, em đề xuất định hướng giải pháp theo hướng sau: (i) Sửa giao diện hệ thống phát hiện sao chép. (ii) Xây dựng mô đun cho bài toán phát hiện lỗi chính tả. (iii) Xây dựng mô đun cho bài toán gợi ý sửa lỗi. Cuối cùng, (iv) Xây dựng hệ thống kiểm tra và sửa lỗi chính tả.

Trước tiên, do hiện tại hệ thống phát hiện sao chép có giao diện không thân thiện với người dùng và hệ thống sử dụng công nghệ ReactJS. Nên em sẽ thừa kế và cải

thiện tiếp giao diện hệ thống giúp cho giao diện trở nên thân thiện với người dùng hơn.

Tiếp theo, đối với mô đun phát hiện lỗi chính tả thì em lựa chọn mô hình Transformer để giải quyết vấn đề phát hiện lỗi chính tả.

Đối với mô đun cho bài toán gợi ý sửa lỗi chính tả, em kết hợp giữa 3 phương pháp là BartPho, N-gram và tập luật để xây dựng mô đun.

Về tập luật được nêu ở trên, em xây dựng tập luật dựa vào những nguyên nhân gây ra lỗi sai chính tả và những lỗi sai thường gặp để tạo một tập các luật để tăng độ chính xác cho việc đưa ra gợi ý sửa lỗi. Phương pháp này sẽ hiệu quả khi đầu vào là một từ đơn.

Bên cạnh đó, ĐATN hướng tới sử dụng hai mô đun cho hệ thống kiểm tra và sửa lỗi chính tả. Chính vì vậy, ĐATN sẽ hướng tới xây dựng và phát triển một hệ thống kiểm tra và sửa lỗi chính tả bằng cách sử dụng các công nghệ ReactJS cho phần giao diện, Flask cho phần máy chủ. Bên cạnh đó, em sử dụng Firestore (một công nghệ của Google) để lưu trữ dữ liệu cho hệ thống.

1.5 Đóng góp của đồ án

Đồ án này có 3 đóng góp chính như sau:

- Cải thiện giao diện của hệ thống phát hiện đạo văn của hệ thống Coopy để thân thiện với người dùng hơn.
- Thiết kế và triển khai tích hợp mô đun kiểm tra và sửa lỗi chính tả vào hệ thống Coopy.
- Xây dựng tập luật cho gợi ý lỗi chính tả.

1.6 Bố cục đồ án

Phần còn lại của báo cáo đồ án tốt nghiệp này được tổ chức như sau.

Chương 2 trình bày về tổng quan ngữ cảnh của bài toán, cùng với những kiến thức nền tảng bao gồm mô hình huấn luyện trước BART, mô hình BartPho, mô hình Transformer, một số thuật toán như N-gram. Đây là tiền đề để hiểu được các giải pháp được trình bày trong các chương tiếp theo.

Trong Chương 3, em trình bày về tổng quan giải pháp cho bài toán phát hiện và sửa lỗi chính tả mà ĐATN hướng tới. Từ tổng quan giải pháp đó sẽ trình bày chi tiết các thuật toán, mô hình đã mô tả trong tổng quan giải pháp.

Trong Chương 4 này sẽ trình bày về các tham số đánh giá cho bài toán, phương pháp thí nghiệm mà ĐATN hướng tới. Bên cạnh đó, chương này sẽ nêu ra các kết

quả mà ĐATN đạt được.

Tiếp theo, trong Chương 5 sẽ trình bày chi tiết về hệ thống phát hiện và sửa lỗi chính tả bao gồm: Phần (i) Phân tích tổng quát yêu cầu của hệ thống, (ii) Trình bày những công nghệ sử dụng nhằm xây dựng hệ thống, (iii) Tổng quan về kiến trúc của hệ thống, (iv) Khái quát về giao diện của hệ thống sẽ hướng tới. Tiếp theo, phần (v) và phần (vi) lần lượt trình bày thiết kế máy chủ và thiết kế api của hệ thống. Cuối cùng, phần (vii) và phần (viii) sẽ bàn về xây dựng hệ thống và đóng gói hệ thống.

Tiếp theo, Chương 6 sẽ trình bày lại các vấn đề mà ĐATN đã giải quyết được, những vấn đề còn tồn đọng, chưa giải quyết, từ đó đưa ra hướng phát triển trong tương lai.

Cuối cùng, trong Chương 7 sẽ trình bày về toàn bộ tài liệu tham khảo trong ĐATN này.

Sau đây là chi tiết của từng chương của ĐATN này.

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

Chương 1 đã nêu ra các vấn đề hiện tại, cùng với mục tiêu và giải pháp cho đồ án này. Chương này đi sâu vào việc trình bày ngữ cảnh của bài toán phát hiện và sửa lỗi sai chính tả. Đặc biệt, tập trung vào việc trình bày cơ sở lý thuyết, đây là cơ sở để hiểu được có giải pháp và kết quả được nêu ở Chương 3 và Chương 4. Các nội dung có trong chương này bao gồm: (i) Ngữ cảnh của bài toán, (ii) Mô hình huấn luyện trước BART, (iii) Mô hình BartPho, (iv) Mô hình Transformer, (v) Thuật toán N-gram. Ngoài ra phần cơ sở lý thuyết dành cho việc phát triển hệ thống sẽ được nêu rõ trong Chương 5.

2.1 Ngữ cảnh của bài toán

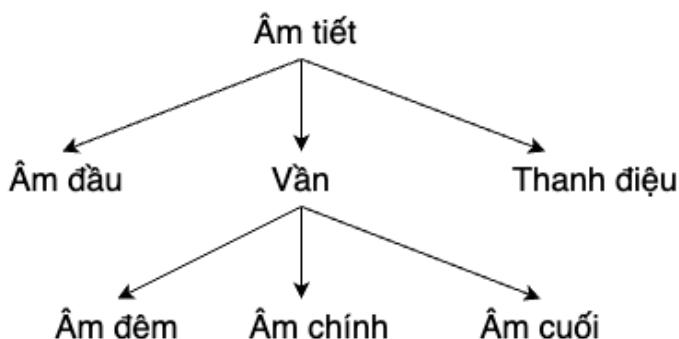
Tiếng Việt là một ngôn ngữ đơn lập, đặc điểm này bao quát toàn bộ tiếng Việt về mặt ngữ âm.

2.1.1 Tiếng (âm tiết)

Là đơn vị cơ sở của ngữ pháp

Về mặt ngữ âm: trong tiếng Việt, tiếng (âm tiết) là đơn vị phát âm tự nhiên nhỏ nhất, người ta dùng khoảng trống để phân định âm tiết.

Về mặt sử dụng: trong tiếng Việt, tiếng có thể là từ hoặc yếu tố cấu tạo từ.



Hình 2.1: Cấu tạo của âm tiết

Có 3 bộ phận chính là: âm đầu, vần và thanh điệu

Âm đầu: Có những cách mở đầu âm tiết khác nhau (tắc, xát, rung), chúng có tác dụng khu biệt các âm tiết. Ví dụ: toán – hoán

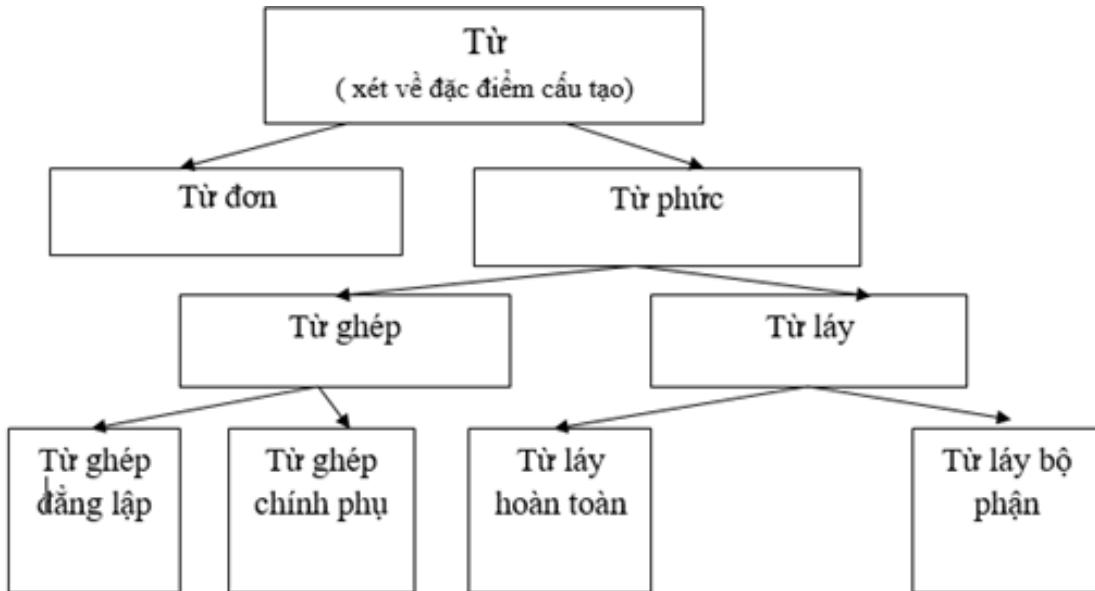
Thanh điệu: Có tác dụng khu biệt âm tiết về cao độ. Mỗi âm tiết có một trong 6 thanh điệu. Ví dụ: toán – toàn

Vần: Âm đệm có tác dụng biến đổi âm sắc của âm tiết sau lúc mở đầu, nó có chức năng khu biệt các âm tiết. Ví dụ: toán – tán. Âm chính: Mang âm sắc chủ đạo

của âm tiết và là hạt nhân của âm tiết. Ví dụ: túy – túi. Âm cuối: Có chức năng kết thúc âm tiết với nhiều cách khác nhau (tắc, không tắc...) làm thay đổi âm sắc của âm tiết và do đó để phân biệt âm tiết này với âm tiết khác. Ví dụ: bàn – bài

2.1.2 Từ

Là đơn vị ngôn ngữ nhỏ nhất có nghĩa, được tạo nên bởi tiếng, dùng để đặt câu. Có từ gồm một tiếng, có từ gồm hai tiếng trở lên.



Hình 2.2: Cấu tạo của từ

Từ đơn: là những từ chỉ có một tiếng tạo thành. Ngoại trừ một số trường hợp đặc biệt như từ mượn nước ngoài (ghi-dông, tivi, ra-đa,...) được xếp vào từ đơn đa âm tiết.

Từ phức là từ gồm ít nhất hai tiếng tạo thành. Ví dụ: cha mẹ, chó mèo, cây cối, mưa gió, lạnh lẽo, sạch sành sanh,... Từ ghép đẳng lập có từ 2 tiếng trở lên mà 2 tiếng đó có nghĩa ngang bằng, có thể tách ra để tạo 1 từ khác riêng biệt. Ví dụ: học tập, ăn uống... Từ ghép chính phụ là từ có 2 tiếng trở lên trong đó tiếng phụ bổ sung nghĩa cho tiếng chính, không thể tách thành từ đơn được. Ví dụ: xanh lơ, bà ngoại,...

Từ láy là từ được cấu tạo bằng cách láy lại (điệp lại) một phần phụ âm hoặc nguyên âm, hay toàn bộ tiếng ban đầu. Từ láy hoàn toàn: Là loại từ được láy giống nhau cả phần âm, vần, dấu câu ví dụ như xanh xanh, ào ào. Đôi khi để nhấn mạnh một âm thanh hay hành động mà dấu câu có thể khác nhau như thăm thăm, lanh lanh... Từ láy bộ phận thì có láy âm đầu và láy vần. Láy âm đầu: Là những từ láy lặp lại về phần âm đầu – người ta thường gọi là từ láy âm như: Lắp lánh, xinh xắn, ngọt ngác,... Láy vần: Từ láy lặp lại về phần vần – người ta thường gọi là từ láy vần

như: Liêu xiêu, cheo leo, chênh vênh, chao đảo...

2.2 Các kết quả nghiên cứu liên quan

2.2.1 Mô đun phát hiện lỗi chính tả

RNN (Recurrent Neural Network) và LSTM (Long Short-term memory) là các phương pháp tiếp cận hiện đại thường được sử dụng trong mô hình về xử lý ngôn ngữ tự nhiên. Từ đó đã có nhiều nỗ lực cải tiến mô hình ngôn ngữ và kiến trúc mã hóa - giải mã. Nhưng do phải xử lý câu đầu vào một cách tuần tự nên nhược điểm của RNN và LSTM là tốc độ xử lý chậm và hạn chế trong việc biểu diễn sự phụ thuộc xa giữa các từ trong một câu [1].

2.2.2 Mô đun gợi ý sửa lỗi chính tả

Để đưa ra gợi ý cách sửa cho từ sai, hiện tại có 1 số giải pháp như Fill mask, N-gram. Fill mask là nhiệm vụ che một số từ trong câu và dự đoán từ nào sẽ thay thế các mặt nạ đó. Mô hình BERT (Bidirectional Encoder Representation from Transformer) tạo ra các biểu diễn từ từ quá trình ẩn các vị trí token một cách ngẫu nhiên trong câu input và dự báo chính xác từ đó ở output dựa trên bối cảnh là các từ xung quanh. Như vậy khi đã biết các từ xung quanh, chúng ta hoàn toàn có thể dự báo được từ phù hợp nhất với vị trí đã được masking. Vấn đề của PhoBert là mô hình chỉ cho phép gợi ý 1 vị trí mask trong câu. Tức là 1 câu chỉ có thể gợi ý cho 1 vị trí. Chính vì vậy, đối với những câu sai nhiều hơn 2 lỗi chính tả, tức là cần gợi ý cho từ 2 vị trí trở lên thì các mô hình này chưa đáp ứng được.

Mô hình N-gram [2] dựa vào các âm tiết xung quanh để đưa ra gợi ý sửa lỗi. Để chọn âm tiết tốt nhất trong tập hợp nhầm lẫn (tập hợp các ứng cử viên để sửa), chúng ta cần đo lường mối quan hệ giữa một âm tiết và các âm tiết lân cận của nó. Trong trường hợp này, mô hình N-gram rất hữu ích để mô hình hóa các mối quan hệ này. Mô hình này được tạo ra dựa trên phương pháp thống kê. Do đó, sự phân bố và đa dạng từ vựng trên dữ liệu đào tạo đã ảnh hưởng đáng kể đến hiệu suất của hệ thống.

Ưu điểm: Cài đặt đơn giản và hoàn toàn áp dụng tư tưởng của mô hình N-gram. Đã đưa được yếu tố ngữ cảnh vào khi thực hiện.

Nhược điểm: Không đánh giá được ngữ cảnh của toàn bộ câu, do đó, trong nhiều trường hợp, nó không thể đưa ra một xác suất chính xác.

2.3 Mô hình huấn luyện BART

BART [3] là mô hình được giới thiệu bởi Facebook AI, một mô hình đã huấn luyện trước mới, kết hợp ưu điểm của BERT và GPT. Sức mạnh của BERT nằm ở việc nắm bắt ngữ cảnh hai chiều, trong khi đó GPT có khả năng tự hồi quy. Với sự

ra đời của BART, các nhiệm vụ sinh và đọc hiểu văn bản có thể được thực hiện với cùng một mô hình.

BART là một autoencoder khử nhiễu trên kiến trúc seq2seq ((sequence-to-sequence), có thể được áp dụng trong đa dạng các nhiệm vụ khác nhau. Nó sử dụng kiến trúc transformers chuẩn cho bài toán dịch máy. Việc huấn luyện BART bao gồm việc tạo nhiễu trong văn bản với một hàm tùy ý và sử dụng mô hình để tái cấu trúc lại văn bản ban đầu. Ưu điểm chính của cách thức này là mô hình trở nên linh hoạt với văn bản đầu vào và tái tạo lại văn bản một cách hiệu quả.

BART cho thấy hiệu quả vượt trội trong cả nhiệm vụ sinh lẩn đọc hiểu văn bản. Cụ thể, BART có hiệu quả sánh ngang RoBERTa trên GLUE và SQuAD và đạt SOTA trong các nhiệm vụ về đối thoại trừu tượng, trả lời câu hỏi và tóm tắt.

Trong đó:

- GLUE score (General Language Understanding Evaluation score) là tiêu chuẩn được sử dụng để đào tạo, đánh giá và phân tích các hệ thống hiểu ngôn ngữ tự nhiên dựa trên trí tuệ nhân tạo.
- SOTA (State-of-the-art) là mức độ phát triển cao nhất của một công nghệ, một lĩnh vực khoa học, hoặc một thiết kế nào đó đạt được trong một khoảng thời gian nhất định.

Giống như các mô hình Transformer, BART gồm hai thành phần là Encoder và Decoder. Encoder được lấy từ BERT, nó có thể mã hóa xâu đầu vào theo cả hai chiều và lấy được nhiều thông tin ngữ cảnh hơn. Một số lượng ngẫu nhiên các token được che bằng mặt nạ và mô hình phải tự khôi phục chúng.

Decoder từ GPT được sử dụng để tái tạo lại đầu vào bị nhiễu. Mặc dù vậy, các từ chỉ có thể sinh từ bên trái, mô hình không thể học được tương tác hai chiều. Hàm kích hoạt GeLU được sử dụng thay thế cho ReLU. Kiến trúc cơ bản của BART sử dụng 6 tầng encoder và decoder trong khi kiến trúc mở rộng sử dụng 12 tầng.

2.4 Mô hình BartPho

BartPho là mô hình đào tạo trước được phát triển bởi Nguyễn Lương Trân, Dương Minh Lê và Đạt Quốc Nguyễn thuộc VinAI.

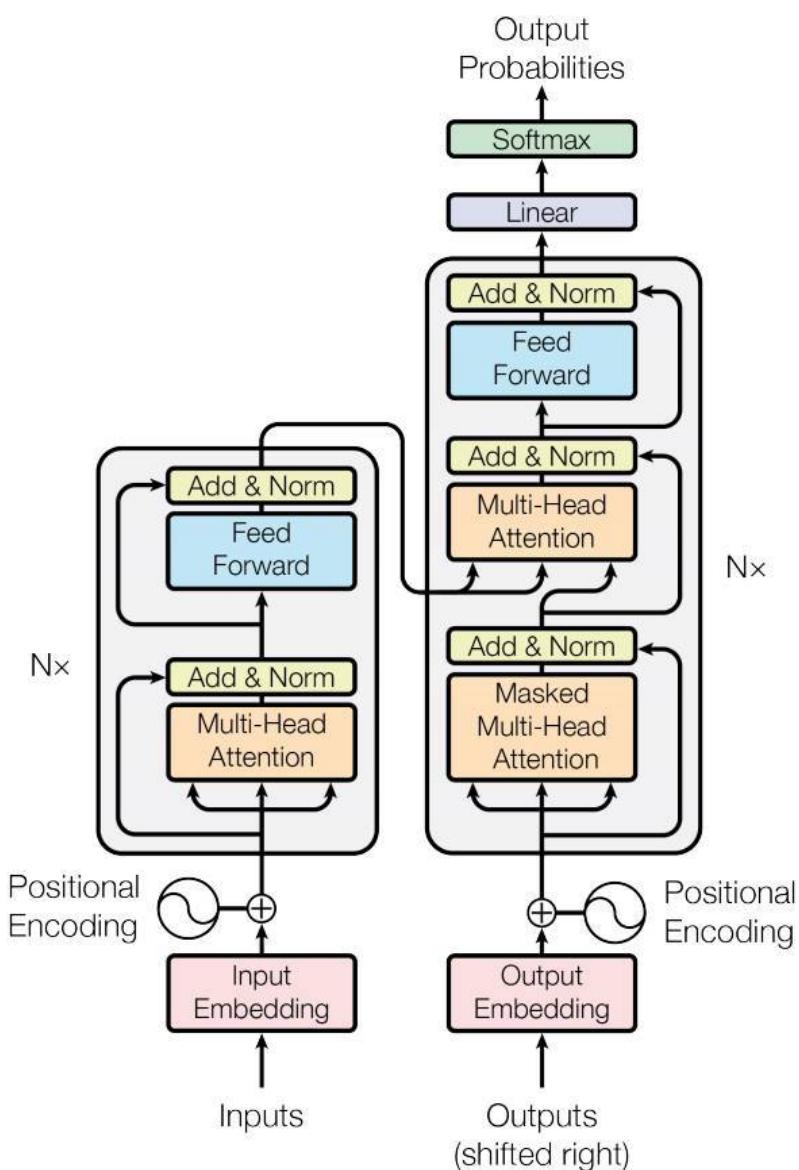
BartPho [4] với hai phiên bản BartPho syllable và BartPho word, là những mô hình seq2seq đơn ngữ có quy mô lớn đầu tiên được đào tạo trước cho tiếng việt. BartPho sử dụng kiến trúc “lớn” (“large”) và sơ đồ đào tạo trước của mô hình seq2seq biểu thị trình tự động mã hóa BART, do đó mô hình đặc biệt thích hợp cho các nhiệm vụ NLP chung.

BartPho word sử dụng 20GB văn bản không nén của phoBERT (khoảng 145 triệu câu phân đoạn tự động)

2.5 Mô hình Transformer

2.5.1 Encoder và Decoder

Transformer là một lớp mô hình seq2seq gồm hai phân đoạn encoder và decoder. Mô hình hoàn toàn không sử dụng các kiến trúc Recurrent Neural Network của RNN mà chỉ sử dụng các lớp attention để nhúng (embedding) các từ trong câu. Kiến trúc cụ thể của mô hình được mô tả trong **Hình 2.3**.



Hình 2.3: Sơ đồ kiến trúc transformer kết hợp với attention

Mô hình bao gồm hai giai đoạn:

- Encoder: Bao gồm 6 layers liên tiếp nhau. Mỗi một layer sẽ bao gồm một sub-layer là Multi-Head Attention kết hợp với fully-connected layer như mô tả ở nhánh encoder bên trái của hình vẽ. Kết thúc quá trình encoder ta thu được một vector nhúng xuất ra cho mỗi từ.
- Decoder: Kiến trúc cũng bao gồm các layers liên tiếp nhau. Mỗi một layer của Decoder cũng có các sub-layers gần tương tự như layer của Encoder nhưng bổ sung thêm sub-layer đầu tiên là Masked Multi-Head Attention có tác dụng loại bỏ các từ trong tương lai khỏi quá trình attention.

2.5.2 Các tiến trình self-attention và encoder-decoder attention

Trong kiến trúc transformer chúng ta áp dụng hai dạng attention khác nhau tại từng bước huấn luyện.

- Self-attention: Được sử dụng trong cùng một câu nhập, tại encoder hoặc tại decoder. Đây chính là attention được áp dụng tại các Multi-Head Attention ở đầu vào của cả hai giai đoạn encoder và decoder.
- Encoder-decoder attention: Sở dĩ được gọi là encoder-decoder attention vì đây là kiến trúc attention tương tác giữa các vec tơ nhúng của encoder và decoder. Vec tơ context được tính toán trên encoder, đã được tính tương quan với vec tơ decoder nên sẽ có ý nghĩa giải thích bối cảnh của từ tại vị trí time step decoder tương ứng. Sau khi kết hợp giữa vec tơ context và vec tơ decoder ta sẽ chiếu tiếp qua một fully connected layer để tính phân phối xác suất cho đầu ra.

Mặc dù kiến trúc chỉ gồm các biến đổi attention nhưng Transformer lại có kết quả rất tốt trong các tác vụ NLP như sentiment analysis và dịch máy.

2.6 Định lý Bayes

Định lý Bayes (Bayes' Theorem) là một định lý toán học để tính xác suất xảy ra của một sự kiện ngẫu nhiên A khi biết sự kiện liên quan B đã xảy ra. Định lý này đặt theo tên nhà toán học Thomas Bayes, người Anh sống ở thế kỷ 18. Đây là một trong những công cụ vô cùng hữu ích, người bạn thân của các Data Scientist, những người làm trong ngành khoa học dữ liệu.

Công thức định lý Bayes

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} = \frac{P(H \cap E)}{P(E)} \quad (2.1)$$

Trong đó:

$P(H|E)$: xác suất H xảy ra khi biết E xảy ra

$P(H)$, $P(E)$: xác suất xảy ra sự kiện H, E

$P(E|H)$: xác suất E xảy ra khi biết H xảy ra

$P(HE)$: xác suất kết hợp sự kiện H và E

2.7 Thuật toán N-gram

2.7.1 Mô hình ngôn ngữ

Là phân bố xác suất trên tập các văn bản, cho biết xác suất của một câu (hoặc 1 cụm từ) thuộc 1 ngôn ngữ là bao nhiêu. Mô hình ngôn ngữ tốt sẽ đánh giá đúng các câu đúng ngữ pháp, trôi chảy hơn các từ có thứ tự ngẫu nhiên.

Ví dụ: $P(\text{"hôm nay trời đẹp"}) > P(\text{"trời đẹp nay hôm"})$

2.7.2 Mô hình ngôn ngữ n-gram

Mục tiêu: Xác định xác suất của 1 câu hoặc một cụm từ:

$$P(W) = P(w_1, w_2, w_3, w_4, w_5, \dots, w_m) \quad (2.2)$$

Theo công thức Bayes:

$$P(A, B) = P(B|A) * P(A) \quad (2.3)$$

$$P(w_1, w_2, w_3, w_4, w_5, \dots, w_m) = P(w_1) * P(w_2|w_1) * P(w_3|w_1w_2) * \dots * P(w_m|w_1w_2w_3\dots w_{m-1}) \quad (2.4)$$

Ví dụ:

$$P(\text{"hôm nay trời đẹp"}) = P(\text{hôm}) * P(\text{nay|hôm}) * P(\text{trời|hôm nay}) * P(\text{đẹp|hôm nay trời})$$

a, Mô hình bigram

$$P(w_n|w_1, w_2, \dots, w_{n-1}) \approx \prod_{i=1}^n P(w_i, w_{i-1}) \quad (2.5)$$

Để đơn giản hóa, sử dụng đánh giá Maximum Likelihood

b, Mô hình trigram

$$P(w_n|w_1, w_2, \dots, w_{n-1}) \approx \prod_{i=1}^n P(w_i, w_{i-2}w_{i-1}) \quad (2.6)$$

Kết chương

Chương này đã trình bày tổng quan về một số cơ sở lý thuyết được sử dụng trong đồ án. Bên cạnh đó Chương 2 còn nêu ra ngữ cảnh của bài toán phát hiện và sửa lỗi chính tả. Từ những cơ sở lý thuyết này sẽ làm tiền đề cho phần đề xuất của đồ án trong chương tiếp theo - Chương 3.

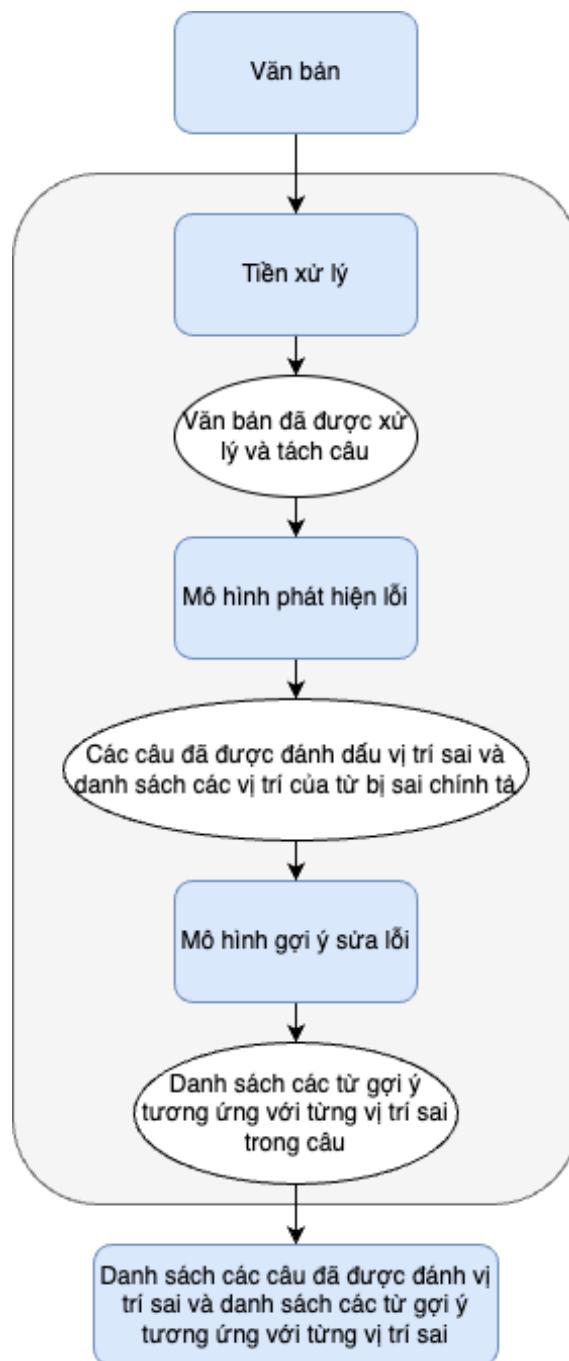
CHƯƠNG 3. PHƯƠNG PHÁP ĐỀ XUẤT

Trong Chương 2 em đã trình bày về cơ sở lý thuyết của các phương pháp liên quan tới bài toán phát hiện và sửa lỗi chính tả trong ĐATN. Trong Chương này em sẽ khái quát về giải pháp ĐATN hướng tới, cùng với những nội dung có trong giải pháp. Về giải pháp cho mục tiêu cải thiện hệ thống Coopy của trường Công Nghệ Thông Tin và Truyền Thông sẽ được trình bày trong Chương 5.

3.1 Tổng quan giải pháp

Bài toán phát hiện và sửa lỗi chính tả trước đây đã có nhiều đề xuất giải quyết bằng các phương pháp dựa vào bản chất ngôn ngữ. Tuy nhiên, do sự phức tạp của ngôn ngữ tự nhiên, sự nhập nhằng nghĩa của từ và cụm từ, sự phụ thuộc cú pháp và ngữ nghĩa của các từ vào ngữ cảnh, nên các phương pháp nêu trên còn hạn chế về tính chính xác.

Dưới đây em xin đề xuất giải pháp tổng quan thực hiện việc kiểm tra lỗi chính tả và gợi ý sửa lỗi chính tả:



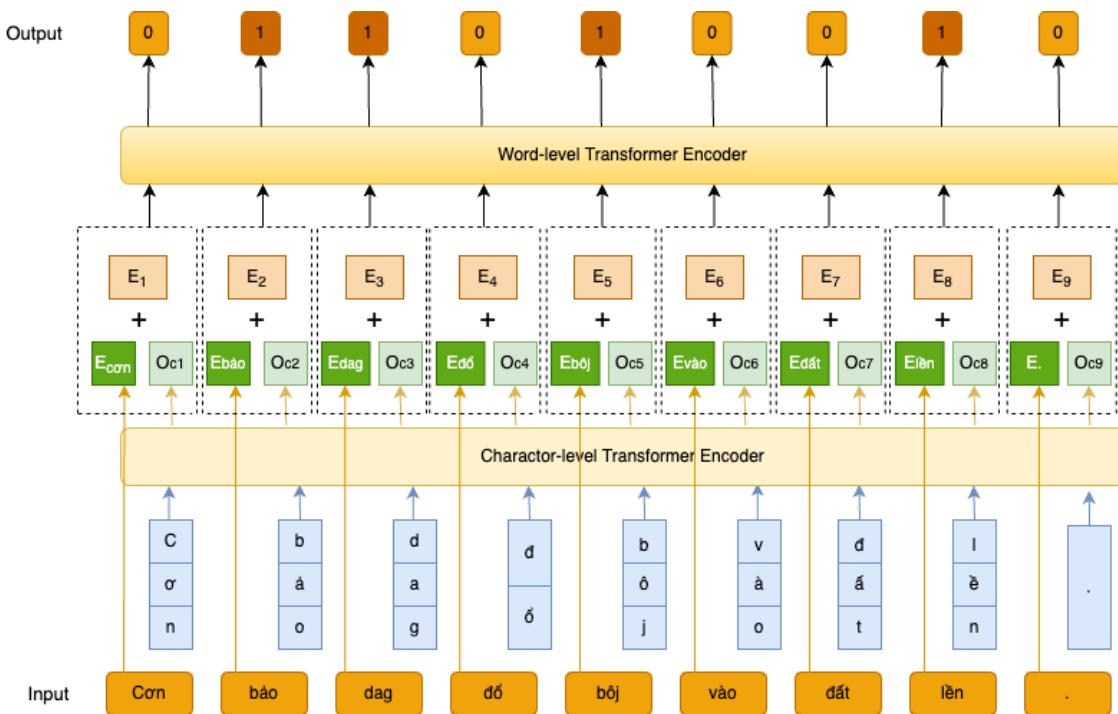
Hình 3.1: Quy trình thực hiện kiểm tra lỗi chính tả và gợi ý sửa lỗi chính tả

Từ **Hình 3.1** em đưa ra giải pháp cho ĐATN này như sau:

Đầu vào của mô hình là một văn bản và đầu ra là danh sách các câu đã được đánh vị trí sai và danh sách các từ gợi ý tương ứng với từng vị trí sai.

Mô hình sẽ được chia làm hai mô đun là mô đun phát hiện lỗi và mô đun gợi ý sửa lỗi. Đầu vào của mô đun phát hiện lỗi là văn bản đã được xử lý và tách câu. Đầu ra là các câu được đánh vị trí sai và danh sách các vị trí của từ bị sai chính tả và nó chính là đầu vào của mô đun gợi ý sửa lỗi. Đầu ra của mô đun gợi ý sửa lỗi là danh sách các từ gợi ý tương ứng với từng vị trí sai trong câu.

3.2 Mô đun phát hiện lỗi sử dụng transformer



Hình 3.2: Model phát hiện lỗi sử dụng transformer [5]

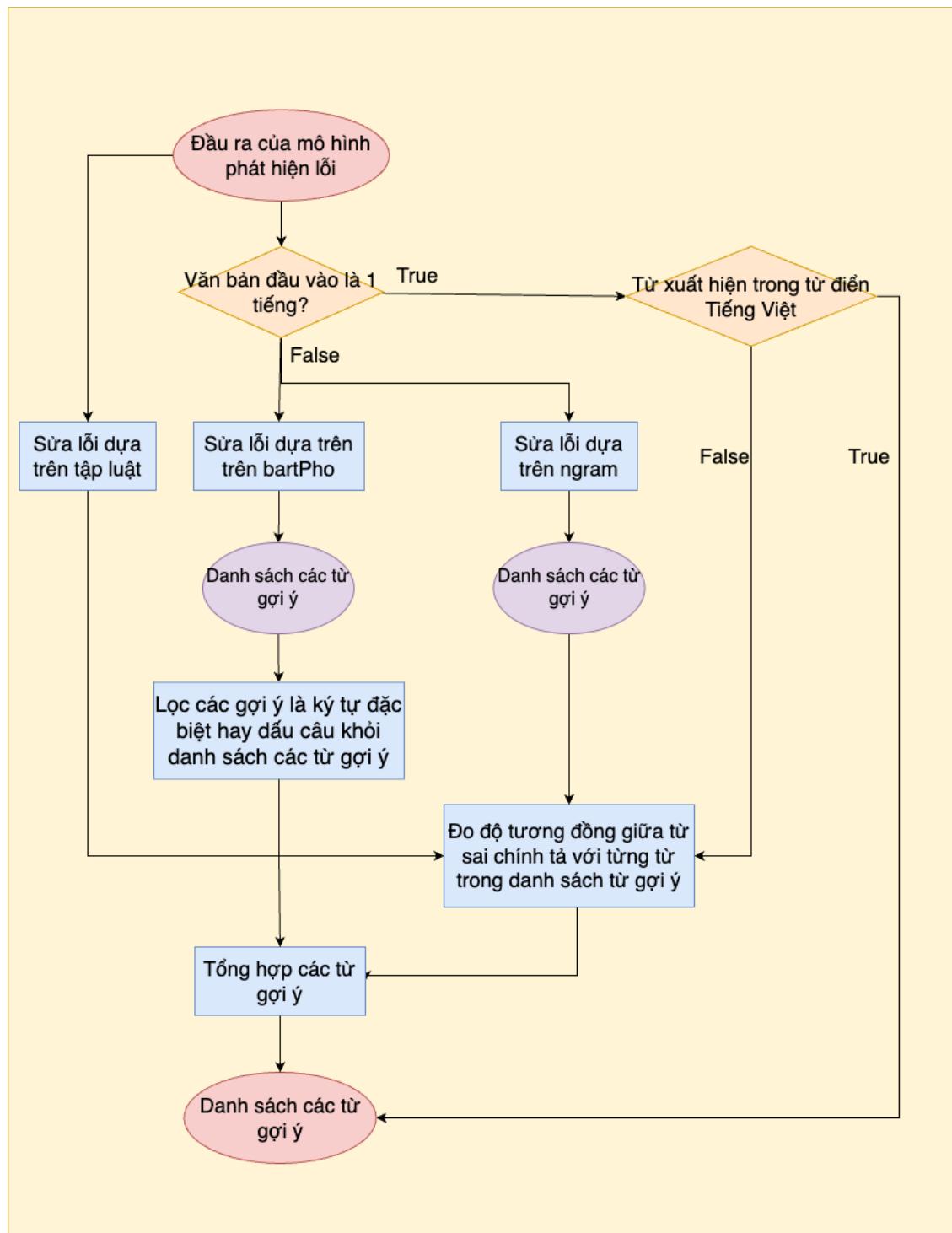
Model được xây dựng dựa trên 2 phần chính là Word-level Transformer Encoder và Charactor-level Transformer Encoder

Từ 1 câu input đầu vào được đi qua tokenizer để tách câu thành các từ riêng biệt. Mỗi từ được phân tách để đi qua Charactor-level Transformer Encoder. Đầu ra của model sẽ được nối vào Word Embedding sau đó cộng với Position Embedding tương ứng với từng vị trí. Và cuối cùng sẽ đi qua Word-level Transformer để kiểm tra xem từ đó có sai chính tả hay không.

Tập vocab được xây dựng dựa trên tần suất xuất hiện của mỗi từ được sắp xếp từ cao xuống thấp. File vocab word em cố định là 10000 từ, file sub vocab của em là 106 ký tự. Những từ không tồn tại ở file vocab sẽ được đánh dấu là <UNK> Embedding.

3.3 Mô đun gợi ý sửa lỗi

1. Mô hình đề xuất

**Hình 3.3:** Mô đun gợi ý sửa lỗi

Từ một câu input đầu vào, đầu tiên em sẽ lấy ra các từ sai và cho đi qua rule. Sau đó em sẽ kiểm tra câu đó được cấu tạo từ 1 từ đơn (1 tiếng) hay nhiều tiếng trở lên.

Nếu câu được cấu tạo từ 1 từ đơn thì kiểm tra từ đó có thuộc trong từ điển hay không. Nếu từ đó không ở trong từ điển thì so sánh độ tương đồng giữa từ cần kiểm tra với từ điển và đưa ra những từ gần với từ cần kiểm tra nhất. Nếu từ

đó có xuất hiện trong từ điển thì trả luôn về output.

Nếu câu được cấu tạo từ 2 từ đơn trở lên thì em sẽ đưa câu lần lượt qua 2 mô hình là N-gram, BartPho.

Sau khi đưa qua BartPho em sẽ thu được danh sách các từ gợi ý và đưa danh sách đó qua phần lọc kết quả. Tại đây, sẽ lọc khỏi danh sách các ký tự đặc biệt hay dấu câu.

Sau khi đưa qua N-gram em thu được danh sách các từ gợi ý. Với mỗi từ trong danh sách em sẽ đo độ tương đồng (3.3.4) với từ sai chính tả cần sửa. Từ kết quả đo độ tương đồng, em sắp xếp lại danh sách các từ gợi ý và lấy ra những từ gần với từ cần sửa nhất.

Cuối cùng, em tổng hợp lại từ gợi ý từ tập luật, danh sách từ gợi ý từ BartPho, danh sách từ gợi ý từ N-gram và danh sách từ gợi ý từ từ điển.

3.3.1 Mô hình N-gram sử dụng trong sửa lỗi chính tả

N-gram nhận input đầu vào là 1 cụm từ. Và đưa ra gợi ý cho vị trí tiếp theo.

Ở N-gram, em sử dụng kết hợp bigram và trigram.

- Trigram: Lấy hai từ phía trước của từ bị sai rồi tìm trong mô hình N-gram từ có xác suất cao nhất xuất hiện phía sau.

Ví dụ: Hệ thống kiểm tra lỗi chính **tar**.

Đầu vào của trigram: lỗi chính

Đầu ra của trigram là một mảng các từ gợi ý được sắp xếp theo xác suất giảm dần: ['tả', 'là', 'thức', 'em', 'bản', 'xác', 'vì', 'họ', 'trái', 'phụ', 'những', 'dẫn', 'quyền', 'tôi', 'vẫn', 'ở', 'mình', 'nằm', 'thuộc', 'gia', 'của', 'cho', 'và', 'như']

- Bigram: Lấy một từ phía trước của từ bị sai rồi tìm trong mô hình N-gram từ có xác suất cao nhất xuất hiện phía sau.

Ví dụ: Hệ thống kiểm **cha** lỗi chính tả.

Đầu vào của bigram: kiểm

Đầu ra của bigram: ['tra', 'soát', 'nghiêm', 'chứng', 'dịch', 'định', 'điểm', 'duyệt', 'thảo', 'sát']

3.3.2 Mô hình huấn luyện trước BartPho

BartPho nhận input đầu vào là 1 câu đã được đánh dấu các vị trí sai. Dựa vào ngữ cảnh của câu BartPho sẽ đưa ra gợi ý theo từng vị trí.

```
[ ] import torch
from transformers import AutoModel, AutoTokenizer, XLMRobertaTokenizer
syllable_tokenizer = AutoTokenizer.from_pretrained("vinai/bartpho-syllable", use_fast=False)
from transformers import MBartForConditionalGeneration
bartpho_syllable = MBartForConditionalGeneration.from_pretrained("vinai/bartpho-syllable")
```

Hình 3.4: Sử dụng mô hình BartPho

Ví dụ: Hệ thống kiểm **cha** lỗi chính tả.

Đầu vào: Hệ thống kiểm <mask> lỗi chính tả.

Đầu ra: ['tra', 'soát', 'thủ', 'lỗi', 'duyệt']

3.3.3 Tập luật sử dụng cho bài toán sửa lỗi chính tả

Ở bài toán này em đề xuất 7 dạng luật sửa 7 dạng lỗi:

- Thêm phụ âm còn thiếu trong từ

Ví dụ: sih - sinh, báh - bánh, cah - canh

- Xóa các ký tự lặp lại

Ví dụ: biinh -> binh, bangg -> bang

- Đổi chỗ ký tự bị sai thứ tự

Ví dụ: nếu gặp các cặp ký từ 'hn','hc','ig','hg','hk','gn','ht','rt','uq','hp','ei','ey','ou','or', thì đổi chỗ lại cho nhau

- Xử lý các lỗi với y và i

Ví dụ: các vẫn chứa i nhưng không tồn tại với y vd: uy => ui, ym=>im và ngược lại

- Thiếu dấu phụ

Ví dụ: hươu -> hươu, yeu -> yêu, tẫu -> tẫu

- Sai vị trí dấu trong từ, sai bộ gõ

Ví dụ: hoà thuận -> hòa thuận, hiếu thảo -> hiếu thảo

- Các lỗi chính tả với ngh,ng,g,gh,... (đứng đầu câu):

Ví dụ: ngiêng -> nghiêng, gê -> ghê

- Viết rõ cho các ký tự hay được viết tắt

Ví dụ: fâi => phải

- Thay thế các từ teencode

Ví dụ: mk -> mình, ck -> chồng, vk -> vợ

3.3.4 Phương pháp đo độ tương đồng giữa hai từ

Phương pháp đo độ tương đồng là phương pháp để đo độ giống nhau giữa hai từ.

Với mỗi từ đầu vào, em chuyển từ đó thành mảng các ký tự bàn phím cấu tạo nên từ đó. Sau đó sẽ tính độ tương đồng giữa mảng ký tự của từ gợi ý với từ sai lỗi chính tả. Từ độ tương đồng tính được, em sắp xếp lại vị trí các từ gợi ý và đưa ra các từ gợi ý có độ tương đồng cao nhất.

Ví dụ chuyển một từ thành các ký tự từ bàn phím cấu tạo nên từ đó: kiểm -> kieemr, tra -> tra, lỗi -> looix, chính -> chinhhs, tả -> tar.

Độ tương đồng của hai từ được tính theo công thức sau:

$$\text{similarity} = \frac{N_{sameOfCharacters}}{N_{totalOfCharacters}} \quad (3.1)$$

Trong đó:

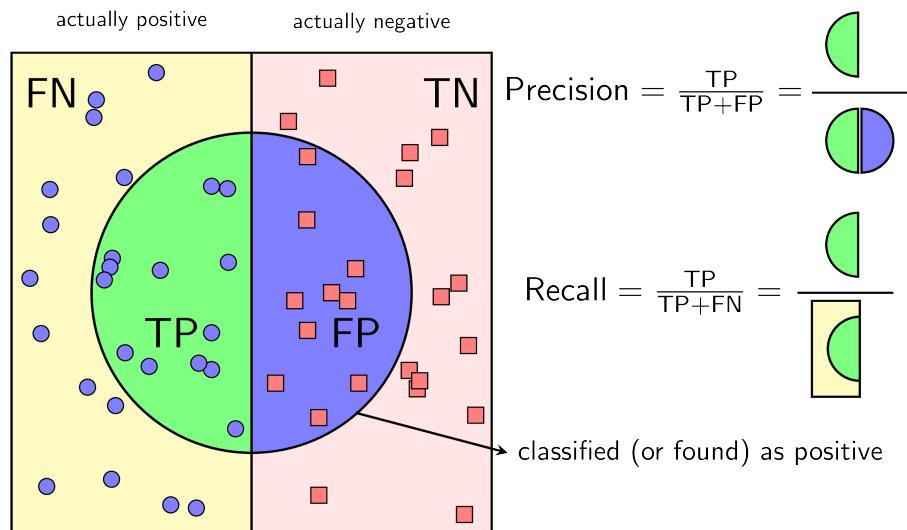
- similarity: Độ tương đồng
- $N_{sameOfCharacters}$: Số ký tự khớp nhau
- $N_{totalOfCharacters}$: Tổng số ký tự

CHƯƠNG 4. ĐÁNH GIÁ THỰC NGHIỆM

Chương này là thực nghiệm cho các đề xuất ở Chương 3 nên trong chương này sẽ trình bày về các tham số đánh giá kết quả, phương pháp thực hiện thí nghiệm và các kết quả đạt được.

4.1 Các tham số đánh giá

Đối với bài toán này em sử dụng cách đánh giá bằng Recall.



Hình 4.1: Công thức Precision và Recall

Recall: được định nghĩa là tỉ lệ số điểm true positive trong số những điểm thực sự là positive ($TP + FN$).

4.2 Phương pháp thí nghiệm

Em tiến hành thí nghiệm theo quy trình như sau: (i) Thực hiện chuẩn bị dữ liệu, (ii) Chuẩn bị môi trường, (iii) Tiền xử lý dữ liệu, (iv) Huấn luyện mô hình Transformer cho mô đun phát hiện lỗi sai, (v) Huấn luyện mô hình N-gram cho mô đun gợi ý sửa lỗi.

4.2.1 Thực hiện chuẩn bị dữ liệu

a, Dữ liệu thô

Em sử dụng hai bộ dữ liệu chính là dữ liệu phoBert và dữ liệu là các file đồ án, báo cáo khoa học.

- Dữ liệu PhoBERT [6]: Trích xuất từ khoảng 14.896.998 bài báo trên internet bao gồm các báo: Tuyên Giáo, Tuổi Trẻ TĐ, Tài Chính, Tạp chí Công thương, Tạp chí Xây dựng Đảng, Tạp chí cộng sản, Tổ Quốc,... Vì bộ dữ liệu PhoBERT rất lớn (sấp xỉ 29GB) nên em chỉ sử dụng 4000000 câu trong toàn bộ dữ liệu

PhoBert được chia theo tỉ lệ 7:3 như sau: dữ liệu huấn luyện là 2800000 câu và dữ liệu kiểm thử là 1200000 câu.

- Dữ liệu đồ án: Thu thập được 1981699 câu từ 1648 file đồ án được lấy trên hệ thống Coopy. Được chia ra thành 2 bộ dữ liệu huấn luyện và dữ liệu kiểm thử. Trong đó dữ liệu huấn luyện là 1387189 câu, dữ liệu kiểm thử là 594510 câu.

b, Dữ liệu thực nghiệm

Đối với dữ liệu thực nghiệm em tạo bộ dữ liệu bằng hai cách là tạo bằng hàm tạo lỗi và xây dựng một trang web để thu thập dữ liệu một cách thực tế. Chi tiết như sau:

- **Bằng hàm tạo lỗi:** Sử dụng dữ liệu phoBert cho đi qua hàm tạo lỗi để tạo thành các câu sai lỗi chính tả. Các lỗi được tạo ra từ hàm tạo lỗi theo các cách sau: tạo lỗi bằng teencode, tạo lỗi bằng cách đổi chỗ các ký tự, tạo lỗi bằng cách thay thế ngẫu nhiên bằng các ký tự lân cận, loại bỏ ký tự bất kỳ, thêm một ký tự bất kì vào, lỗi VNI (ví dụ: ă -> a8), thay thế bằng các từ tương đồng để nhầm lẫn (ví dụ: i với y, ch với tr,...), tạo các cặp từ dễ nhầm lẫn do vùng miền hay do cách phát âm (ví dụ: sē -> sέ,...).
- **Bằng dữ liệu thực tế:** Xây dựng trang web để thu thập dữ liệu thực tế. Vì lượng dữ liệu có hạn nên phần dữ liệu này em chủ yếu sử dụng cho bộ dữ liệu kiểm thử.

Đường dẫn trang web thu thập dữ liệu thực tế: <https://web-dataset.herokuapp.com/>

4.2.2 Chuẩn bị môi trường

Mô hình được xây dựng và thực nghiệm trên Google Colab có cấu hình như sau:

- Hệ điều hành: Ubuntu 18.04.5.
- GPU Tesla T4.
- CUDA.
- RAM: 25GB.

4.2.3 Tiền xử lý dữ liệu

Em sẽ thay đổi cấu trúc của dữ liệu đầu vào, từ đó giúp mô hình làm việc dễ dàng hơn trong các ngữ cảnh thực tế.

Quá trình tiền xử lý như sau:

- Chuyển dữ liệu đầu vào thành chữ thường.
- Xoá dấu câu và các ký tự không phải chữ cái nhưng giữ nguyên index của từ.

- Đánh lại dấu cho từ nếu chưa phù hợp. Ví dụ xõa -> xoã.
- Loại bỏ ký tự ở đầu mỗi câu ở danh sách. Ví dụ a), b),...
- Loại bỏ các ký tự khoảng trắng không cần thiết ở đầu câu, giữa câu và cuối câu.

4.2.4 Huấn luyện mô hình Transformer cho mô đun phát hiện lỗi chính tả

Sau khi đã chuẩn bị dữ liệu và tiền xử lý dữ liệu xong, em tiến hành huấn luyện mô hình Transformer trên tập dữ liệu đã chuẩn bị. Trước khi huấn luyện em tiến hành cài đặt và sử dụng một số thư viện cho quá trình huấn luyện như:

- torch: pip install torch

Mô hình được huấn luyện trên bộ dữ liệu là 4 lần. Theo như thiết lập cho mô hình thì mỗi lần chạy tối đa số epoch là 20, nhưng mô hình đạt giá trị sớm nên sau khi huấn luyện được từ 5 epoch đến 6 epoch thì mô hình đã dừng lại.

4.2.5 Huấn luyện mô hình với N-gram cho mô đun gợi ý sửa lỗi

Từ cơ sở lý thuyết và phần đề xuất ở mục 3.3.1 của mô hình N-gram thì em đã triển khai và thực nghiệm 2 mô hình bigram và trigram theo luồng sau:

Đầu tiên, đối với dữ liệu đầu vào thì em nối dữ liệu thành một chuỗi sau đó tiền xử lý dữ liệu và chuyển thành một mảng các token.

Tiếp theo, em sử dụng bigram và trigram trong thư viện nltk để huấn luyện tạo ra hai mô hình bigram và trigram.

4.3 Kết quả của mô hình Transformer cho mô đun phát hiện lỗi chính tả

Đối với bài toán này của em thì độ chính xác sẽ là tỉ lệ số từ sai chính tả do mô hình dự đoán trên tổng số từ thật sự sai chính tả. Có nghĩa là 1 câu mà có 100 từ, trong đó có 10 từ sai, mô hình đoán đúng được vị trí của 7 từ sai thì độ chính xác là $7/10 = 70\%$.

Đối với mô hình Transformer cho mô đun phát hiện lỗi chính tả trên bộ dữ liệu huấn luyện có kết quả theo cách đánh giá recall là 97.30%.

```
Save model with acc = 97.30 %
save model parameters to [model5.bin]
```

Hình 4.2: Độ chính xác của mô hình transformer

Từ hình ảnh trên em thấy mô hình có độ chính xác khá cao.

4.4 Kết quả của mô đun sửa lỗi chính tả

Đối với mô đun sửa lỗi chính tả em kết hợp giữa ba phương pháp là mô hình N-gram, mô hình huấn luyện trước BartPho và tập luật.

Đối với bài toán này của em thì độ chính xác sẽ là tỉ lệ số từ do mô hình gợi ý đúng trên tổng số từ thật sự sai chính tả. Có nghĩa là 1 câu mà có 100 từ, trong đó có 10 từ sai, mô hình gợi ý đúng được vị trí của 7 từ sai thì độ chính xác là $7/10 = 70\%$.

- Sửa lỗi bằng n-gram

Correct spelling with ngram: 40.32258064516129

Hình 4.3: Độ chính xác của mô hình N-gram cho bài toán gợi ý sửa lỗi chính tả

- Sửa lỗi bằng BartPho

Correct spelling with bartpho: 35.483870967741936

Hình 4.4: Độ chính xác của mô hình huấn luyện trước BartPho cho bài toán gợi ý sửa lỗi chính tả

- Sửa lỗi bằng tập luật

Correct spelling with rule: 79.03225806451613

Hình 4.5: Độ chính xác của phương pháp sử dụng tập luật cho bài toán gợi ý sửa lỗi chính tả

- Sửa lỗi kết hợp giữa 3 phương pháp là mô hình N-gram, BartPho và tập luật cho bài toán gợi ý sửa lỗi chính tả

Correct spelling: 93.54838709677419

Hình 4.6: Độ chính xác sau khi kết hợp ba phương pháp là mô hình n-gram, mô hình huấn luyện trước BartPho và tập luật

Từ các hình ảnh trên em thấy độ chính xác của mỗi mô hình n-gram (40.32%) (**Hình 4.3**), BartPho (35.48%) (**Hình 4.4**) và phương pháp tập luật (79.03%) (**Hình**

4.5) đều chưa cao đều nhỏ hơn 80%. Nhưng khi kết hợp 3 phương pháp lại thì độ chính xác của cho bài toán gọi ý sửa lỗi chính tả khá cao (93.55%) (**Hình 4.6**).

Kết chương

Chương này đã đưa ra các phương pháp đánh giá cùng với các kết quả mà ĐATN đạt được. Trong chương tiếp theo em sẽ trình bày chi tiết về hệ thống phát hiện và sửa lỗi chính tả - Chương 5.

CHƯƠNG 5. PHÁT TRIỂN HỆ THỐNG TRANG WEB THU THẬP VÀ ĐÁNH GIÁ SẢN PHẨM

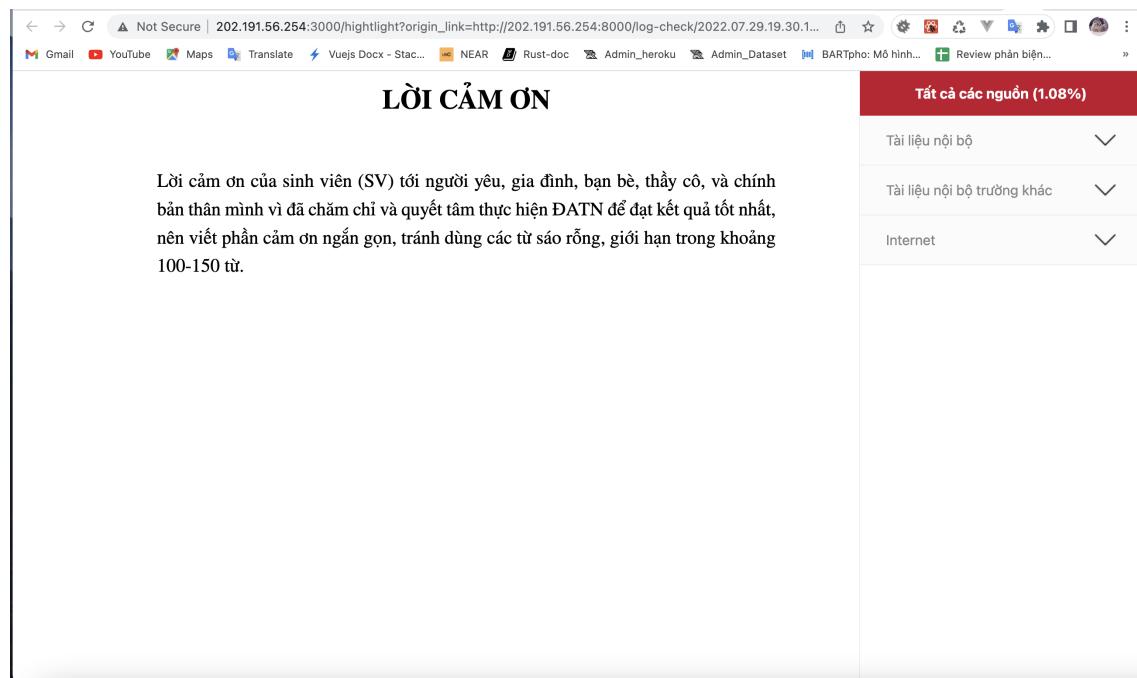
Chương 5 tập trung trình bày về hai vấn đề: (i) Cải thiện giao diện đạo văn của hệ thống Coopy và (ii) Kiến trúc tổng quan của hệ thống phát hiện và sửa lỗi sai chính tả và những nội dung liên quan đến xây dựng, triển khai hệ thống bao gồm: chức năng kiểm tra file báo cáo, chức năng sửa lỗi chính tả, chức năng tải file sau khi sửa lỗi.

5.1 Cải thiện giao diện đạo văn của hệ thống Coopy

Đầu tiên là em xin trình bày về việc cải thiện giao diện hệ thống kiểm tra đạo văn của trường Công Nghệ Thông Tin và Truyền Thông.

5.1.1 Khảo sát giao diện hiện tại

Trước tiên, em sẽ đi khảo sát giao diện của hệ thống kiểm tra đạo văn. **Hình 5.1** là giao diện hiện tại của hệ thống kiểm tra đạo văn.



Hình 5.1: Giao diện hiện tại của hệ thống Coopy

Giao diện phần hiển thị file pdf sau khi kiểm tra đạo văn trong trang web Coopy trước đây có không có phần header và không có tên file nên người dùng khó nắm bắt được là mình đang kiểm tra file nào. Ở sidebar, trong trường hợp danh sách trùng lặp bị dài thì sẽ khó theo dõi và tốn thời gian thao tác do phải lướt lên xuống nhiều hơn. Trong sidebar thì chỉ có hiển thị tổng quan về các tài liệu, chưa có tính năng để tải file sau khi kiểm tra về trên sidebar và đặc biệt chưa có tính năng xoá

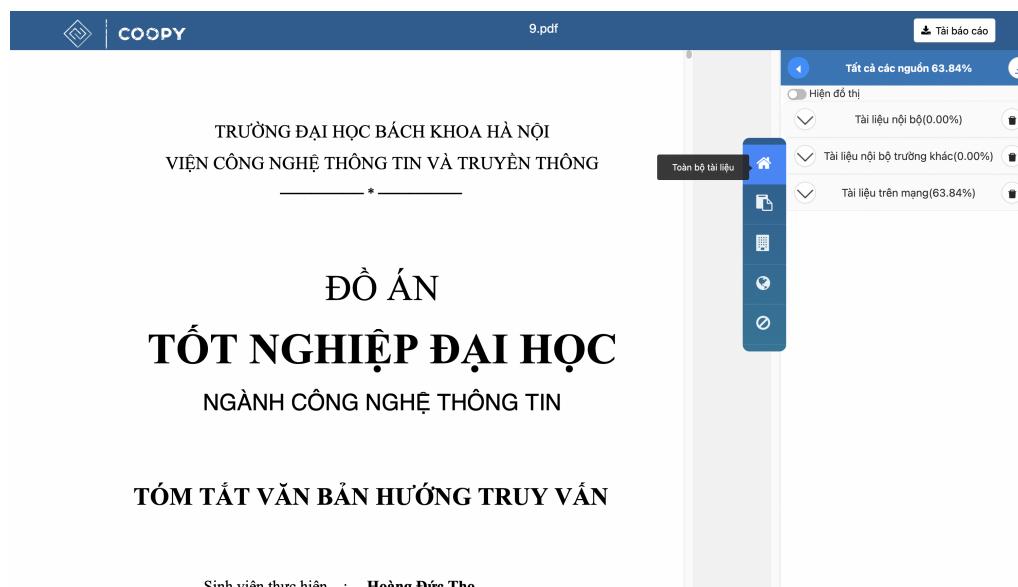
các file không cần kiểm tra ra khỏi những tài liệu cần kiểm tra. Ngoài ra, ở sidebar cũng không có icon nêu độ nhận dạng của người dùng sẽ kém đi.

5.1.2 Cải thiện giao diện

Từ khảo sát hệ thống ở mục trên, em đã cải thiện như sau:

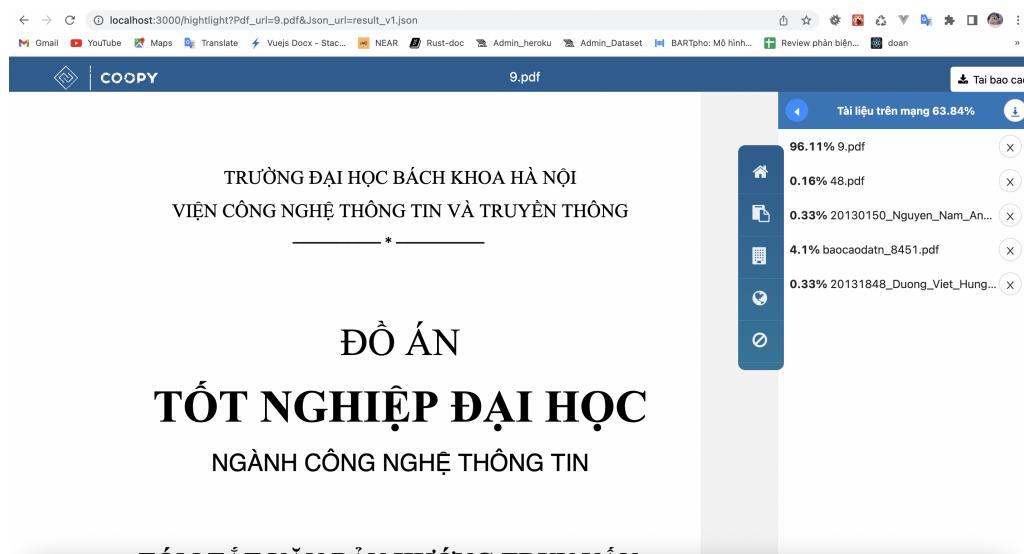
a, Cải thiện giao diện chính

Đối với giao diện khảo sát hiện tại em đã thay đổi màu từ đỏ sang xanh, cùng với việc thêm menu chứa các icon dễ hiểu và có gợi ý khi trỏ chuột vào sẽ hiện lên như "Toàn bộ tài liệu". Trên header có thêm tên của tài liệu, logo của hệ thống, ngoài ra có cả nút tải file xuống máy. Ta có thể thấy rõ theo **Hình 5.2** bên dưới:



Hình 5.2: Giao diện chính sau khi cải thiện

Ngoài ra em có chỉnh lại cho phần sidebar khi không cần dùng tới sẽ ẩn đi và có phân rõ các tài liệu ra theo menu đã thiết kế như "Tài liệu nội bộ", "Tài liệu trên mạng", "Tài liệu trường khác". Ví dụ như trong **Hình 5.3** bên dưới là tài liệu trên mạng:

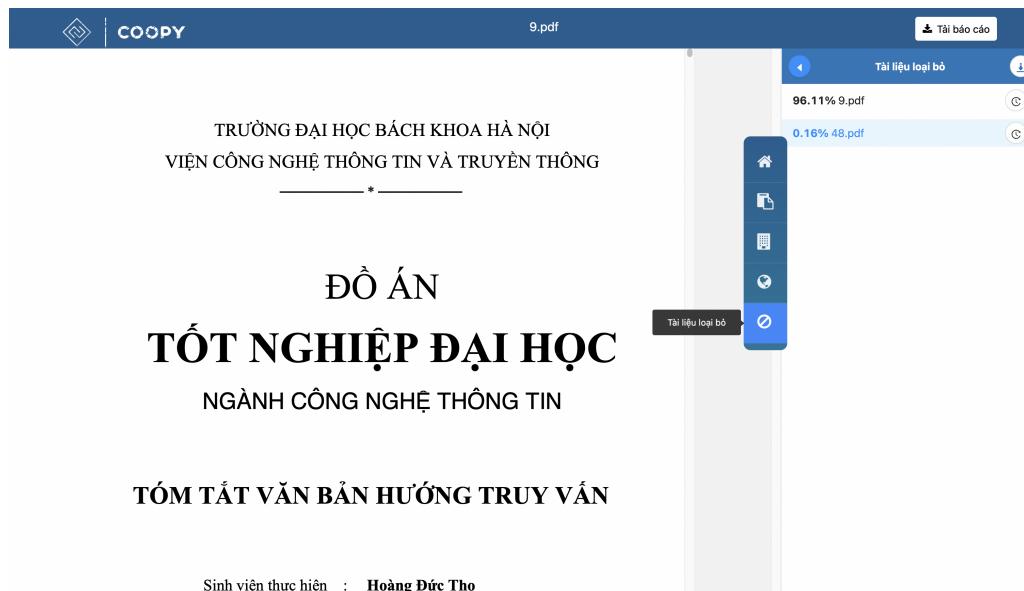


Hình 5.3: Giao diện mục tài liệu trên mạng

b, Loại bỏ tài liệu không cần kiểm tra

Ngoài ra trên mỗi tài liệu em có thiết kế thêm 1 nút xoá tài liệu với icon thùng rác nhằm loại bỏ tài liệu không cần kiểm tra trùng chỉ giáo viên mới có thể loại bỏ.

Sau khi nhấn nút bỏ tài liệu thì tài liệu đó sẽ được chuyển sang mục "Tài liệu loại bỏ" trên sidebar. Như **Hình 5.4** bên dưới:

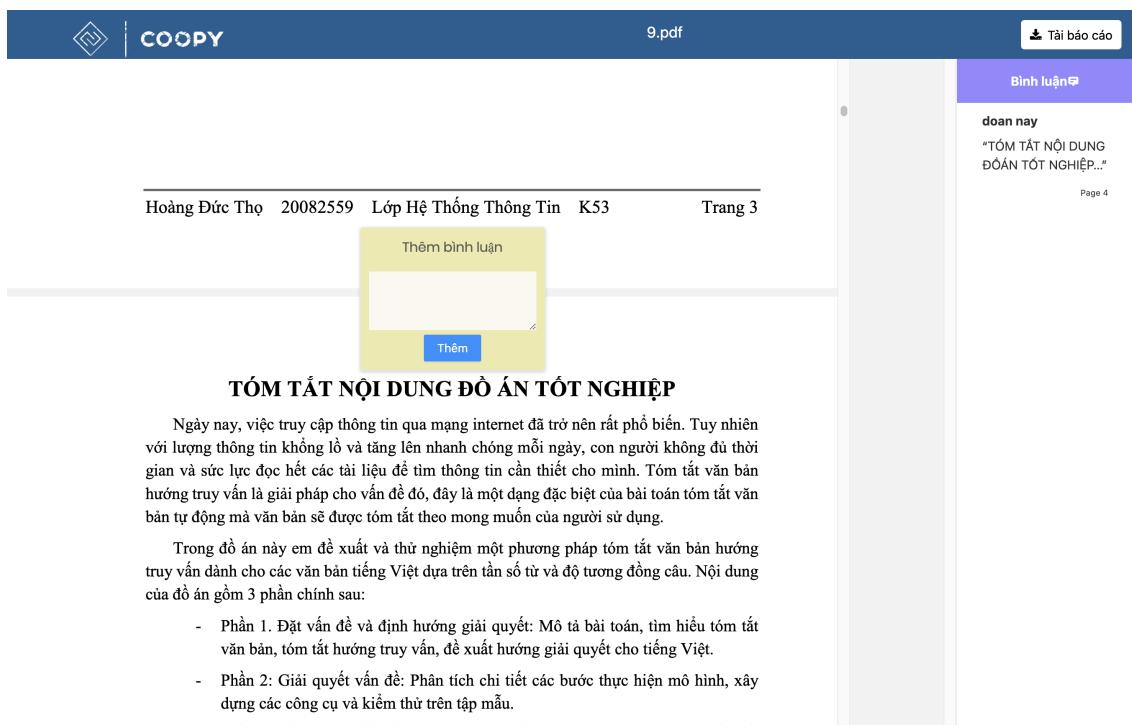


Hình 5.4: Giao diện mục tài liệu loại bỏ

Sau khi loại bỏ tài liệu mà người dùng muốn khôi phục lại thì có thể nhấn vào nút có biểu tượng đồng hồ ở cuối tài liệu.

c, Bình luận tài liệu

Do vấn đề giáo viên mỗi khi muốn góp ý cho sinh viên thì cần phải liên hệ với sinh viên để góp ý gây mất thời gian và bất tiện nên em có thiết kế chức năng giúp cho giáo viên có thể góp ý, nhận xét trực tiếp trên hệ thống. Chức năng và thiết kế được mô tả qua **Hình 5.5** bên dưới:



Hình 5.5: Giao diện sau khi cải thiện của hệ thống Coopy

Khi góp ý xong các góp ý sẽ được thêm vào sidebar. Khi nhấp vào góp ý sẽ nhảy tới vị trí mà giáo viên đã góp ý.

5.2 Thiết kế và triển khai hệ thống tích hợp mô đun kiểm tra và sửa lỗi chính tả với hệ thống Coopy

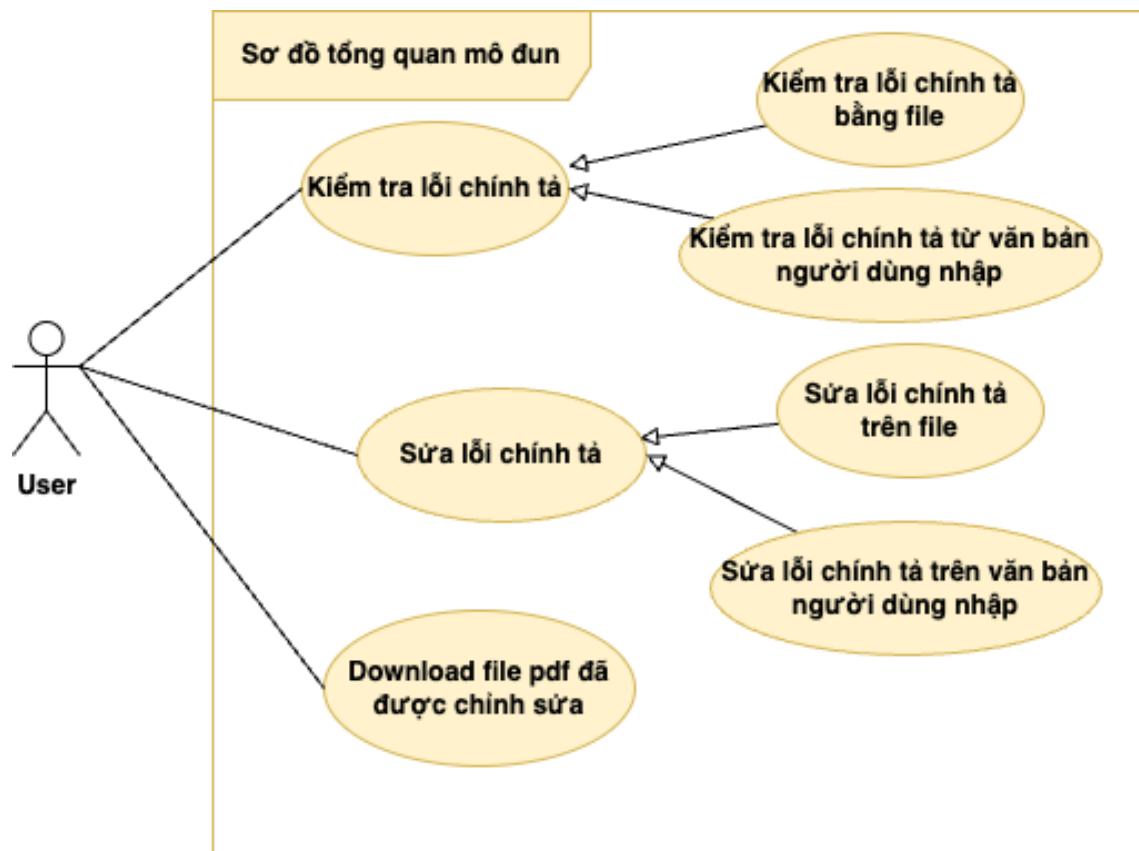
Trong phần này, em sẽ trình bày về quá trình phát triển hệ thống kiểm tra và sửa lỗi chính tả từ thiết kế tới triển khai hệ thống.

5.2.1 Phân tích yêu cầu

Hệ thống có 1 tác nhân là người dùng. Người dùng ở đây có thể là sinh viên hay người viết báo cáo khoa học. Người dùng có thể truy cập vào hệ thống để kiểm tra lỗi chính tả bằng hai phương án: nhập văn bản từ bàn phím hoặc tải lên một file văn bản. Sau khi kiểm tra lỗi chính tả thì người dùng có thể lựa chọn các gợi ý tương ứng mà hệ thống đưa ra để sửa lỗi đó. Đối với phần kiểm tra lỗi chính tả bằng file người dùng có thể tải file mới sau khi đã lựa chọn gợi ý chỉnh sửa phù hợp.

5.2.2 Tổng quan chức năng

Từ phân tích yêu cầu của hệ thống bên trong thì trong phần này, em sẽ thiết kế tổng quan sơ đồ use case cho hệ thống. Sơ đồ được thể hiện trong **Hình 5.7**.



Hình 5.6: Biểu đồ use case tổng quan của hệ thống

5.2.3 Đặc tả chức năng

Phần này sẽ đặc tả một số chức năng trong hệ thống phát hiện và gợi ý sửa lỗi chính tả. **Bảng 5.1** liệt kê số use case được xây dựng.

Mã use case	Tên use case
UC001	Kiểm tra lỗi chính tả bằng file
UC002	Kiểm tra lỗi chính tả bằng văn bản người dùng nhập
UC003	Sửa lỗi chính tả trên file
UC004	Sửa lỗi chính tả trên văn bản người dùng nhập
UC005	Tải file đã được chỉnh sửa

Bảng 5.1: Danh sách use case

a, Đặc tả use case UC001 kiểm tra lỗi chính tả bằng file

Đặc tả use case UC001 được mô tả trong **Bảng 5.2**. Đây là chức năng giúp Người dùng có thể kiểm tra lỗi chính tả trên một file văn bản như báo cáo khoa học, báo cáo tốt nghiệp.

Mã use case	UC001	Tên use case	Kiểm tra lỗi chính tả bằng file
Tác nhân	Người dùng		
Tiền điều kiện	Không		
Luồng xử lý chính	STT	Thực hiện bởi	Hành động
	1	Người dùng	Nhấn nút có biểu tượng "SELECT A FILE" để tải file lên
	2	Người dùng	Chọn file để tải lên
	3	Hệ thống	Nhận file từ người dùng và xử lý
Luồng xử lý thay thế	STT	Thực hiện bởi	Hành động
Hậu điều kiện	Không		

Bảng 5.2: Đặc tả use case "Kiểm tra lỗi chính tả bằng file"

b, Đặc tả use case UC002 kiểm tra lỗi chính tả bằng câu văn

Đặc tả use case UC002 được mô tả trong **Bảng 5.3**. Đây là chức năng giúp người dùng kiểm tra lỗi văn bản bằng cách nhập một đoạn văn bản. Đây là chức năng hữu ích khi nhu cầu của người dùng chỉ là kiểm tra xem câu văn của họ có đúng chính tả không.

Mã use case	UC002	Tên use case	Kiểm tra lỗi chính tả bằng văn bản người dùng nhập
Tác nhân	Người dùng		
Tiền điều kiện	Khách sau khi truy cập vào hệ thống		
Luồng xử lý chính	STT	Thực hiện bởi	Hành động
	1	Người dùng	Nhập văn bản vào ô "Input text"
	2	Người dùng	Nhấn nút "Submit"
	3	Hệ thống	Nhận văn bản từ người dùng để xử lý
Luồng xử lý thay thế	STT	Thực hiện bởi	Hành động
	Hậu điều kiện		
	Không		

Bảng 5.3: Đặc tả use case "Kiểm tra lỗi chính tả bằng văn bản người dùng nhập"

c, Đặc tả use case UC003 sửa lỗi chính tả trên file

Đặc tả use case UC003 được mô tả trong **Bảng 5.4**. Đây là chức năng giúp Người dùng có thể sửa trực tiếp những lỗi sai trên file báo cáo.

Mã use case	UC003	Tên use case	Sửa lỗi chính tả trên file
Tác nhân	Người dùng		
Tiền điều kiện	Đã có file kiểm tra xong chính tả		
Luồng xử lý chính	STT	Thực hiện bởi	Hành động
	1	Người dùng	Di chuột đến vị trí highlight
	2	Hệ thống	Hiển thị những gợi ý sửa lỗi cho vị trí highlight mà người dùng di chuột đến
	3	Người dùng	Lựa chọn từ gợi ý phù hợp
Luồng xử lý thay thế	STT	Thực hiện bởi	Hành động
	Hậu điều kiện		
	Không		

Bảng 5.4: Đặc tả use case "Sửa lỗi chính tả trên file"

d, Đặc tả use case UC004 sửa lỗi chính tả trên văn bản người dùng nhập

Đặc tả use case UC004 được mô tả trong **Bảng 5.5**. Đây là chức năng giúp người dùng có thể sửa trực tiếp những lỗi chính tả của văn bản do người dùng nhập từ bàn phím.

Mã use case	UC003	Tên use case	Sửa lỗi chính tả trên văn bản người dùng nhập
Tác nhân	Người dùng		
Tiền điều kiện	Câu văn đã được kiểm tra chính tả		
Luồng xử lý chính	STT	Thực hiện bởi	Hành động
	1	Người dùng	Di chuột đến vị trí highlight
	2	Hệ thống	Hiển thị những gợi ý sửa lỗi cho vị trí highlight mà người dùng di chuột đến
	3	Người dùng	Lựa chọn từ gợi ý phù hợp
	4	Hệ thống	Thay thế từ đã được người lựa chọn cho vị trí sai chính tả được highlight
Luồng xử lý thay thế	STT	Thực hiện bởi	Hành động
Hậu điều kiện	Không		

Bảng 5.5: Đặc tả use case "Sửa lỗi chính tả trên văn bản người dùng nhập"

e, Đặc tả use case UC005 tải file pdf đã được chỉnh sửa

Đặc tả use case UC005 được mô tả trong **Bảng 5.6**. Đây là chức năng giúp Người dùng có thể tải file báo cáo sau khi sửa lỗi chính tả.

Mã use case	UC005	Tên use case	Tải file pdf đã được chỉnh sửa
Tác nhân	Người dùng		
Tiền điều kiện	Tồn tại file đã được chỉnh sửa lỗi chính tả		
Luồng xử lý chính	STT	Thực hiện bởi	Hành động
	1	Người dùng	Nhấp vào nút "Download"
	2	Hệ thống	Xử lý file và mở 1 tab mới để hiển thị file đã được sửa
Luồng xử lý thay thế	STT	Thực hiện bởi	Hành động
Hậu điều kiện	Không		

Bảng 5.6: Đặc tả use case "Tải file đã được chỉnh sửa"

5.2.4 Yêu cầu phi chức năng

1. Tính dễ dùng.

Do hệ thống hướng tới tất cả người dùng gồm cả những người có ít kinh nghiệm về máy tính và công nghệ nên hệ thống xây dựng giao diện một cách đơn giản, dễ dùng, đặc biệt là hạn chế các thao tác không cần thiết. Các ký hiệu sử dụng trong hệ thống là ký hiệu được sử dụng phổ biến, dễ hiểu, dễ dùng.

2. Tính dễ bảo trì.

Do hệ thống đang trong quá trình xây dựng và phát triển nên rất cần thiết kế để dễ dàng chỉnh sửa và nâng cấp theo yêu cầu từ phía người dùng.

Hệ thống cần thiết kế tốt để đảm bảo khi xây dựng tính năng mới sẽ không ảnh hưởng tới các tính năng ổn định hiện có.

3. Tính khả thi.

Đối với tính năng sửa lỗi chính tả trên file, hệ thống cần đảm bảo thời gian sửa lỗi cho văn bản nhập từ bàn phím từ 2 giây đến 10 giây, không để người dùng chờ đợi lâu dẫn tới trải nghiệm không tốt.

5.3 Công nghệ sử dụng

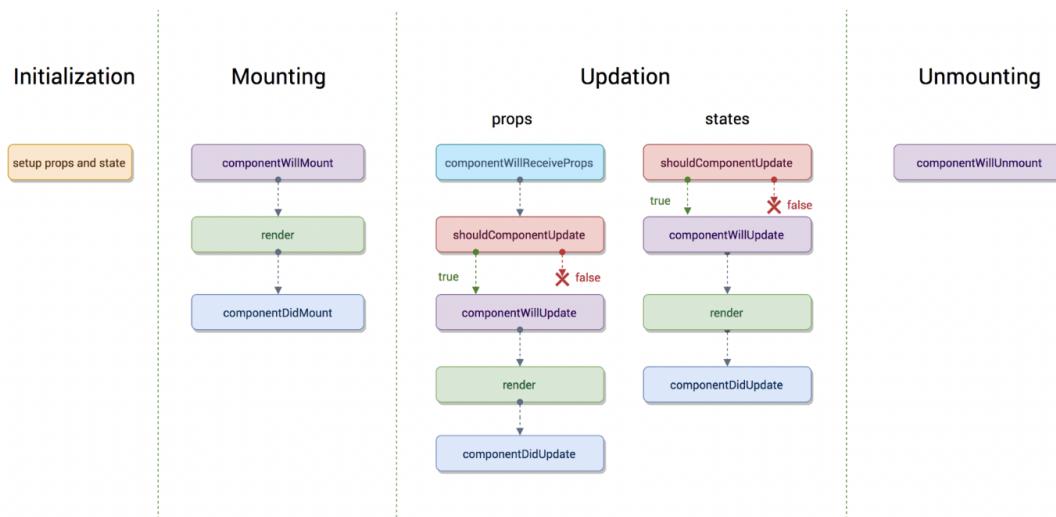
5.3.1 Frontend

a) ReactJS

ReactJS [7] là một thư viện JavaScript mã nguồn mở được thiết kế bởi Facebook để tạo ra những ứng dụng web Single Page Application hấp dẫn, nhanh và hiệu quả với mã hóa tối thiểu. Mục đích cốt lõi của ReactJS không chỉ khiến cho trang web trải nghiệm mượt mà mà còn phải nhanh, khả năng mở rộng cao và

đơn giản.

Sức mạnh của ReactJS xuất phát từ việc tập trung vào các thành phần riêng lẻ (component). Chính vì vậy, thay vì làm việc trên toàn bộ ứng dụng web, ReactJS cho phép một developer có thể chuyển đổi giao diện người dùng phức tạp thành các thành phần đơn giản hơn.



Hình 5.7: Vòng đời của ReactJS

Initialization: Đây là giai đoạn đầu tiên của một component, bắt đầu bằng cách khởi tạo state, props, các biến cần thiết hoặc bind các function thường. Điều này thực hiện bên trong phương thức constructor().

Mounting: Giai đoạn này được thực hiện sau khi khởi tạo xong. Đây là quá trình gắn React element (Virtual DOM) của một component vào Real DOM để thể hiện kết quả lên trình duyệt. Mounting có 3 phương thức là: componentWillMount, render, componentDidMount.

Updating: Đây là giai đoạn thứ 3 sau khi component đã render thành công lần đầu tiên. Trong giai đoạn này, dữ liệu của state và props sẽ được cập nhật để đáp ứng với các events theo yêu cầu của người dùng. Điều này dẫn đến việc re-render ở component. Tương ứng với nó ta có 4 phương thức chính: componentWillReceiveProps, shouldComponentUpdate, componentWillUpdate, componentDidUpdate

Unmounting: Đây là bước cuối cùng như kết thúc một của vòng đời trong một component. Khi tất cả các tác vụ hoàn thành và ta cần tiến hành unmount DOM (component bị xoá khỏi cây DOM). Giai đoạn này chỉ có 1 phương thức là componentWillUnmount.

Ưu điểm:

Tốc độ: Về cơ bản, React cho phép các nhà phát triển sử dụng các phần riêng lẻ của ứng dụng của họ ở cả phía máy khách và phía máy chủ, điều này cuối cùng giúp tăng tốc độ của quá trình phát triển. Nói một cách dễ hiểu, các nhà phát triển khác nhau có thể viết các phần riêng lẻ và tất cả các thay đổi được thực hiện sẽ không gây ra tính logic của ứng dụng.

Linh hoạt: So với các frontend framework khác, mã React dễ bảo trì hơn và linh hoạt hơn do cấu trúc mô-đun của nó. Do đó, sự linh hoạt này giúp tiết kiệm rất nhiều thời gian và chi phí cho doanh nghiệp.

Hiệu suất: React JS được thiết kế để cung cấp hiệu suất cao. Cốt lõi của khung cung cấp chương trình DOM ảo và kết xuất phía máy chủ, giúp các ứng dụng phức tạp chạy cực nhanh.

Khả năng sử dụng: Việc triển khai React khá dễ thực hiện nếu bạn có một số kiến thức cơ bản về JavaScript. Trên thực tế, một nhà phát triển JavaScript chuyên nghiệp có thể dễ dàng tìm hiểu tất cả các thông tin chi tiết của React framework chỉ trong một hoặc hai ngày.

Đặc trưng của ReactJS

Single-way data flow (Luồng dữ liệu một chiều): Những Framework sử dụng Virtual-DOM như ReactJS khi Virtual-DOM thay đổi, chúng ta không cần thao tác trực tiếp với DOM trên View mà vẫn thấy được sự thay đổi đó. Do Virtual-DOM ngoài đóng vai trò là Model, còn đóng vai trò là View nên mọi sự thay đổi trên Model đã kéo theo sự thay đổi trên View và ngược lại. Có nghĩa là mặc dù chúng ta không tác động trực tiếp vào các phần tử DOM ở View nhưng vẫn thực hiện được cơ chế Data-binding. Điều này làm cho tốc độ ứng dụng tăng lên đáng kể – một lợi thế khi sử dụng Virtual-DOM.

b) Ant Design

Ant Design cung cấp một đánh giá thực tế về các thiết kế đẹp dành cho cả các nhà thiết kế của Ant Design và những người đang sử dụng nó. Đồng thời, nó xây dựng một nền tảng trên cùng các nguyên tắc và mẫu thiết kế để đưa ra hướng dẫn và giải pháp chung cho mục tiêu thiết kế được chỉ định.

c) Tailwind css

Tailwind css là một utility-first CSS framework nó hỗ trợ phát triển xây dựng nhanh chóng giao diện người dùng, nó cũng có điểm chung giống như Bootstrap và điểm làm nó nổi bật hơn cả đó là chúng ta có thể tùy biến phát triển css theo cách mà chúng ta định nghĩa ra.

5.3.2 Backend

a) Flask

Flask là một web framework, nó là một Python module cho phép bạn phát triển các ứng dụng web một cách dễ dàng. Nó có tính mở rộng và là một microframework không bao gồm ORM (Object Relational Manager) hoặc các tính năng tương tự.

b) Firebase

Firebase là nền tảng do Google cung cấp, nhằm hỗ trợ việc tạo ra các web application, mobile application với chất lượng cao.

Với việc sử dụng Firebase, nhà phát triển có thể tập trung vào việc phát triển application mà không cần lo về việc sản phẩm của mình sẽ hoạt động và được quản lý thế nào ở phía Backend.

Trong dự án này, em sử dụng dịch vụ Firebase Cloud Storage để lưu trữ file PDF - file văn bản cần kiểm tra chính tả. Firebase Cloud Storage hỗ trợ việc quản lý, chia sẻ các content người dùng upload lên như ảnh, video; cũng như phục vụ các yêu cầu sử dụng những tài nguyên của ứng dụng.

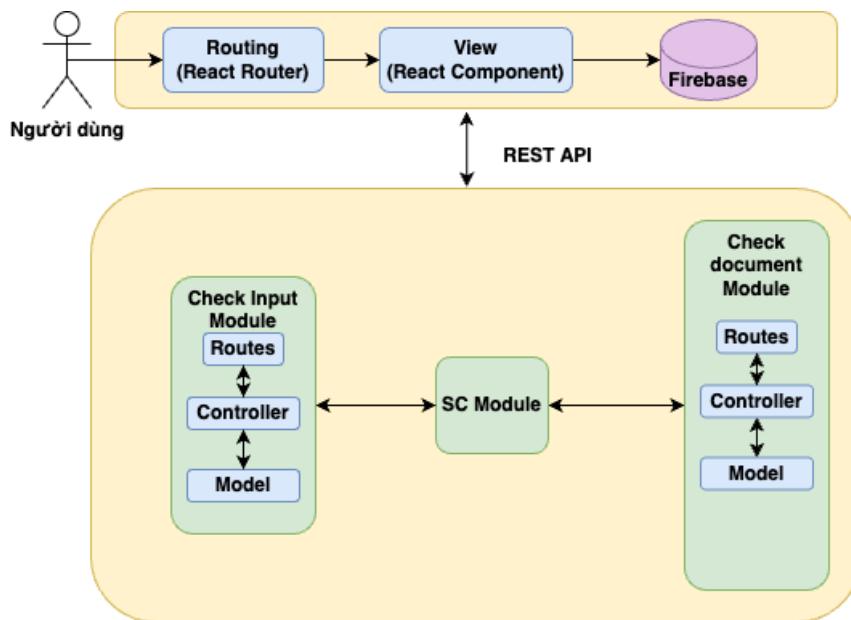
5.4 Thiết kế kiến trúc

Hình 5.8 mô tả kiến trúc chung của hệ thống. Bao gồm hai phần chính là Frontend và Backend.

Frontend sử dụng thư viện ReactJS nhằm xây dựng các thành phần giao diện nhận và gửi dữ liệu đến máy chủ thông qua API.

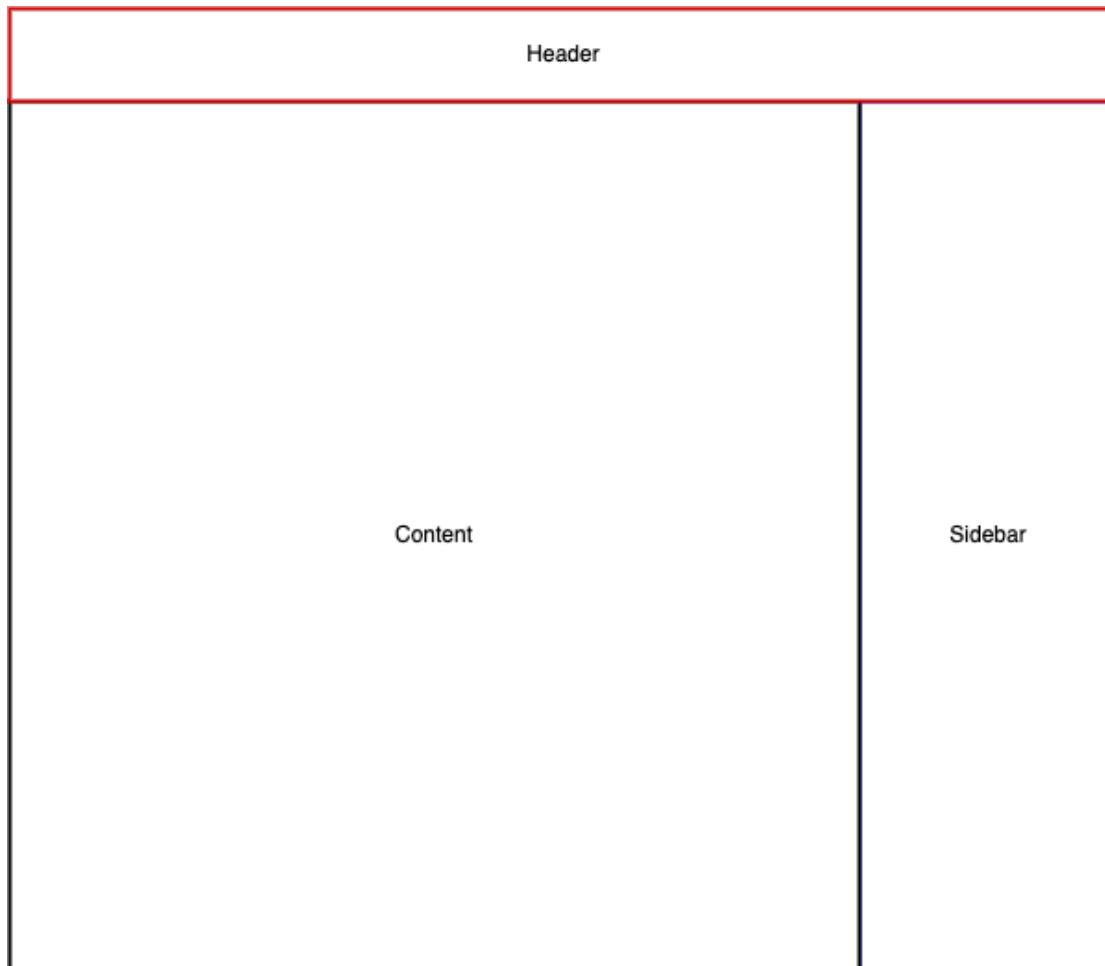
Backend được chia thành 3 mô đun: Mô đun kiểm tra và sửa lỗi của câu văn (Check Input Module), Mô đun kiểm tra và sửa lỗi chính tả (SC Module), Mô đun kiểm tra bằng file (Check Document Module).

Frontend sẽ kết nối tới các tính năng của Backend thông qua việc gọi API. Ngoài ra, việc giao tiếp giữa các mô đun cũng thông qua API của các mô đun. Nhờ thiết kế như vậy mà Frontend - Backend, các mô đun trong Backend được tách biệt và dễ dàng phát triển độc lập với nhau.

**Hình 5.8:** Kiến trúc chung của hệ thống

Chi tiết thiết kế các tính năng sẽ được trình bày trong phần 5.6.

5.5 Thiết kế chi tiết giao diện

**Hình 5.9:** Thiết kế mockup giao diện

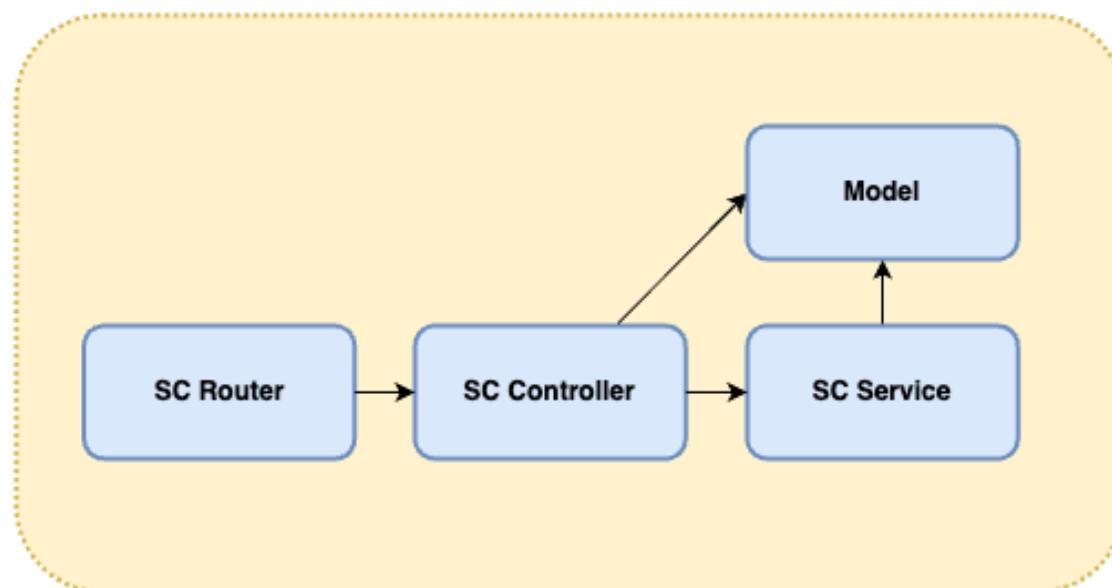
Hình 5.9 mô tả bô cục giao diện hệ thống bao gồm: Header, Sidebar, Content. Giao diện sử dụng thư viện ReactJS nên được thiết kế thành các thành phần (component).

Hệ thống được thiết kế đa phần sẽ đều có phần Header và Content chỉ thay đổi việc tồn tại hay không tồn tại Sidebar. Header là thanh tiêu đề của hệ thống gồm logo, menu chức năng. Content sẽ là phần được thay đổi. Sidebar là thanh bên phải của giao diện, chứa các từ sai và từ đã được lựa chọn để sửa.

5.6 Thiết kế chi tiết mô đun kiểm tra bằng file (Check document module)

Trong mô đun này sẽ có hai tính năng là: tính năng tải lên file để kiểm tra và tính năng tải file từ hệ thống xuống máy. Dưới đây là thiết kế và chi tiết về hai tính năng.

a, tính năng tải file lên để kiểm tra



Hình 5.10: Thiết kế tính năng tải file lên để kiểm tra chính tả

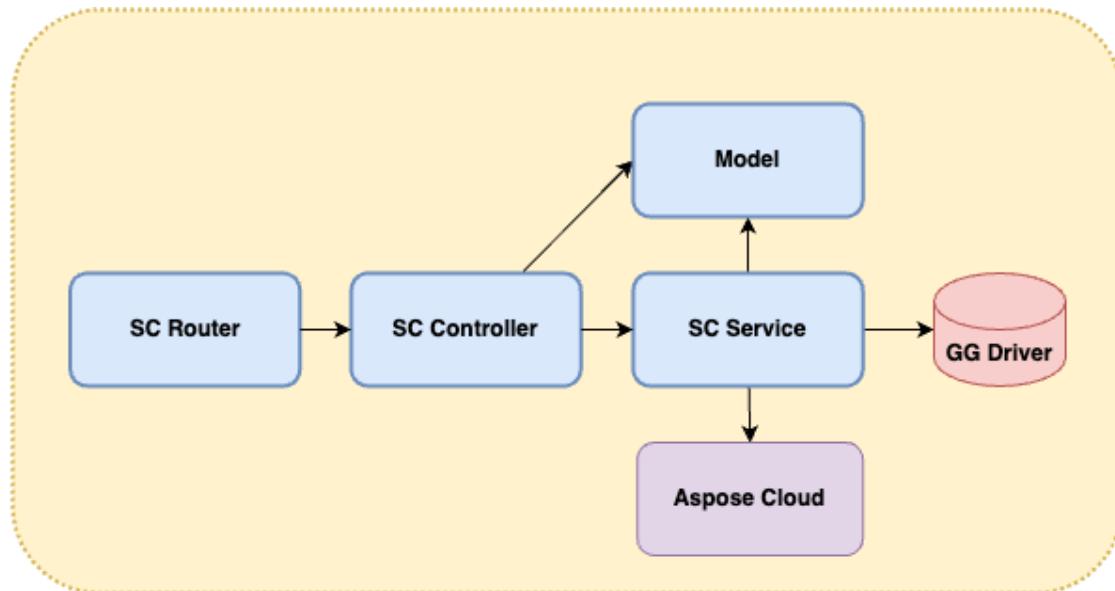
Hình 5.10 mô tả kiến trúc của tính năng tải file lên để kiểm tra lỗi chính tả. Phần tính năng này được xây dựng bằng thư viện Flask, được chia thành các phần: Routes, Controllers, Services, Models. Phần tính năng là nơi xử lý request từ client gửi đến.

Khi người dùng gửi yêu cầu (request) kiểm tra chính tả của file kèm với file cần kiểm tra đến server, các Router có nhiệm vụ định tuyến, gọi tới Controllers tương ứng với router nhằm xử lý việc kiểm tra lỗi chính tả của file. Các Controller tiếp nhận yêu cầu từ router tiếp theo là gọi đến Services để xử lý tác vụ kiểm tra lỗi chính tả. Cuối cùng service sẽ gọi tới mô đun kiểm tra và sửa lỗi chính tả.

Sau khi kết thúc quy trình trên thì máy chủ sẽ trả về cho người dùng một file đã

bôi vàng những vị trí của từ sai chính tả cùng với tập các từ có thể sửa.

b, tính năng tải file xuống sau khi sửa lỗi



Hình 5.11: Thiết kế tính năng tải file xuống sau khi sửa lỗi chính tả

Hình 5.11 mô tả kiến trúc của tính năng tải file xuống sau khi sửa lỗi chính tả. Phần tính năng này được xây dựng bằng thư viện Flask, được chia thành các phần: Routes, Controllers, Services, Models. Ngoài ra còn có hai công cụ bên thứ ba là Aspose Cloud và Google Cloud. Phần tính năng là nơi xử lý request từ client gửi đến.

Khi client gửi yêu cầu (request) tải file đến server, các Router có nhiệm vụ định tuyến, gọi tới các Controllers tương ứng với router nhằm xử lý tác vụ tải file sau khi sửa lỗi chính tả. Các Controller tiếp nhận yêu cầu từ router tiếp theo là gọi đến Services để xử lý tác vụ. Tiếp theo, service sẽ gọi tới Model hoặc Controller sẽ trực tiếp gọi tới Model. Cuối cùng Service sẽ thực hiện việc lưu file vào các công cụ bên thứ ba.

Sau khi kết thúc quy trình trên thì máy chủ sẽ trả về cho người dùng một đường dẫn để tải file xuống.

5.7 Thiết kế API

Như đã trình bày các tính năng và trang web giao tiếp với nhau thông qua API. Hệ thống cung cấp 3 API, cụ thể được ghi trong **Bảng 5.7**

STT	Ý nghĩa	Phương thức	Địa chỉ
1	Kiểm tra lỗi chính tả bằng cách nhập từ bàn phím	POST	/api/v1/check/input
2	Kiểm tra lỗi chính tả bằng file	POST	/api/v1/check/file
3	Tải file sau khi đã lựa chọn gợi ý sửa	POST	/api/v1/download

Bảng 5.7: Danh sách API của hệ thống

5.8 Xây dựng hệ thống

5.8.1 Thư viện và công cụ sử dụng

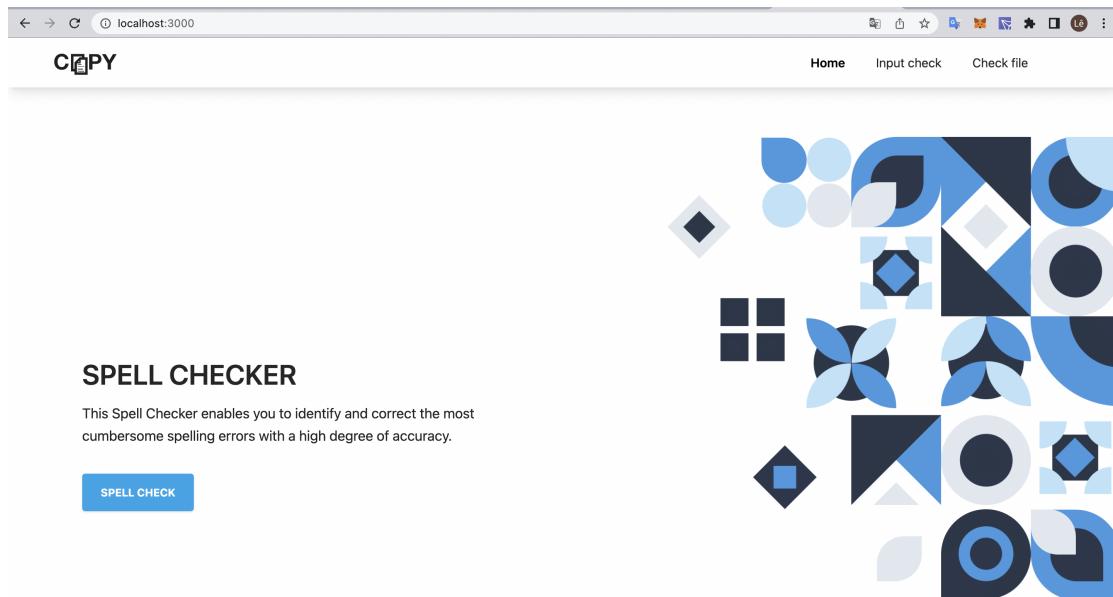
Trong quá trình thực hiện đồ án em sử dụng một số công cụ hỗ trợ việc xây dựng và phát triển frontend, backend và triển khai hệ thống như **Bảng 5.8**

Mục đích	Công cụ	Địa chỉ
IDE	Visual Studio Code	https://code.visualstudio.com/
Backend Framework	ExpressJS	https://expressjs.com/
Backend Framework	Flask	https://flask.palletsprojects.com/en/2.1.x/
Frontend	ReactJS	https://reactjs.org/
Huấn luyện mô hình	Google Colab	https://colab.research.google.com/
Huấn luyện mô hình	PyTorch	https://pytorch.org/
Huấn luyện mô hình	Conda	https://conda.io/

Bảng 5.8: Danh sách công cụ và thư viện sử dụng

5.8.2 Kết quả đạt được

a, Trang chủ



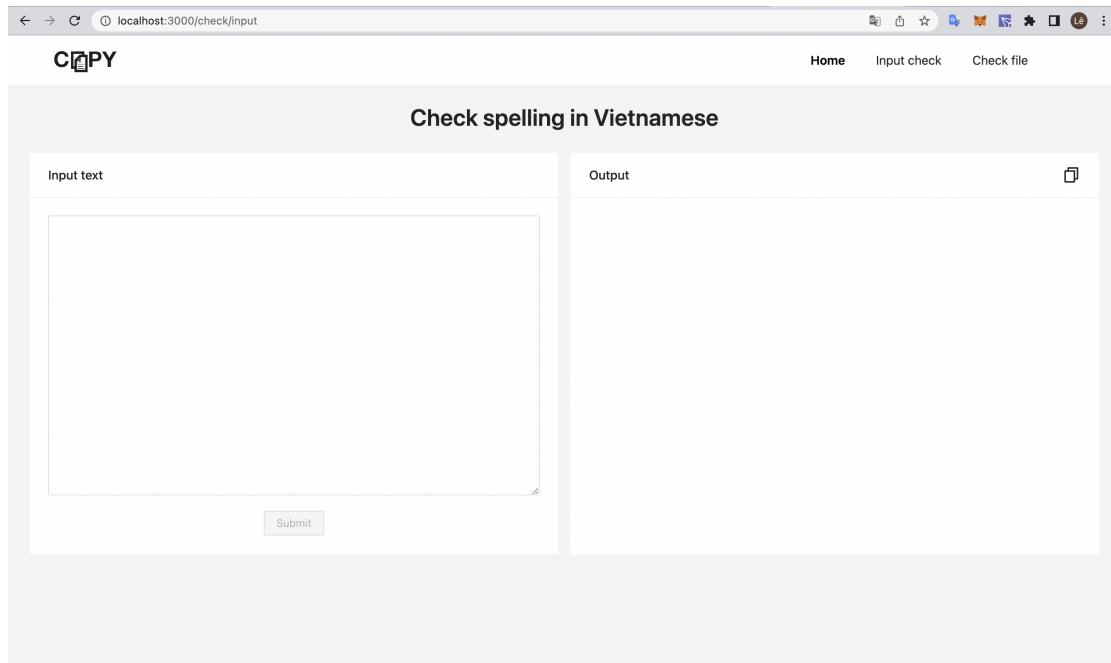
Hình 5.12: Giao diện trang chủ

Trang chủ có header bao gồm logo và menu. Tại menu có ba nút chính là:

- Home: trang chủ
- Input check: chức năng kiểm tra lỗi chính tả bằng cách nhập từ bàn phím.
- Check file: kiểm tra lỗi chính tả bằng file.

Phía dưới header là phần nội dung giới thiệu về chức năng phát hiện và gợi ý lỗi chính tả của hệ thống.

b, Giao diện chức năng kiểm tra lỗi chính tả bằng input



Hình 5.13: Giao diện chức năng kiểm tra lỗi chính tả bằng input

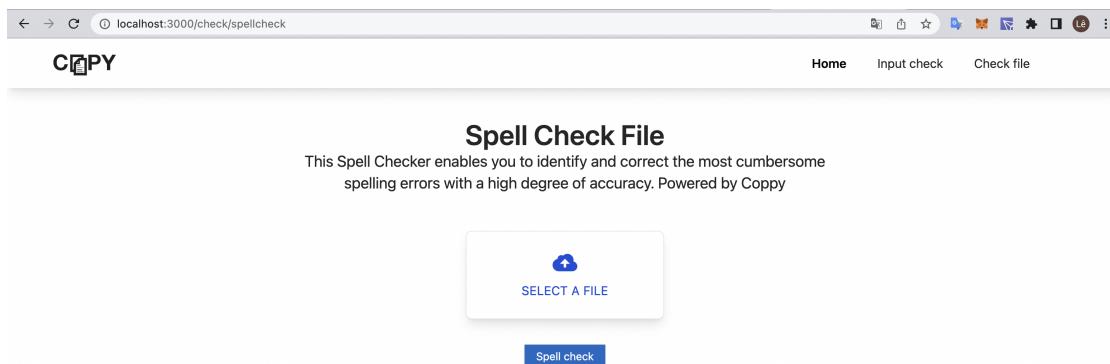
Người dùng nhập văn bản vào ô "Input text". Sau khi nhập xong đoạn văn bản muốn kiểm tra lỗi sai thì người dùng nhấp nút "Submit" để kiểm tra. Sau đó, hệ thống sẽ xử lý và trả lại kết quả ở khung "Output" (**Hình 5.14**). Tại đây, người dùng sẽ nhìn thấy những vị trí sai của từ, khi di chuột tới vị trí sai sẽ được liệt kê ra gợi ý các cách sửa. Người dùng lựa chọn 1 gợi ý. Hệ thống sẽ sửa lại câu theo gợi ý người dùng chọn. Sau khi sửa xong người dùng có thể nhấp vào icon sao chép ở góc phải màn hình ở khung "Output" để sao chép đoạn văn bản đã được chỉnh sửa.

Input text	Output
Lớn lên tôi có thiis quen deer tóc dài phủ gáy là do vậy	thoi quen một lẽ thê
	Lớn lên tôi có thiis quen deer tóc dài phủ gáy là do vậy.

Submit

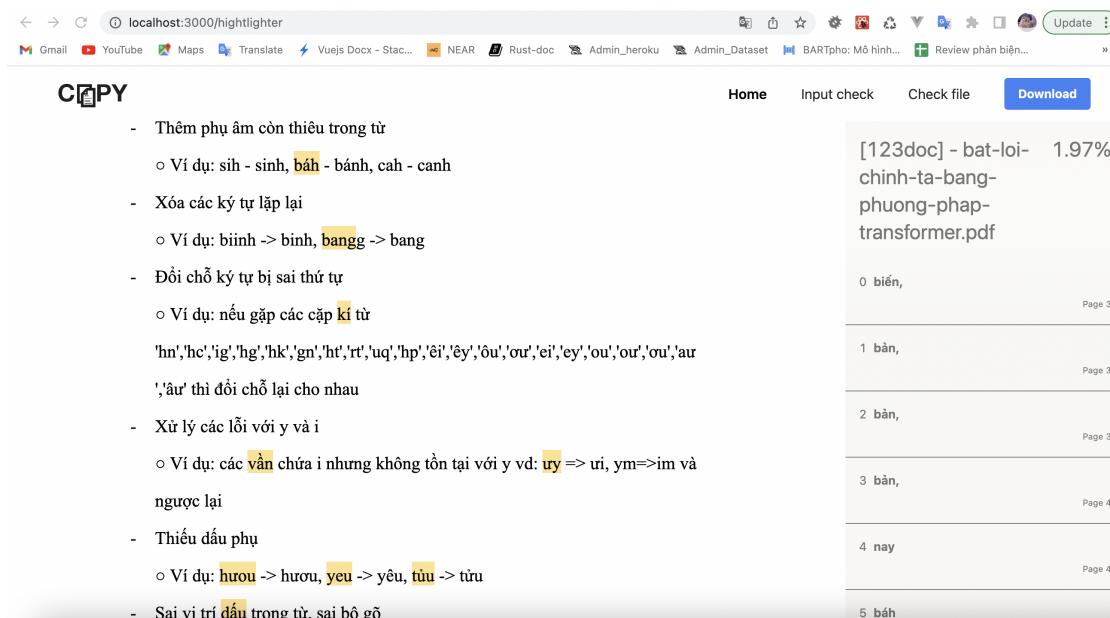
Hình 5.14: Giao diện hiển thị kết quả của chức năng kiểm tra lỗi chính tả bằng cách nhập văn bản từ bàn phím

c, Giao diện chức năng kiểm tra lỗi chính tả bằng file



Hình 5.15: Giao diện chức năng kiểm tra lỗi chính tả bằng file

Người dùng nhập vào nút **SELECT A FILE** và lựa chọn file cần kiểm tra. Sau đó nhập vào nút **Spell Check**. Sau khi hệ thống kiểm tra lỗi chính tả của file sẽ trả về giao diện trang như Hình 5.16



Hình 5.16: Giao diện file sau khi đã được kiểm tra lỗi sai

Tại trang này, những từ bị sai chính tả trong file sẽ được bôi màu và liệt kê thành danh sách ở bên phải màn hình. Khi di chuột đến vị trí sai hệ thống sẽ đưa ra gợi

ý cách sửa cho vị trí đó. Sau khi người dùng lựa chọn thì hệ thống sẽ lưu lại cập nhập bên danh sách bên phải. Khi người dùng sửa xong file thì có thể nhấp vào nút "Download" trên header để tải file mới sau khi đã chỉnh sửa.

5.9 Đóng gói và triển khai

Hệ thống đã được xây dựng và đóng gói thành các file zip. Trong các file zip gồm có source code và một file hướng dẫn cài đặt thực thi source code.

Cấu trúc của một file hướng dẫn sẽ gồm: (i) Ý nghĩa của source code, (ii) Các bước thực thi source code từ cài đặt tới chạy source.

Tất cả source code của toàn bộ hệ thống từ server tới client đã được lưu trữ trên driver với đường dẫn: https://husteduvn-my.sharepoint.com/:f/g/personal/thao_dth176878_sis_hust_edu_vn/Env3IFKxC7VNis8aqiF5Nx4BvUKyW1IL3ICFg58h-Aq2KQ?e=s4sf9k

Kết chương

Chương 5 này đã trình bày chi tiết quá trình thiết kế, xây dựng và phát triển cũng như những công nghệ sử dụng trong quá trình triển khai xây dựng hệ thống phát hiện và sửa lỗi sai chính tả. Từ những nghiên cứu và thực nghiệm các mô hình cùng với việc phát triển một hệ thống phát hiện và sửa lỗi chính tả, tiếp theo em sẽ trình bày kết luận dành cho toàn bộ đồ án. Phần đó sẽ được trình bày trong chương tiếp theo - Chương 6.

CHƯƠNG 6. KẾT LUẬN

Trong chương này sẽ trình bày kết luận của đồ án và hướng phát triển trong tương lai.

6.1 Kết luận

Trong đồ án này, em đã cải thiện giao diện của hệ thống kiểm tra đạo văn của trường Công Nghệ Thông Tin và Truyền Thông. Bên cạnh đó em đã xây dựng hai mô đun là mô đun kiểm tra lỗi chính tả và mô đun sửa lỗi chính tả nhằm xây dựng hệ thống kiểm tra và sửa lỗi chính tả tích hợp vào hệ thống kiểm tra đạo văn của trường Công Nghệ Thông Tin và Truyền Thông.

Về mặt cơ chế đề xuất, em đã đã xây dựng và hoàn thiện mô đun cho bài toán phát hiện và sửa lỗi chính tả, ngoài ra em cũng đã hoàn thiện việc cải thiện giao diện cho hệ thống kiểm tra đạo văn và hoàn thiện hệ thống phát hiện và gợi ý sửa lỗi chính tả giúp nâng cao chất lượng báo cáo khoa học. Ngoài ra, trong quá trình làm đồ án em đã tìm hiểu và học hỏi được nhiều kiến thức về đặc trưng của tiếng Việt và hiểu được mô hình Transformer.

Về mặt ứng dụng, em đã hoàn thành được hệ thống phát hiện và gợi ý sửa lỗi sai cùng với một số tính năng phù hợp với nhu cầu của người dùng thực tế như: (i) Kiểm tra chính tả bằng file, (ii) Sửa lỗi chính tả trên file báo cáo, (iii) Tải file sau khi sửa lỗi chính tả. Việc triển khai và phát triển hệ thống giúp cho giải pháp mà ĐATN đã đưa ra sẽ tiếp cận gần hơn với thực tiễn.

Dưới đây là một số vấn đề còn tồn đọng trong ĐATN này:

- Về tiền xử lý dữ liệu: dữ liệu báo cáo khoa học có đa dạng các loại nội dung: văn bản, công thức toán học, mã nguồn, URL... nên việc loại bỏ các phần không cần xử lý khó khăn.
- Tồn tại các lỗi sai chính tả trên các đồ án, dẫn đến tập dữ liệu huấn luyện và tập dữ liệu kiểm thử vẫn có một số ít lỗi chính tả.
- Số lượng gợi ý sửa cho 1 từ sai chính tả khá nhiều, người dùng sẽ cần tìm từ thích hợp trên một danh sách dài.
- Mô hình chưa được tối ưu hóa trong các file lớn.
- Mô hình N-gram cần bộ dữ liệu lớn, khi thực thi chiếm nhiều tài nguyên.
- Một số luật phổ biến nhưng chưa được áp dụng trong đồ án.
- Chưa tích hợp được hệ mô-đun phát hiện và gợi ý sửa lỗi chính tả vào trong

hệ thống Coopy.

6.2 Hướng phát triển trong tương lai

Đối với các nghiên cứu tiếp theo, em sẽ tiến hành mở rộng hệ thống và giải quyết các vấn đề còn tồn đọng phía trên.

- Cải thiện phương pháp tiền xử lý dữ liệu: áp dụng biểu thức chính quy (regex) hoặc xây dựng các model hỗ trợ tiền xử lý dữ liệu...
- Cải thiện chất lượng dữ liệu đồ án: xử lý bằng tay hoặc sử dụng mô hình phát hiện lỗi sai chính tả để loại bỏ các câu có lỗi sai chính tả.
- Tìm cách thu gọn danh sách từ gợi ý xuống còn nhỏ hơn 3: xây dựng phương pháp đánh trọng số cho các từ gợi ý, sau đó chọn 3 từ có trọng số lớn nhất để hiển thị.
- Tối ưu hóa mô hình để có thể xử lý các file lớn nhanh và chính xác hơn.
- Bổ sung thêm những tập luật phổ biến.
- Tích hợp hai mô đun phát hiện và sửa lỗi chính tả vào hệ thống kiểm tra đạo văn của trường Công Nghệ Thông Tin và Truyền Thông

TÀI LIỆU THAM KHẢO

- [1] Y. Lu, Y. Shi, G. Jia **and** J. Yang, “A new method for semantic consistency verification of aviation radiotelephony communication based on LSTM-RNN,” **in***2016 IEEE International Conference on Digital Signal Processing (DSP)* IEEE, 2016. DOI: 10.1109/icdsp.2016.7868592.
- [2] N. Thi Xuan Huong, T.-T. Dang, A.-C. Le **and others**, “Using large n-gram for vietnamese spell checking,” **in***Knowledge and Systems Engineering* Springer, 2015, **pages** 617–627.
- [3] M. Lewis, Y. Liu, N. Goyal **and others**, “Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” *arXiv preprint arXiv:1910.13461*, 2019.
- [4] N. L. Tran, D. M. Le **and** D. Q. Nguyen, “BARTpho: Pre-trained Sequence-to-Sequence Models for Vietnamese,” **in***Proceedings of the 23rd Annual Conference of the International Speech Communication Association* 2022.
- [5] H. Tran, C. V. Dinh, L. Phan **and** S. T. Nguyen, “Hierarchical transformer encoders for vietnamese spelling correction,” **in***International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems* Springer, 2021, **pages** 547–556.
- [6] D. Q. Nguyen **and** A. T. Nguyen, “PhoBERT: Pre-trained language models for Vietnamese,” **in***Findings of the Association for Computational Linguistics: EMNLP 2020* 2020, **pages** 1037–1042.
- [7] P. Rawat **and** A. N. Mahajan, “Reactjs: A modern web development framework,” *International Journal of Innovative Science and Research Technology*, **jourvol** 5, **number** 11, 2020.