

BỘ GIÁO DỤC VÀ ĐÀO TẠO

ĐẠI HỌC UEH - TRƯỜNG CÔNG NGHỆ VÀ

THIẾT KẾ

KHOA CÔNG NGHỆ THÔNG TIN KINH DOANH

CHUYÊN NGÀNH KHOA HỌC DỮ LIỆU



CÔNG TY TNHH PASONA TECH

VIỆT NAM



P A S O N A

Khóa luận tốt nghiệp - học kỳ doanh nghiệp

BÁO CÁO HỌC KỲ DOANH NGHIỆP TẠI CÔNG TY TNHH PASONA TECH VIỆT NAM

Họ tên sinh viên: Đoàn Vũ Minh Thanh

Lớp: DS001

Khóa: 46

Họ tên giáo viên hướng dẫn: Nguyễn An Tế

Họ tên người hướng dẫn tại doanh nghiệp:

Nguyễn Phi Hùng

Niên khóa: 2020 - 2024

Tp Hồ Chí Minh, ngày 25 tháng 10 năm 2023

MỤC LỤC

MỤC LỤC	1
DANH MỤC CÁC CHỮ VIẾT TẮT VÀ KÝ HIỆU	4
DANH MỤC CÁC SƠ ĐỒ, ĐỒ THỊ	7
DANH MỤC CÁC BẢNG BIỂU	8
CHƯƠNG 1: GIỚI THIỆU	9
1. Giới thiệu doanh nghiệp và bộ phận trực thuộc	9
2. Mục tiêu thực tập.....	11
a. Từ phía doanh nghiệp	11
b. Từ phía sinh viên.....	11
CHƯƠNG 2: NỘI DUNG THỰC TẬP	13
1. Đề án 1: Quản lý source code bằng Git	13
a. Mục tiêu đề án.....	13
b. Nội dung công việc.....	13
c. Những kết quả đạt được	17
2. Đề án 2: Nhận nội dung thực tập và lên kế hoạch thực hiện dự án	18
a. Mục tiêu đề án.....	18
b. Nội dung công việc.....	18
c. Những kết quả đạt được	33
3. Đề án 3: Tổ chức source code theo mô hình thiết kế MVC và tạo API cho chatbot	34
a. Mục tiêu đề án.....	34
b. Nội dung công việc.....	34

c.	<i>Những kết quả đạt được</i>	36
4.	Đề án 4: Tiền xử lý văn bản từ tin nhắn tự do của khách hàng	37
a.	<i>Mục tiêu đề án</i>	37
b.	<i>Nội dung công việc</i>	37
c.	<i>Những kết quả đạt được</i>	41
5.	Đề án 5: Ứng dụng mô hình học sâu để xác định chủ đề tin nhắn.....	42
a.	<i>Mục tiêu đề án</i>	42
b.	<i>Nội dung công việc</i>	42
c.	<i>Những kết quả đạt được</i>	48
6.	Đề án 6: Cấu thành câu truy vấn SQL	50
a.	<i>Mục tiêu đề án</i>	50
b.	<i>Nội dung công việc</i>	50
c.	<i>Những kết quả đạt được</i>	56
7.	Đề án 7: Thiết kế giao diện người dùng cho chatbot và gọi API.....	57
a.	<i>Mục tiêu đề án</i>	57
b.	<i>Nội dung công việc</i>	57
c.	<i>Những kết quả đạt được</i>	60
CHƯƠNG 3: KẾT LUẬN		61
1.	Những kết quả đạt được	61
a.	<i>Lý thuyết</i>	61
b.	<i>Công nghệ, kỹ năng chuyên môn</i>	63
c.	<i>Kỹ năng mềm</i>	74
2.	Những thiếu sót, hạn chế cần cải thiện	75

CHƯƠNG 4: PHỤ LỤC KỸ THUẬT	81
<i>Đề án 1: Quản lý source code bằng Git.....</i>	<i>81</i>
<i>Đề án 2: Nhận nội dung thực tập và lên kế hoạch thực hiện dự án.....</i>	<i>93</i>
<i>Đề án 3: Xây dựng mô hình thiết kế MVC và tạo API cho chatbot</i>	<i>94</i>
<i>Đề án 4: Tiền xử lý văn bản từ tin nhắn tự do của khách hàng</i>	<i>96</i>
<i>Đề án 5: Ứng dụng mô hình học sâu để xác định chủ đề tin nhắn</i>	<i>102</i>
<i>Đề án 6: Cấu thành câu truy vấn SQL.....</i>	<i>113</i>
<i>Đề án 7: Thiết kế giao diện người dùng cho chatbot và gọi API.....</i>	<i>118</i>
CHƯƠNG 5: NHẬT KÝ CÔNG VIỆC.....	126
TÀI LIỆU THAM KHẢO.....	130
PHÊ DUYỆT VÀ NHẬN XÉT CỦA CÔNG TY VÀ NGƯỜI HƯỚNG DẪN ĐẠI DIỆN DOANH NGHIỆP	131

DANH MỤC CÁC CHỮ VIẾT TẮT VÀ KÝ HIỆU

Chữ viết tắt/ký hiệu	Từ gốc	Ý nghĩa
KPI	Key Performance Indicator	Là một cụm từ tiếng Anh, có nghĩa là "Chỉ số hiệu suất chính"; là các thước đo được sử dụng để đánh giá mức độ hoàn thành của mục tiêu hoặc kế hoạch trong một tổ chức, doanh nghiệp, hoặc dự án cụ thể.
AI	Artificial Intelligence	Là một cụm từ tiếng Anh, có nghĩa là "Trí tuệ nhân tạo"; là một lĩnh vực trong khoa học máy tính và công nghệ liên quan đến việc phát triển các hệ thống và máy móc có khả năng thực hiện các tác vụ yêu cầu sự "suy nghĩ" hoặc "ra quyết định" tương tự con người. Mục tiêu của AI là tạo ra các hệ thống có khả năng học hỏi, tư duy, và đưa ra quyết định dựa trên dữ liệu và kinh nghiệm trước đó mà không cần phải được lập trình một cách cụ thể.
RESTful API	Representational State Transferful Application Programming Interface	Là một kiểu thiết kế giao diện (API) cho các ứng dụng web. Nó tuân thủ các nguyên tắc và quy tắc của REST, một kiến trúc chuẩn dành cho các hệ thống phân tán. RESTful API được thiết kế để giúp các ứng dụng giao tiếp với nhau thông qua mạng.
MVC	Model-View-Controller	Là một mô hình thiết kế phổ biến trong lập trình phần mềm. Nó tách biệt các thành phần của ứng dụng thành ba phần chính để giúp quản lý mã nguồn và cung cấp tính linh hoạt cao.
SQL	Structured Query Language	Là một ngôn ngữ lập trình sử dụng để quản lý và truy vấn cơ sở dữ liệu quan hệ (relational database). SQL

		cho phép người dùng thực hiện các tác vụ như tạo, cập nhật, xóa và truy vấn dữ liệu từ cơ sở dữ liệu.
HTML	Hypertext Markup Language	Là ngôn ngữ đánh dấu tiêu chuẩn được sử dụng để tạo cấu trúc và định dạng nội dung trên các trang web. HTML cung cấp các phần tử (elements) và cú pháp để mô tả cách một trang web sẽ hiển thị thông tin.
CSS	Cascading Style Sheets	Là một ngôn ngữ đánh dấu được sử dụng để kiểm soát cách mà các trang web được hiển thị trên trình duyệt web. CSS giúp phân tách cấu trúc (HTML) của một trang web và cách mà nó được hiển thị (layout, màu sắc, font chữ, v.v.).
PHP	Personal Home Page Tools	Là một ngôn ngữ lập trình phía máy chủ (server-side scripting language) dùng để phát triển các ứng dụng web động và tương tác với cơ sở dữ liệu.
VSCode	Visual Studio Code	Là một môi trường phát triển tích hợp (IDE) dành cho việc phát triển ứng dụng web và các ứng dụng khác. Được phát triển bởi Microsoft, VSCode là một ứng dụng mã nguồn mở và miễn phí.
repo	repository	Là kho lưu trữ, nơi chứa các tệp tin, mã nguồn, dự án hoặc dữ liệu. Trong ngữ cảnh phát triển phần mềm, một repository thường được sử dụng để quản lý và lưu trữ mã nguồn của một dự án phần mềm.
CSKH	Chăm sóc khách hàng	Là một ngành nghề hoặc bộ phận trong các doanh nghiệp và tổ chức, có trách nhiệm tiếp nhận, xử lý và giải quyết các yêu cầu, thắc mắc, khiếu nại của khách hàng.
NLP	Natural	Là một lĩnh vực của trí tuệ nhân tạo (AI) tập trung vào

	Language Processing	việc phân tích, xử lý và tương tác với ngôn ngữ tự nhiên một cách tự động và thông minh.
NER	Name of Entity Recognition	Là một tác vụ trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên (NLP). Nhiệm vụ của NER là nhận diện và phân loại các thực thể trong văn bản thành các loại như tên riêng, địa danh, ngày tháng, số tiền, v.v.
PHR	Porters Human Resource	Là một bộ phận về dịch vụ tuyển dụng của công ty Pasona Tech Việt Nam
BA	Business Analyst	Một chuyên viên phân tích kinh doanh, người tìm hiểu, đánh giá và đề xuất các giải pháp trong lĩnh vực kinh doanh và công nghiệp.
AJAX	Asynchronous JavaScript and XML	Là một tập hợp các kỹ thuật phát triển web được sử dụng để tạo ra ứng dụng web bất đồng bộ, cho phép cập nhật một phần của trang web mà không cần phải tải lại toàn bộ trang.
JSON	JavaScript Object Notation	Là một định dạng dữ liệu phổ biến được sử dụng để truyền tải và lưu trữ dữ liệu. Nó được thiết kế để dễ đọc và dễ viết cho con người, cũng như dễ dùng và dễ phân tích cho máy móc. JSON sử dụng cú pháp giống JavaScript để mô tả dữ liệu dưới dạng các cặp "key-value" (tức là cặp "tên": "giá trị"). Dữ liệu có thể là số nguyên, số thập phân, chuỗi, danh sách, đối tượng, hoặc giá trị boolean (true hoặc false). JSON không hỗ trợ các hàm hoặc mã lệnh phức tạp.

DANH MỤC CÁC SƠ ĐỒ, ĐỒ THỊ

<i>Hình 1. Kiến trúc Transformers</i>	<i>71</i>
<i>Hình 2. Mô hình BERT</i>	<i>73</i>

DANH MỤC CÁC BẢNG BIỂU

<i>Bảng 1. Các ngôn ngữ và framework sẽ sử dụng trong dự án</i>	<i>25</i>
<i>Bảng 2. Bảng ước lượng công việc (Estimation)</i>	<i>32</i>

CHƯƠNG 1: GIỚI THIỆU

1. Giới thiệu doanh nghiệp và bộ phận trực thuộc

Pasona Tech Vietnam thuộc sở hữu của Pasona Group, một tập đoàn đa quốc gia có trụ sở chính tại Tokyo, Nhật Bản. Tập đoàn được thành lập vào năm 1976, tập trung vào các lĩnh vực như tư vấn nhân sự, đào tạo và giáo dục, giải pháp công nghệ thông tin, và nhiều lĩnh vực khác. Một số khách hàng quan trọng của Pasona Group gồm Mitsubishi Corporation, Toshiba, Toyota Group, Panasonic, Canon,... Tập đoàn đã mở rộng sự hiện diện của mình tới nhiều quốc gia trên thế giới, có thể kể đến Hoa Kỳ, Trung Quốc, Hàn Quốc, Singapore,..., và vào năm 2004, Pasona chính thức mở công ty con tại Việt Nam với tên gọi là Pasona Tech Vietnam, với 100% vốn đầu tư ở Nhật. Ngoài trụ sở chính được đặt tại thành phố Hồ Chí Minh, Pasona còn có các chi nhánh tại Hà Nội và Đà Nẵng. (PasonaVN, 2022)

Pasona Tech Vietnam tiên phong trong lĩnh vực công nghệ tại Việt Nam, với sứ mệnh thúc đẩy sự phát triển bền vững bằng cách tạo ra các giải pháp công nghệ tiên tiến, cùng khách hàng và đối tác cố gắng giải quyết các vấn đề việc làm cho người lao động Việt Nam. Hiện nay, Pasona Tech Vietnam hoạt động trong ba lĩnh vực kinh doanh cốt lõi là HR, BPO và ITO.

- Dịch vụ ITO: Lập trình Android; QA/QC; Thuê ngoài (outsourcing) tùy theo mô hình lab; Hỗ trợ mở rộng chi nhánh
- Dịch vụ BPO
- Dịch vụ nhân sự (HR): Hỗ trợ tuyển dụng; Tư vấn nhân sự; Hợp tác huấn luyện nhân sự

Công ty tập trung vào việc áp dụng công nghệ hiện đại như Trí tuệ nhân tạo (AI), Internet of Things (IoT), và Blockchain để đưa ra những giải pháp sáng tạo giúp doanh

nghiệp vượt qua các thách thức kỹ thuật số. Tại Pasona Tech Vietnam, công ty tin tưởng rằng công nghệ có thể thay đổi cách thức hoạt động và tạo nên giá trị bền vững.

Trong khoảng thời gian thực tập tại doanh nghiệp, sinh viên được phân công về phòng Software. Nhiệm vụ chính của phòng Software là thực hiện xây dựng, thiết kế, lập trình các sản phẩm theo đơn hàng của khách hàng thông qua các tài liệu được cung cấp từ phòng PHR và BA, hỗ trợ khách hàng cài đặt, kiểm tra các sản phẩm, chương trình, phần mềm, thiết bị hạ tầng tùy theo mô hình lab, đồng thời nghiên cứu các công nghệ mới và tham gia vào quá trình chuyển đổi số của công ty, xây dựng các ứng dụng hỗ trợ các công việc chuyên môn từ yêu cầu của các phòng ban khác.

Chương trình thực tập tại Pasona Tech Vietnam kéo dài 8 tuần (từ 07/08/2023 đến 13/10/2023), tập trung đào tạo kiến thức và kỹ năng thực hành lập trình web, xây dựng cơ sở dữ liệu, thiết kế Dashboard hỗ trợ việc nghiên cứu, phân tích khách hàng, tình hình kinh doanh, thị trường, và phát triển các dự án trí tuệ nhân tạo (AI), làm quen và thành thạo các ngôn ngữ lập trình thông dụng như Python, SQL, HTML, CSS, Javascript, PHP,... Cụ thể, trong thời gian này, sinh viên sẽ tham gia một dự án cá nhân đó là xây dựng một Chatbot tích hợp trên trang web công việc Pasona.vn của công ty, với sự hướng dẫn của anh Nguyễn Phi Hùng - Technical Manager và sự cố vấn của anh Nguyễn Hoàng Hiệp - người xây dựng trang Pasona.vn. Qua việc thực hiện dự án cá nhân của sinh viên, công ty kỳ vọng sinh viên học tập các ngôn ngữ, framework cần thiết cho lập trình front-end và back-end, xây dựng Chatbot cơ bản và chạy thành công trên trang web trong môi trường thử nghiệm.

2. Mục tiêu thực tập

a. Từ phía doanh nghiệp

Nhận thấy được giá trị khai thác tiềm năng của dữ liệu, cùng sự phát triển, phổ biến của công nghệ Trí tuệ nhân tạo (AI), công ty Pasona Tech Vietnam, đặc biệt là bộ phận ITO, kỳ vọng sẽ đào tạo thành công một thế hệ lao động trẻ nghiên cứu, tạo thành các sản phẩm ứng dụng AI, nhằm đa dạng hóa, mở rộng các sản phẩm, dịch vụ của công ty nhằm cung cấp những trải nghiệm tối ưu nhất về mảng nhân sự (xử lý hồ sơ ứng viên, hỗ trợ giải đáp thắc mắc của ứng viên, nghiên cứu phương thức quảng bá để tiếp cận nhiều ứng viên mục tiêu, đạt các chỉ tiêu (KPI) đã đề ra...); ngoài ra với nhu cầu xử lý khối lượng thông tin, nhiều tác vụ còn thủ công, rườm rà, phía công ty kỳ vọng xây dựng kho dữ liệu và hệ thống kế toán nội bộ và các tính năng khác (trong tương lai), hướng đến mục tiêu chuyển đổi số trong doanh nghiệp, tối ưu hóa luồng thông tin, giao tiếp giữa các phòng ban.

Với kế hoạch phát triển dài hạn, công ty chào đón, đào tạo ngay từ giai đoạn đầu cho các thực tập sinh có cùng đam mê, mục tiêu về phát triển AI, chuyển đổi số, đồng thời có kiến thức, kinh nghiệm làm việc với các ngôn ngữ back-end, front-end web; các thuật toán, mô hình học sâu và tư duy xử lý ngôn ngữ tự nhiên, tư duy phân tích dữ liệu. Sau thời gian đào tạo, nếu thực tập sinh hoàn thành quá trình này với kết quả tốt và có định hướng gắn bó, phát triển sự nghiệp tại công ty thì sẽ được xem xét về việc trở thành nhân viên chính thức của Pasona Tech Vietnam.

b. Từ phía sinh viên

Mục tiêu chính của bản thân sinh viên khi tham gia thực tập tại Pasona Tech Vietnam là tích lũy được thật nhiều kinh nghiệm thực tiễn trong công việc lập trình viên, tiếp cận và học hỏi các công cụ, ngôn ngữ, kiến thức chuyên ngành và xu hướng công nghệ nổi bật, phổ biến được áp dụng rộng rãi trong các doanh nghiệp để nâng cao trình độ, nắm bắt nhu cầu, xu thế của các doanh nghiệp và thị trường nhân lực hiện nay. Cụ thể:

- Về phần lập trình back-end, sinh viên hy vọng bản thân nắm rõ mô hình thiết kế MVC để sắp xếp code có tổ chức, sạch sẽ hơn, dễ dàng trong việc truy vết khi cần điều chỉnh, nghiên cứu; nắm rõ hơn về cách phát triển một trang web, tận dụng và mở rộng những kiến thức, hiểu biết của sinh viên về ngôn ngữ Python, đồng thời học hỏi thêm một ngôn ngữ lập trình mới là PHP và framework Laravel để biết cách xây dựng cơ sở dữ liệu, triển khai truy vấn cơ sở dữ liệu, cấu hình cơ sở dữ liệu ở phần Back-end, và hiểu hơn về cách vận hành của một trang web để triển khai dữ liệu cho front-end.
- Về phần lập trình front-end, sinh viên kỳ vọng mình học được cách cấu hình để chia components, cách gọi dữ liệu từ web API và cách lấy dữ liệu truyền tải lên giao diện thông qua các ngôn ngữ cơ bản như HTML, CSS, Javascript, PHP và framework Laravel, các tư duy lập trình của những ngôn ngữ này để phân code, bộ nhớ cũng như sản phẩm được tối ưu nhất.
- Việc bổ sung, nâng cao cả về mặt back-end và front-end giúp sinh viên tự tin tham gia vào các dự án thực tế của công ty, thông qua đó sinh viên được trải nghiệm, học hỏi về quy trình làm việc của một môi trường kinh nghiệm và văn hóa công sở của công ty; nghiên cứu và xây dựng thành công các sản phẩm, góp phần hoàn thành kế hoạch phát triển dài hạn trong giai đoạn mới của công ty.
- Bên cạnh kiến thức chuyên môn, sinh viên kỳ vọng bản thân tích lũy thêm kinh nghiệm quản lý và chia sẻ tài nguyên, tài liệu, đặc biệt là với các tài liệu mang tính bảo mật cao như source code, kho dữ liệu,...; kỹ năng làm việc nhóm, giao tiếp, truyền tải ý kiến, quản lý thời gian và tiến độ công việc, cũng như khả năng ngôn ngữ tiếng Anh và tiếng Nhật để có thể giao tiếp với cấp trên hoặc đối tác một cách chủ động, hiệu quả hơn.

CHƯƠNG 2: NỘI DUNG THỰC TẬP

1. Đề án 1: Quản lý source code bằng Git

a. Mục tiêu đề án

Sinh viên mong muốn thông qua hướng dẫn từ anh Hùng, sinh viên sẽ học hỏi được cách làm việc với dữ liệu mật (code) và cách trao đổi, giao tiếp tài liệu của lập trình viên qua Git, đồng thời sinh viên sẽ trở nên thành thạo sử dụng các câu lệnh cơ bản và luồng làm việc của các nhánh, Git.

b. Nội dung công việc

Với dự án cá nhân lần này, sinh viên sẽ tạo chatbot như bổ sung thêm một chức năng trên trang Pasona.vn, vì vậy nên trong repository “Pasona”, sinh viên sẽ tạo một nhánh (branch) mới tên search-api, và tất cả phần code back-end xử lý chatbot của sinh viên sẽ được lưu trong nhánh này. Còn phần thiết kế giao diện front-end sẽ được viết tiếp trong file footer.php thuộc nhánh src.

Đầu tiên, sinh viên cài đặt môi trường thực hành bằng cách tải công cụ Git (<https://git-scm.com/>), phiên bản 2.42.0. Branch search-api đã được anh quản lý tạo trước. Để làm việc với git tại local, sinh viên sẽ clone repository “Pasona” về máy bằng câu lệnh “git clone <url>”. Ở local, ta tạo một thư mục có tên trùng với tên repository ta sẽ clone. Trong VSCode, ở phần terminal, ta di chuyển (cd) vào thư mục vừa tạo, và gõ câu lệnh “git clone <copied url>”. Sau khi clone thành công, sinh viên di chuyển (cd) vào folder Pasona. Sau đó dùng lệnh “git status” để kiểm tra sinh viên đang ở branch nào.

Nhận thấy sinh viên đang ở brand develop, khác với branch sinh viên được chỉ định là search-api nên sinh viên sẽ sử dụng lệnh git checkout <tên branch> để đổi sang branch của sinh viên. Cuối ngày làm việc, sinh viên sẽ tải các file, thư mục đã chỉnh sửa, tạo mới từ local lên máy chủ remote: Lệnh git status -uall để hiển thị các file đã chỉnh sửa (bao gồm các file trong folder được tạo mới); Git add <tên file> để chuyển trạng thái file

từ untracked thành staged, sẵn sàng để commit. `Git commit -m "<message>":` chuyển toàn bộ các file từ trạng thái staged thành committed, kèm theo cú pháp `-m "<commit message>"` để thêm chú thích các thay đổi, điều chỉnh gì được thực hiện và đăng tải (bắt buộc). Khi file đã được commit thành công thì có thể dùng câu lệnh `"git push"` để đẩy các file đã chỉnh sửa từ local lên máy chủ remote. Sau khi thực hiện lệnh push, file mới đã được cập nhật trên máy chủ remote.

Anh Hùng đã giới thiệu về 1 số công cụ giúp quản lý source code một cách tiện lợi, thao tác đơn giản hơn giữa các lập trình viên, trong đó sinh viên thấy SourceTree có giao diện thân thiện với người dùng nhất. Vì thế, sinh viên đã tự tìm hiểu về cách sử dụng công cụ này để có thể quản lý source code với các thao tác kéo thả chuột đơn giản, thay vì phải ghi nhớ các câu lệnh git phức tạp.

Để clone một dự án từ remote về local, ta chọn nút Clone trên thanh công cụ. Khi màn hình Clone xuất hiện, ô đầu tiên sẽ điền URL của project, ô thứ 2 sẽ điền địa chỉ thư mục chứa project, ô thứ 3 sẽ điền tên của thư mục chứa project. Khi clone thành công, giao diện làm việc chính trên SourceTree sẽ xuất hiện. Chú ý phải trở đúng vào branch mình đang làm việc. Nếu trở sai thì khi commit/push lên sẽ vào nhầm branch khác

Các loại trạng thái file thường sử dụng:

- Pending: Hiện thị tất cả các lớp - cả những lớp đã được chỉnh sửa và những lớp chưa được chỉnh sửa.
- Untracked: những file mới được tạo hoặc file cũ nhưng di chuyển đến thư mục mới
- Ignored: những file bị SourceTree ignore thường sẽ tự sinh ra trong quá trình chạy chương trình, ví dụ như bộ nhớ cache phụ thuộc (nội dung của `/node_modules` hoặc `/packages.`), mã đã biên dịch (các file `.o` , `.pyc` , và `.class`).
- Modified: những file đã chỉnh sửa nội dung

Khi muốn commit/push các thay đổi ở local lên remote, ta chọn loại trạng thái file mà mình muốn hiển thị file mình cần cập nhật, các file tương thích với trạng thái file được lựa chọn sẽ hiển thị trong cửa sổ “Unstaged files”. Để chuyển các file thành staged, thay vì gõ lệnh “git add <tên file>” thì ta chỉ cần chọn dấu + tương ứng với file đó. Ngoài ra, ta có thể xem những nội dung nào đã được chỉnh sửa trong file (so sánh giữa nội dung của file được lưu trên remote với nội dung được lưu gần đây nhất của file đang được chọn), màu xanh là thêm mới, còn màu đỏ là bị mất đi. Sau khi đã chuyển các file cần thiết thành trạng thái stage, để commit, ta chọn nút “Commit” ở góc phải phía trên của màn hình. Để nhập nội dung commit, thay vì sử dụng lệnh “git commit -m <nội dung commit>”, ta chỉ cần nhập nội dung vào ô cửa sổ phía dưới màn hình. Và nếu ta muốn push những thay đổi lên remote ngay sau khi commit, ta chọn “Push changes immediately to <branch>”.

Sau khi chọn commit, SourceTree hiển thị thay đổi lên mục history (tree graph), và để kiểm tra các file đã được push lên remote chưa, ta vào branch của repo mà ta đã push trên Gitlab, nhận thấy các file đã xuất hiện kèm theo nội dung commit đã nhập khi thực hiện commit tại local.

Để tạo nhánh mới, ta chọn nút Branch trên thanh công cụ. Sau khi điền tên branch mới, chọn “Checkout New Branch” và bấm “Create Branch”, một nhánh mới được tạo ra và khi đó ta sẽ di chuyển và làm việc trên branch mới này. Thông thường khi làm 1 dự án, mỗi thành viên sẽ phụ trách một mảng / một phần tính năng, và dùng 1 branch để cập nhật, lưu trữ các code liên quan đến phần việc của họ. Khi họ đã hoàn thành xong phần việc của họ thì họ sẽ nhập (merge) branch của họ vào branch chính của toàn dự án, thế nhưng đôi khi nội dung code của 2 branch bị mâu thuẫn (conflict), làm gián đoạn quá trình sáp nhập (merge). SourceTree có hỗ trợ giải quyết conflict như sau:

- + Để cấu hình (config) cách xử lý conflict code: Trên thanh công cụ, chọn Tools → Options. Trong cửa sổ hiện ra, vào mục Diff và config các mục

sau: chuyển các mục: External Diff Tool và Merge Tool thành “Custom”; ở mục Diff Command và Merge Command, gắn đường dẫn tới công cụ giải quyết conflict (ở đây sinh viên chọn Visual Studio Code); thêm các argument ‘--diff --wait "\$LOCAL" "\$REMOTE"' vào Diff Command và argument ‘-n --wait "\$MERGED"' vào Merge Command.

Giả sử ta có branch lớn là main và branch con là branch_2, trong hai branch đó đều tồn tại một file .txt nhưng hai nội dung khác nhau. Bây giờ sinh viên muốn merge branch branch_2 vào branch main. Để merge branch branch_2 vào branch main, sinh viên sẽ di chuyển vào branch main, sử dụng “Pull”, chọn branch cần pull từ remote về là branch branch_2, và chọn option “Commit merged changes immediately” để merge branch_2 vào main.

Vì có conflict nên merge không thành công và SourceTree hiện lên thông báo lỗi, việc commit không thành công. Để giải quyết conflict, ta chuột phải vào file cần giải quyết conflict, chọn Resolve Conflict → Launch External Merge Tool. Giao diện Resolve Conflict của Visual Studio Code hiện ra. Sẽ có 3 lựa chọn nhanh cho phần giải quyết:

- Accept current change: giữ lại thay đổi phía nhánh hiện tại, xóa thay đổi của nhánh được merge vào
- Accept incoming change: xóa thay đổi của nhánh hiện tại và giữ lại thay đổi của nhánh được merge vào
- Accept both changes giữ lại cả hai thay đổi

Nếu muốn giữ lại 1 phần từ cả hai phía thì có thể chỉnh sửa thủ công.

Chỉnh sửa xong ta lưu file lại, và ở phía giao diện của SourceTree, thông báo conflict sẽ không còn xuất hiện nữa. Ta chỉ commit file vừa chỉnh sửa, bỏ qua file .orig (lưu giữ bản mặc định thừa ra từ việc Resolve Conflict).

c. Những kết quả đạt được

Thông qua đồ án 1, sinh viên đã thành thạo cách làm việc với Git, các thao tác lấy code từ remote, đưa code từ local lên remote, cũng như biết cách giải quyết khi xảy ra conflict code. Việc làm quen với SourceTree giúp sinh viên tiết kiệm thời gian cập nhật tiến độ công việc vào cuối ngày. Từ đó, sinh viên học thêm được cách giao tiếp, trao đổi tài liệu, công việc giữa các lập trình viên một cách chuyên nghiệp, hiệu quả hơn.

2. Đề án 2: Nhận nội dung thực tập và lên kế hoạch thực hiện dự án

a. Mục tiêu đề án

Sinh viên nhận nội dung thực tập và tự đánh giá quy mô, tính chất của dự án có phù hợp với năng lực và yêu cầu thực tập từ phía nhà trường không, đồng thời sinh viên tự ước lượng và đề xuất bảng chi tiết công việc kèm thời gian dự kiến hoàn thành để quản lý có thể theo dõi được tiến độ thực hiện dự án của sinh viên.

b. Nội dung công việc

Anh quản lý Nguyễn Phi Hùng đã cung cấp cho sinh viên nội dung và các yêu cầu về tính năng của chatbot. Cụ thể:

- Nội dung cần làm:

- Tạo API để nhận thông tin từ user, phân tích nội dung tin nhắn và trả về kết quả là các thông tin để thực thi phương thức tìm việc phù hợp;
- Tạo UI chatbot cho trang Pasona.vn;
- Lưu các thông tin khách hàng nhập vào lúc tìm kiếm việc làm;
- Tạo một trang dashboard để hiển thị các thông tin phân tích dữ liệu mà khách hàng đã nhập để tìm việc.

- Nội dung chi tiết:

- Tạo API để:
 - Sử dụng thư viện Flask để tạo 1 RESTful API server trả về các kết quả;
 - Sử dụng các thư viện ngôn ngữ tự nhiên để trích xuất dữ liệu từ nội dung tin nhắn của user theo các thông tin input từ màn hình tìm việc và trả về kết quả dữ liệu đã được lọc;
 - Lưu lại tin nhắn và các thông tin trích xuất từ nội dung tin nhắn của khách hàng.
- Tạo UI chatbot cho trang Pasona.vn:
 - Sử dụng php/HTML/CSS/Javascript để tạo UI chatbot;

- Nhận thông tin từ user nhập vào chatbot và gửi về back-end.
- Lưu thông tin khách hàng nhập lúc tìm kiếm việc làm:
 - Tạo trang tìm kiếm việc làm, lưu lại các thông tin user đã nhập vào khi thực thi tìm việc.

Sinh viên được giao nhiệm vụ phác thảo luồng hoạt động của Chatbot dựa trên các tính năng anh Hùng đưa ra, xác định các ngôn ngữ, thư viện/framework sẽ sử dụng, và liệt kê các bảng dữ liệu sẽ sử dụng cho Chatbot.

- Đồng thời, anh Hùng cung cấp mẫu bảng ước lượng công việc (Estimation) và yêu cầu sinh viên liệt kê chi tiết các đầu việc, bao gồm các cột nội dung:
 - Tên công việc
 - Các đầu việc chính
 - Chi tiết các bước dự định sẽ làm
 - Thời gian dự kiến bắt đầu
 - Thời gian dự kiến kết thúc

Sau khi sinh viên đã hoàn thành bảng, anh Hùng sẽ duyệt qua và góp ý, điều chỉnh bảng cho thực tế, khả thi hơn.

Trước tiên, sinh viên sẽ đề xuất luồng hoạt động của chatbot:

“Khi khách hàng truy cập vào trang chủ Pasona (Pasona.vn), ở góc phải cuối màn hình, biểu tượng chatbot sẽ xuất hiện với lời nhắn “Ask me”. Khi khách hàng bấm vào biểu tượng chatbot, hệ thống sẽ ghi nhận lại thời gian khung chat được mở ra là thời gian bắt đầu cuộc trò chuyện. Mở đầu cuộc trò chuyện, phía Pasona sẽ tự động hiển thị lời chào “Hello! My name is Na, who is willing to solve your questions about jobs. First off, by what name could Na call you?” Khách hàng sẽ nhập tên vào khung nhập tin nhắn, và nhấn phím Enter/nút Send để gửi tin nhắn. Hệ thống sẽ xử lý, làm sạch tin nhắn khách hàng vừa gửi, và lưu lại tên khách hàng theo dạng Session và các tin nhắn từ phía Pasona sau sẽ gọi đúng tên khách hàng, tạo cảm giác thân thiết, tự nhiên hơn. Ví dụ:

- + Tin nhắn khách hàng gửi: i'm thanh_____
- + Tin nhắn sau khi được làm sạch: Thanh
- + Áp dụng: Hi Thanh, you need assistance in finding job vacancies, right?

Tiếp sau đó, chatbot lần lượt hỏi khách hàng về các thông tin liên lạc gồm email và số điện thoại. Sau đó, chatbot sẽ hiển thị câu hỏi “Hi <customer’s name>, you need assistance in finding job vacancies, right?”

Trường hợp khách hàng trả lời “No”: Hệ thống sẽ tiếp tục đặt câu hỏi “So, is there any question <customer’s name> wants to ask Na? If yes, please give us your question.”. Nếu khách hàng trả lời “No”, chatbot sẽ gửi phản hồi “It seems like <customer’s name> has no question. When the question comes up, don’t hesitate to ask Na. Thank you <customer’s name> for your interest in Na and Pasona Tech. If you want to explore job opportunities, you can visit our Job Page via this link: <Job Page link>.”. Khung soạn tin nhắn và nút “Send” sẽ bị vô hiệu hóa, lịch sử đoạn chat này sẽ không được lưu lại trong hệ thống kho dữ liệu. Cuộc hội thoại kết thúc. Nếu khách hàng đưa câu hỏi, mỗi lượt chat sẽ bắt đầu từ lúc khách hàng gửi tin nhắn cho tới khi đoạn hội thoại kết thúc (tùy vào trường hợp cụ thể). Hệ thống tiến hành xử lý, dán nhãn chủ đề tin nhắn và trích xuất các từ khóa trong tin nhắn đó. Hiện tại có ba chủ đề là: “amount”-số lượng công việc đang tuyển (Ví dụ: Can you give me the whole count of job vacancies?), “salary”-tiền lương (Ví dụ: What is the average earnings range for employees in this department?), “trivia”-câu hỏi không liên quan đến chủ đề việc làm (Ví dụ: What's your favorite type of coffee?). Nếu tin nhắn được dán nhãn là “trivia”, hệ thống sẽ phản hồi “What a pity that this request is beyond Na’s knowledge. Na will take it into consideration and try to find the best answer in the future. Thank you <customer’s name> for your interest in Na and Pasona Tech. If you want to explore job opportunities, you can visit our Job Page via this link: <Job Page link>.”. Lượt chat kết thúc. Ngược lại, nếu tin nhắn được dán nhãn khác với “trivia”, các từ khóa trong đoạn tin nhắn sẽ được xác định và dán nhãn tương ứng với các thành phần cấu tạo nên câu query SQL. Ví dụ:

Câu hỏi: what is the average salary of a Software Engineer?

```
#keywords = ["average", "salary", "software engineer"]
```

```
#expected return: {"average": "calculation"; "on field": "salary"; "software engineer":  
"job_position"}
```

```
#query: calculate the average of (max_salary + min_salary)/2 on rows whose  
"job_position" = "software engineer"
```

Sau khi truy vấn kho dữ liệu, hệ thống sẽ trích xuất, định dạng câu phản hồi và gửi lên giao diện chatbot. Lướt chat kết thúc. Chatbot hiển thị lại câu hỏi “So, is there any question <customer’s name> wants to ask Na? If yes, please give us your question.” và tiếp tục luồng xử lý.

Trong trường hợp khi được hỏi “Hi <customer’s name>, you need assistance in finding job vacancies, right?” và khách hàng trả lời “Yes”, hệ thống sẽ lần lượt hỏi khách hàng các thông tin, nhu cầu tìm việc theo các trường dữ liệu: số năm kinh nghiệm, vị trí mong muốn, địa điểm, lương với câu trúc như sau “It would be nice of {customer’s name} to give Na your preference, so that Na could recommend to {customer’s name} best fit job opportunities. If you have no idea what you prefer in that field, please respond ‘I don’t know’. What is your ...?” Khách hàng sẽ trả lời từng câu hỏi, và câu trả lời sẽ được tổng hợp trong một file JSON. Sau khi khách hàng đã trả lời xong các câu hỏi, chatbot sẽ tự động gửi tin nhắn “Na has received your answers. Please take a moment for Na to prepare the list.”. Đồng thời, hệ thống sẽ phân tích dữ liệu tìm kiếm (làm sạch, lọc từ khóa) và tiến hành truy vấn trên các cột dữ liệu tương ứng trong bảng “job”. Ví dụ:
#exp=5 → chọn các công việc có giá trị trên cột “experience” bằng 5.
#location=hochiminh → chọn các công việc có giá trị trên cột “location” là hochiminh

Kết quả truy vấn trả về sẽ ở dạng .json, thông tin mỗi công việc sẽ là một đối tượng (object), chứa các thông tin về ID_job, tên công việc, đường dẫn (link) công việc trên

trang Job Page; các công việc được sắp xếp theo thứ tự giảm dần theo ngày cập nhật. Câu phản hồi cho khách hàng trên giao diện chatbot có định dạng như thế này:

“Here are top 10 job vacancies matching your preference:

1. [job title] – [job ID] – [job link]

...

10. [job title] – [job ID] – [job link]

If you’re interested in a specific job in the above list, you can see details of that job by searching the job ID on Job Page.”

Lượt chat kết thúc. Hệ thống hiển thị lại câu hỏi “So, is there any question <customer’s name> wants to ask Na? If yes, please give us your question.” và tiếp tục luồng xử lý.

Khi mỗi lượt chat kết thúc, hệ thống sẽ lưu lại lịch sử lượt chat đó vào bảng chat_history với các biến sau:

- Question: tin nhắn gốc của khách hàng
- Topic: chủ đề được dán nhãn cho tin nhắn
- Keywords: tin nhắn sau khi được làm sạch
- Response: câu phản hồi của chatbot

Đồng thời, hệ thống sẽ hiển thị phần đánh giá trải nghiệm chatbot của khách hàng theo thang Likert, tương trưng bằng 5 ngôi sao (1: vô cùng không hài lòng, 5: vô cùng hài lòng). Hệ thống cập nhật cột rating với mã khách hàng tương ứng trong bảng thông tin liên lạc của khách hàng cust_info. Khi đã lưu xong đánh giá của người dùng, khung soạn tin nhắn và nút “Send” sẽ bị khóa lại, không sử dụng được nữa. Nếu người dùng muốn bắt đầu lại cuộc trò chuyện, người dùng phải tải lại trang chủ.”

Sau đó, anh Hùng quản lý đã duyệt qua và đưa ra một số góp ý:

- Ở phần thu thập thông tin khách hàng, việc đặt từng câu hỏi như vậy là không hiệu quả, tối ưu, vì khiến đoạn hội thoại bị loãng, dài nhưng chưa vào được trọng tâm yêu cầu của khách hàng, rủi ro khách hàng cảm thấy phiền phức và từ bỏ việc nhắn tin là rất cao. Thay vào đó, sinh viên có thể tạo 1 biểu mẫu cho khách hàng điền vào sẽ vừa đẹp mắt và tránh sự phức tạp, rườm rà cho khách hàng.
- Việc kết thúc chat ngay chỗ câu hỏi được dán nhãn “trivia” chưa được hợp lý, nên cho thêm câu hỏi “bạn còn thắc mắc nào nữa không?” sẽ hợp lý hơn (nên để cho khách hàng là người chủ động kết thúc cuộc hội thoại)
- Phần tìm kiếm công việc có thể tái sử dụng biểu mẫu và API “search-job” trên trang Job Page đã được xây dựng sẵn.
- Phần xác thực thông tin liên lạc của khách hàng hiện chưa có, nên được bổ sung để tránh trường hợp lặp dòng dữ liệu, gây dư thừa bộ nhớ.
- Thay vì hỏi câu hỏi “Yes/No” thì nên đưa ra các lựa chọn để kêu gọi hành động của khách hàng.

Với những lời góp ý của anh Hùng, sinh viên đã nghiên cứu và điều chỉnh, tinh gọn luồng hoạt động của Chatbot như sau:

“Khi khách hàng truy cập vào trang chủ website Pasona.vn, ở phía góc phải dưới màn hình sẽ hiển thị biểu tượng nhấp nháy của Chatbot. Khi khách hàng bấm chọn biểu tượng, hệ thống sẽ hiển thị lời chào và mời khách hàng điền vào biểu mẫu thu thập thông tin liên lạc. Hệ thống sẽ tiến hành xác thực và xử lý dữ liệu: Nếu số điện thoại và email đã tồn tại trong kho dữ liệu của công ty thì sẽ kiểm tra và cập nhật các thông tin khác với thông tin lưu trữ trong bảng dữ liệu thông tin liên lạc, sau cùng hệ thống lấy mã khách hàng và lưu vào local storage. Nếu số điện thoại và email chưa tồn tại trong kho dữ liệu của công ty thì một dòng dữ liệu mới được tạo, lưu trữ thông tin liên lạc của khách hàng mới. Sau đó hệ thống lấy mã khách hàng và lưu vào local storage.

Sau khi điền xong thông tin liên lạc, hệ thống tiếp tục hiện 1 biểu mẫu với format giống như trang Job Page, để khách hàng nhập các yêu cầu tìm kiếm công việc (không bắt buộc điền đầy đủ hết tất cả các trường). Khi này hệ thống sẽ gọi API “search job” trên trang Job Page để thực hiện truy vấn và hiển thị kết quả tìm kiếm (theo giao diện mobile). Đồng thời, hệ thống sẽ lưu trữ các thông tin tìm kiếm công việc vào kho dữ liệu (bảng job_search_history) (nếu khách hàng không tìm kiếm thì sẽ không lưu lịch sử tìm kiếm).

Dưới phần hiển thị kết quả tìm kiếm, hệ thống sẽ hỏi khách hàng còn câu hỏi nào nữa không và đưa ra ba lựa chọn:

- **“Search job”**: tiếp tục tìm kiếm công việc theo mẫu có sẵn. Khi này hệ thống sẽ hiển thị lại mẫu tìm việc cho khách hàng nhập thông tin tìm kiếm.
- **“Chat with chatbot”**: nếu khách hàng có thắc mắc khác tìm việc, thay vì tương tác với nhân viên CSKH thì hệ thống sẽ là người tiếp nhận câu hỏi và trả lời. Khi khách hàng bấm vào lựa chọn này, hệ thống sẽ mở khung soạn tin nhắn và nút gửi tin nhắn. khách hàng nhập nội dung tin nhắn vào khung soạn tin nhắn, hệ thống sẽ xử lý tin nhắn, tương tác với kho dữ liệu và trả về câu phản hồi, cũng như lưu trữ nội dung tin nhắn, các phân tích xử lý của tin nhắn và câu phản hồi. Sau khi đưa ra phản hồi, hệ thống sẽ hỏi lại khách hàng còn câu hỏi nào nữa không kèm ba sự lựa chọn.
- **“End chat”**: nếu khách hàng không còn nhu cầu dùng chatbot thì chọn kết thúc cuộc hội thoại. Khi này hệ thống sẽ vô hiệu hóa khung soạn tin nhắn và nút gửi tin nhắn, đồng thời, hệ thống sẽ mời khách hàng đánh giá trải nghiệm sử dụng chatbot bằng cách đánh giá trên thang điểm 5, được đại diện bởi 5 ngôi sao. Nếu khách hàng chọn ngôi sao nào thì hệ thống sẽ lưu trữ giá trị (điểm) của ngôi sao đó và cập nhật ở cột rating trong bảng liên lạc khách hàng. Cuối cùng sẽ hiện lên lời cảm ơn và cuộc hội thoại sẽ kết thúc.”

Với luồng hoạt động của chatbot như thể này cùng sự tư vấn của anh Hùng, sinh viên sinh viên sẽ xác định các ngôn ngữ, framework và phiên bản sử dụng của chúng. Trong dự án này, phần back-end sinh viên sẽ sử dụng ngôn ngữ mà sinh viên quen thuộc, được giới thiệu và thực hành nhiều trong các môn học chuyên ngành là ngôn ngữ Python, còn về phần front-end, các ngôn ngữ sinh viên sẽ sử dụng là HTML, CSS, Javascript.

Develop languages	Framework / Lib	Version
PHP	Laravel	8.83.25
Python	Flask	2.3.2
	NLTK	3.8.1
	TensorFlow -> TFAutoModel + PhoBERT for Vietnamese (vinai/phobert)	2.13.0
	VnCoreNLP (pre-process raw text)	1.2
HTML		HTML 5
CSS		CSS3
Javascript	Jquery	3.6
Database	Version	
Mysql	5.7.24	
Flask-SQLAlchemy	2.0.19	

Bảng 1. Các ngôn ngữ và framework sẽ sử dụng trong dự án

Về phía cơ sở dữ liệu, ngoài bảng jobs chứa thông tin về các vị trí tuyển dụng đã có sẵn, ta sẽ tạo thêm 3 bảng để lưu thông tin khách hàng sử dụng chatbot, lịch sử tin nhắn chatbot và lịch sử tìm kiếm công việc. (phụ lục kỹ thuật – đồ án 2)

Sau khi luồng hoạt động chatbot được thông qua, sinh viên sẽ tiến hành lập bảng ước lượng tiến độ công việc (Estimation), để anh quản lý có thể theo dõi tiến độ công việc và thuận lợi trong việc hỗ trợ nếu sinh viên gặp khó khăn hoàn thành công việc. Dựa trên mẫu Estimation, sinh viên đã liệt kê các đầu việc để thực hiện dự án chatbot:

No	Primary Item	To Do	Details	Start	End
1	Workflow and expected output				
1.1	Requirement analysis	Apply 5W-1H model to analyze the requirement given by the manager to get the hang of the project and figure out what type of information is confusing or missing -> list questions for manager to clarify the objectives and requirements	<ul style="list-style-type: none"> - What - what will we build - > a chatbot - Who - who mainly uses this chatbot? -> candidates / applicants looking for job - Where - where is this function displayed? -> on website pasona.vn - When - when is this chatbot active? -> 24/7 - Why - why should we build this chatbot? -> consulting applicants to find the jobs with their own preference, collecting the data reflecting how candidates do the job search on our website -> 	8/8	8/10

			<p>understanding candidates behaviors</p> <p>- How - how will the chatbot work? -> interact with customers via message/chat, analyze the text message and respond with information about job</p>		
1.2	Main elements to build the final product	Determine what elements to complete the product (based on the notification from manager)	<p>- API to pull the request from user (message from chatbot)</p> <p>- UI chatbot</p> <p>- Database to store the request history</p> <p>- Dashboard to visualize the operation of the chatbot</p>	8/8	8/10
1.3	Framework, tools, language, environment	Determine what technical tools used to build the elements/product	<p>- API: Flask</p> <p>- UI: php, HTML, CSS, Java Script</p> <p>- Text processing: NLTK, BERT</p> <p>- Database: Flask-SQLAlchemy</p>	8/8	8/10

			- Dashboard: PHP-Laravel, Javascript		
1.4	Scenario planning	Sketch out the workflow and circumstances while interacting with chatbot	<ul style="list-style-type: none"> - Starting from when the user accesses to our website, click on the chatbot button, how the conversation is held - The dataflow between the website and the system 	8/11	8/14
1.5	Technical training/complementary	Self-study unfamiliar technical knowledge required to build the product	<ul style="list-style-type: none"> - Read the document - Become proficiency in the syntax and usage / use cases - Raise question if there are unexpected condition (account, fee,...) from the chosen tools - What to learn: Flask, php, HTML, CSS, Java Script, Docker, dashboard, what includes in the source code to design a function on a website 	8/8	9/20
1.6	Layout collection	Ask leader for the layout of chatbot,	- what type of information the leader/admin wants to	8/9	8/11

		dashboard	view - specific taste: the structure, concept, theme/vibe, media display,...		
1.7	Data schema	How to organize data collected from chatbot activities	- what type of data is needed? - data schema draft - row-level security	8/15	8/15
1.8	Environment installation	Install necessary environments	-python 3 - Node js - PM2 package - Flask package - Run API Server	8/16	8/16
2	API				
2.1	API	build a REST API in Python using the Flask framework	Config API and create routes for main scopes of work	8/16	8/18
2.2	Data modeling	Define a data schema by SQL		8/17	8/18
2.3	Customer information	- Collect data from client		8/19	8/19

	route	<ul style="list-style-type: none"> - Data validation - Save/update cust_info table 			
2.4	Search job route	<ul style="list-style-type: none"> - Collect data from client - Call API from Search Job - Save to jobsearch_history table 		8/20	8/20
2.5	Rating route	<ul style="list-style-type: none"> - Collect data from client - Update cust_info table 		8/21	8/21
2.6	Request process model	Build a model to process input text (message) of customers from the chat bot, labeling the task to direct to next step	<ul style="list-style-type: none"> - label data: content of the request, category of the request ex: I want to get a list of BA manager vacancies -> categories: job list recommendation what is the average salary of dev fresher -> categories: salary overview - data preprocessing steps -> result: extract fields to 	8/22	9/22

			<p>collect data to respond to customer</p> <p>- choose appropriate model to train and test</p> <p>- evaluation metrics to choose the final model</p> <p>=> big Q: where can we get the dataset for training the model? (ask manager/ create manually)</p>		
2.7	Response formatting	Complete template for specific responses to each type of request	<p>- connect with HRBC Porters API</p> <p>- select, process data to fit the template of the given response</p> <p>- send the response to the chatbot and successfully appear on the chatbot interface</p>	9/23	9/28
2.6	Model updating (reinforcement learning)	Update the chat history key value to the dataset training request processing model		9/28	9/30

2.8	Test and fix	Test how well the chatbot work	consider: latency, accuracy, format, data storage	9/28	9/30
3	Chatbot UI		Design a chatbot interface for users to input and analyze information, then provide relevant search results.		
3.1	Learning HTML, CSS, Javascript			9/21	9/24
3.2	UI design	Design the UI template for chatbot	Design a chatbot interface for users to input information.	9/24	10/2
3.3	Call API	Send the information to be analyzed to the back-end and receive the returned data to display on the chatbot	Send requests to the back-end API to process user-entered information and get response content. Perform job searches based on the returned API content.	10/2	10/6
3.4	Test and fix bug			10/8	10/10

Bảng 2. Bảng ước lượng công việc (Estimation)

c. Những kết quả đạt được

Sinh viên thiết kế, định hình được luồng làm việc, hoạt động của chatbot, cũng như xác định được các ngôn ngữ lập trình, framework cần sử dụng để xây dựng chatbot, nắm được các loại ngôn ngữ cần học thêm để từ đó lên bảng kế hoạch làm việc một cách chi tiết, thực tế và hiệu quả nhất. Ở bảng ước lượng đầu tiên, các công việc tự học chưa được thêm vào, dẫn đến bảng kế hoạch được nhận xét là thiếu tính khả thi và thực tế, các công việc bị dồn nhiều vào cùng 1 thời điểm và điều này là không cần thiết, nên ước lượng 1 cách rộng rãi vừa phải để tránh trường hợp làm không kịp và xin dòi hạn chót – 1 điểm không tốt trong làm việc nhóm vì sẽ ảnh hưởng đến tiến độ công việc cả tập thể.

3. Đề án 3: Tổ chức source code theo mô hình thiết kế MVC và tạo API cho chatbot

a. Mục tiêu đề án

Sinh viên làm quen với mô hình thiết kế MVC khi lập trình để dễ quản lý, điều chỉnh source code hơn, đồng thời tìm hiểu sâu vào web API vào biết cách tạo, gọi API để thực hiện tác vụ/truyền dữ liệu. Sau đó, sinh viên xác định luồng hoạt động của chatbot và các công cụ, ngôn ngữ, framework sẽ sử dụng, và áp dụng các kiến thức đã được giới thiệu về API để tạo các định tuyến (routes) cho các tác vụ của chatbot.

b. Nội dung công việc

Sau khi được quản lý giới thiệu mô hình MVC và dành thời gian tự tìm hiểu, sinh viên đã thiết kế cấu trúc source code và đây là luồng chạy giữa các file.

- app.py: thiết lập các biến môi trường, kết nối với database và đăng ký blueprint cho các route
- .env: chứa các thông tin về khóa, tài khoản của các ứng dụng thứ 3 liên kết với chương trình
- Trong folder routes, file chat.py: chỉ định các hàm, phương thức thực hiện với từng endpoint.
- Trong folder controller, file chat_controller.py: định nghĩa các hàm được gọi ở phần routes, trong mỗi hàm sẽ định nghĩa các loại dữ liệu input và các chức năng (service) sẽ được áp dụng để xử lý các dữ liệu input đó.
- Trong folder services, mỗi file tương ứng với từng giai đoạn trong quy trình chatbot, trong mỗi file sẽ định nghĩa các object và các hàm thực hiện những thao tác cụ thể (thêm mới, cập nhật, xóa bỏ...)
- Trong folder model, mỗi file tương ứng với mỗi bảng dữ liệu sẽ được sử dụng trong quy trình chatbot, trong mỗi file sẽ định nghĩa các cột của từng bảng và kiểu dữ liệu của từng trường dữ liệu

- Trong folder utils, các file sẽ định nghĩa cấu trúc (format) của dữ liệu đầu vào/đầu ra
- Folder fine-tuning_model chứa các file fine-tune (điều chỉnh) mô hình học sâu xử lý ngôn ngữ tự nhiên NLP, các file trong folder này sẽ không chạy chung với toàn chương trình chatbot mà sẽ được chạy riêng vào 1 thời điểm nhất định để cải thiện mô hình.
- Luồng chạy: áp dụng mô hình MVC

Routes → controllers → utils → services → models → controllers → utils → routes

Để xây dựng API cho hệ thống, ta sẽ sử dụng thư viện Flask. Đầu tiên, ta tải Flask bằng câu lệnh “pip install flask”. Ở file app.py, ta cài đặt các thư viện cần thiết gồm flask, blueprint, loadenv.

Sau khi cài đặt cấu hình các môi trường sẽ sử dụng (các bên thứ 3 nếu có sử dụng như database-MySQL, AWS,...), ta sẽ khai báo biến app sử dụng Flask() và đăng ký blueprint cho app. Một blueprint trong Flask là một cấu trúc luận lý đại diện cho một phần của ứng dụng. Một blueprint có thể bao gồm các thành phần như là định tuyến (route), hàm hiển thị, form, template và các file tĩnh. Các thành phần bên trong của một blueprint sẽ không được kích hoạt cho đến khi nó được đăng ký với ứng dụng. Trong quá trình đăng ký, tất cả các thành phần bên trong blueprint sẽ được truyền vào ứng dụng. Các định tuyến chat_bp đc khai báo trong file ‘./routes/chat.py’ sẽ được đăng ký blueprint.

Ở file chat.py trong folder ‘routes’, ta sẽ định nghĩa các endpoint (định tuyến) cho các tác vụ của chatbot, bao gồm:

- ‘/userInfo’: lưu/cập nhật thông tin liên lạc của khách hàng từ biểu mẫu trong chatbot
- ‘/search’: lưu lịch sử tìm kiếm công việc từ trang Job Page và biểu mẫu trong chatbot

- ‘/chat’: xử lý tin nhắn gửi từ khách hàng, tìm kiếm câu trả lời và lưu các phân tích lời nhắn vào lịch sử chat
- ‘/rating_chatbot’: lưu đánh giá của khách hàng về trải nghiệm với chatbot.

c. Những kết quả đạt được

Thông qua đồ án 3, sinh viên được làm quen với 1 mô hình tổ chức source code mới là MVC, khiến cho cấu trúc source code trở nên gọn gàng, rõ ràng, dễ hiểu và quản lý hơn. Từ đó, sinh viên áp dụng một cách linh hoạt mô hình này để tự xây dựng cấu trúc source code của dự án chatbot. Sinh viên còn được làm quen với khái niệm RESTful API và gói Flask, từ đó sinh viên tự thiết lập, đăng ký được API cho chatbot và định nghĩa được các endpoint cho từng giai đoạn trong cuộc hội thoại của chatbot.

4. Đề án 4: Tiền xử lý văn bản từ tin nhắn tự do của khách hàng

a. Mục tiêu đề án

Sinh viên áp dụng các kiến thức về xử lý ngôn ngữ tự nhiên (NLP) đã được học ở trường để xây dựng hàm xử lý, chuyển đổi từ tin nhắn gốc của khách hàng được gửi về từ chatbot thành 1 cụm các từ quan trọng của tin nhắn đó để làm dữ liệu đầu vào áp dụng mô hình NLP học sâu cho việc xác định chủ đề tin nhắn cũng như các điều kiện tìm kiếm của câu truy vấn SQL.

b. Nội dung công việc

Để xử lý tin nhắn tự do của khách hàng và có thể xác định các yếu tố cấu thành câu truy vấn đúng với yêu cầu của khách hàng, việc xử lý, làm sạch tin nhắn, loại bỏ các từ dư thừa, sửa chính tả, xác định vai trò của các từ trong câu rất quan trọng. Và toàn bộ phần làm sạch văn bản này sẽ được lưu thành 1 file trong thư mục service. Ngôn ngữ lập trình chính là Python, ngôn ngữ tự nhiên được xử lý là tiếng Anh, và các thư viện được sử dụng gồm:

- re: Regex (Regular Expression), hay còn được hiểu là biểu thức chính quy, là các mẫu kí tự đại diện, thay vì các chuỗi cụ thể, được dùng để tìm kiếm/thay thế văn bản.
- emoji: tập hợp các biểu tượng, kí tự cảm xúc, và một số hàm để tìm, thay thế, loại bỏ... các biểu tượng cảm xúc.
- nltk: đây là một thư viện xử lý ngôn ngữ tự nhiên rất phổ biến trên Python, đa dạng các thao tác xử lý như:
 - + sent_tokenize, word_tokenize: tách 1 đoạn văn bản thành từng câu
 - + stopwords: chứa danh sách từ dừng (stopwords) phổ biến. Các từ này thường không mang nhiều ý nghĩa trong việc phân tích ngôn ngữ, ví dụ như "and", "the", "is",...; loại bỏ các từ này để tập trung vào các từ quan trọng hơn.

- + WordNetLemmatizer: chuyển các từ về dạng cơ bản, giúp giảm sự biến thể của từ (ví dụ: "better" thành "good").
- + Stemming: cắt bớt phần cuối của từ (ví dụ: "running" thành "run")
- + pos_tag: cung cấp các công cụ để gán nhãn cho mỗi từ trong câu dựa trên vai trò ngữ pháp của từ đó (ví dụ: danh từ, động từ, tính từ, v.v.).
- spaCy: Named Entity Recognition (Nhận diện thực thể): SpaCy có khả năng nhận diện các thực thể được đặt tên như tên riêng, địa danh, tổ chức, v.v.
- autocorrect: hàm Speller() của thư viện này sẽ sửa những từ bị sai chính tả thành từ đúng.

(ChatGPT, 2021)

Từng bước xử lý tin nhắn sẽ được định nghĩa trong hàm tương ứng, và sẽ có một hàm `clean_data()` tập hợp tất cả các bước xử lý và trả về tin nhắn đã được xử lý và các thư viện phân tích từ ngữ cần thiết.

Đầu tiên, ta sẽ dùng hàm `sent_tokenize()` để tách từng câu trong đoạn tin nhắn và xử lý từng câu. Ta sẽ xây dựng hàm `replace()`, thực hiện các bước sau:

- Thay thế các thẻ viết tắt của động từ thành từ đầy đủ ("`s`:" is", "`t`:" not", "`m`:" am", "`ll`:" will") qua hàm `.sub()` của thư viện `regex`.
- Loại bỏ các biểu tượng cảm xúc thông qua hàm `replace_emoji()` của thư viện `emoji`
- Loại bỏ các ký tự không liên quan (đường dẫn URL, tài khoản @username, các ký tự khác chữ cái và số qua hàm `.sub()` của thư viện `regex`.
- Thay thế các từ viết tắt tên các địa danh phổ biến thành tên đầy đủ bằng cách định nghĩa 1 từ điển với khóa là từ viết tắt, giá trị là từ gốc đầy đủ; duyệt từng cặp khóa-giá trị trong từ điển trên, nếu trong câu có từ viết tắt, tạo tạm thời 2 biến `before` và `after` để chứa nội dung trước và sau từ viết tắt, và cập nhật chuỗi cũ bằng cách ghép chuỗi của biến `before` và giá trị tương ứng với từ viết tắt trong từ điển và chuỗi của biến `after`.

- Loại trừ các ký tự bị lặp (mỗi ký tự trong 1 từ không được xuất hiện quá 2 lần) (helloooooooooo → hello) (tuy từ chỉnh sửa có thể vẫn bị sai chính tả nhưng việc khác 1-2 ký tự thì hàm vẫn phát hiện và chỉnh sửa được, trong trường hợp từ gốc có quá nhiều ký tự dư thừa bị lặp thì hệ thống không xác định được từ đó bị sai chính tả mà có thể xem từ đó là những từ đặc biệt).
- Loại bỏ các khoảng trống dư thừa và lưu câu đã xử lý vào 1 danh sách (list), nếu câu sau khi xử lý không còn từ gì thì sẽ bỏ qua câu này.

Kết quả trả về của hàm `replace()` là danh sách các câu đã được làm sạch 1 phần.

Sau đó, ta xây dựng hàm `grammar_vocab()` để phân tích sâu hơn vai trò, vị trí ngữ pháp của các từ trong câu và sửa từ sai chính tả.

- Sử dụng hàm `nlp()` của thư viện `spaCy` để xác định các thực thể trong câu, ở đây ta sẽ tìm tên các địa danh (GPE) và thời gian (DATE). Các từ này sẽ được lưu vào thư viện tương ứng, và ở trong câu, các từ này sẽ tạm thời được thay thế bằng “location/date + <số thứ tự>”. Lý do cho việc tạm thời thay thế này vì khi dùng hàm `Speller()` của thư viện `autocorrect` để sửa chính tả, hàm không hiểu một số địa danh do sự khác biệt về ngôn ngữ nên sẽ chỉnh thành từ khác.
- Một số từ ngữ viết tắt các vị trí, ngành nghề cũng sẽ được xử lý giống như các từ địa danh.
- Hàm `Speller()` được sử dụng để sửa các từ sai chính tả.
- Sau khi câu đã được sửa chính tả, các từ chỉ nghề nghiệp sẽ được trả lại trong câu.
- Hàm `grammar_vocab()` sẽ trả về 1 object chứa tin nhắn đã được xử lý, từ điển chứa các từ chỉ địa danh, thời gian, số thứ tự của địa điểm và thời gian.

Trong hàm `clean_data()`, từng câu trong danh sách các câu đã được xử lý sẽ được. Danh sách các câu tin nhắn sẽ được cập nhật, thay câu cũ thành câu đã được xử lý bởi

hàm `grammar_vocab()`. Từ điển các từ địa danh và thời gian được thêm mới, số thứ tự của địa danh và thời gian được cập nhật. Ví dụ:

- + `Old_cleantext = ['what is the average salari in Ho Chi Minh', 'tell me the highest salary range of IT in Hanoi in July']`
- + Sau khi xử lý câu 1 là “what is the average salary in Ho Chi Minh”, kết quả của hàm `grammar_vocab()`:

```
{‘data’: ‘what is the average salary in Ho Chi Minh’,  
‘ner_dict’: [‘location1’: ‘Ho Chi Minh’],  
‘location_count’: 1,  
‘date_count’:0}
```
- + Sau khi xử lý câu 2 là “what is the average salary in Ho Chi Minh”, kết quả của hàm `grammar_vocab()`:

```
{  
    ‘data’: ‘tell me the highest salary range of IT in Hanoi in July’,  
    ‘ner_dict’: [‘location1’: ‘Ho Chi Minh’,  
                 ‘location2’: ‘Hanoi’,  
                 ‘date1’: ‘July’],  
    ‘location_count’: 2,  
    ‘date_count’:1  
}
```

Bước xử lý kế tiếp là xác định vai trò ngữ pháp của từng từ trong câu, được định nghĩa trong hàm `POSTagger()`, và chọn lọc giữ lại các từ là danh từ, tính từ, giới từ. Kết quả trả về của hàm là từ điển với khóa (key) là câu tin nhắn sau khi được xử lý, giá trị (value) là từ điển chứa các từ và thẻ (tag) chỉ định loại từ. Trong hàm `clean_data()`, danh sách các câu tin nhắn sẽ được cập nhật bởi danh sách các khóa của kết quả hàm `POSTagger()`, từ điển các loại từ (`POSTag`) sẽ là giá trị của kết quả hàm `POSTagger()`.

Tiếp theo, ta sẽ loại bỏ các từ dừng (stopwords) và chuyển từ về thể gốc của nó bằng hàm `vocab()`. Cụ thể, ta sẽ điều chỉnh danh sách các từ dừng được cung cấp bởi hàm `stopword()`, loại bỏ các từ 'each', 'and', 'of', 'now', vì các từ này sẽ ảnh hưởng đến việc xác định câu truy vấn SQL (each: nhận biết liệt kê danh sách, now: cho biết thời gian khách hàng muốn biết thông tin là hiện tại, and: cho biết khách hàng muốn xem các loại thông tin nào / yêu cầu của khách hàng có nhiều hơn 1 điều kiện...). Sau khi đã điều chỉnh xong danh sách, ta tiến hành loại bỏ các từ dừng trong câu và cập nhật danh sách các câu tin nhắn. Tiếp đó, ta chuyển các từ trong câu về thể gốc của nó (running → run, cities → city). Kết quả hàm trả về là danh sách các câu tin nhắn được xử lý. Trong hàm `clean_data()`, danh sách các câu tin nhắn được cập nhật bởi kết quả trả về của hàm `vocab()`. Từ điển chứa các loại từ (POStag) được cập nhật: các khóa (từ trong câu) cũ sẽ được thay thế bằng các từ mới, các từ dừng (stopwords) sẽ bị loại khỏi danh sách. Hàm `clean_data()` kết thúc và trả về 1 object chứa câu tin nhắn được xử lý, từ điển NER và từ điển POStag.

Trong file `chat_controller`, ở hàm `chat()`, sau khi trích xuất tin nhắn, hàm `clean_data()` được gọi, với tham số đầu vào là tin nhắn gốc của khách hàng. Kết quả trả về của hàm có 3 thành phần: câu tin nhắn đã được làm sạch, từ điển NER, từ điển POStag. 3 thành phần này sẽ được lưu vào 3 biến tương ứng.

c. Những kết quả đạt được

Sinh viên xây dựng thành công một quy trình xử lý, làm sạch tin nhắn gốc được gửi từ khách hàng thành 1 cụm các từ quan trọng, và gán nhãn cho một số từ đặc biệt (địa điểm, thời gian, loại từ). Với văn bản đã được làm sạch, việc huấn luyện, sử dụng mô hình học sâu sẽ đem lại kết quả tốt hơn.

5. Đề án 5: Ứng dụng mô hình học sâu để xác định chủ đề tin nhắn

a. Mục tiêu đề án

Sinh viên áp dụng các kiến thức, hiểu biết về các dạng bài toán phân loại: phân lớp (classification), phân cụm (clustering), các thư viện, cách vận hành của một số mô hình học sâu phổ biến cho xử lý ngôn ngữ tự nhiên, các chủ đề nội dung tin nhắn sẽ xuất hiện/khách hàng có thể quan tâm để định nghĩa 1 hàm dán nhãn chủ đề cho tin nhắn sau khi đã được làm sạch, ứng dụng ít nhất một mô hình học sâu để thực hiện phân loại chủ đề tin nhắn. Qua các lần thử nghiệm sinh viên biết cách xác định được yêu cầu dữ liệu huấn luyện, điều chỉnh dữ liệu, tham số để tăng độ chính xác của mô hình.

b. Nội dung công việc

Sau khi đã xử lý, làm sạch tin nhắn gốc và trích lọc, giữ lại các từ khóa của tin nhắn, sinh viên sẽ thực hiện việc phân loại chủ đề tin nhắn để điều hướng đến những bước xử lý khác nhau tương ứng với từng chủ đề. Hiện tại anh Hùng quản lý đưa ra cho sinh viên hai chủ đề chính là số lượng công việc (amount) và tiền lương (salary), và sinh viên thêm 1 chủ đề nữa là các câu hỏi ngoài lề, không liên quan đến tuyển dụng, việc làm (trivia). Việc thêm chủ đề trivia này sẽ cắt giảm quá trình xử lý, trích lọc, tạo thành câu truy vấn: khi hệ thống nhận biết tin nhắn thuộc chủ đề trivia, hệ thống sẽ chuyển hẳn sang bước lưu tin nhắn và tự động hồi đáp “Sorry, your request is beyond our knowledge. We are in the progress of expanding our intelligence. Thanks for your resource.”. Trường hợp tin nhắn được phân loại là amount hoặc salary, hệ thống sẽ tiến hành phân tích sâu hơn các từ ngữ để xác định được các đối tượng và vai trò của chúng trong câu truy vấn SQL.

Đây là một bài toán phân lớp dữ liệu (classification), vì tập dữ liệu huấn luyện đã được dán nhãn sẵn, ta sẽ áp dụng một mô hình học sâu để phân tích, “học hỏi” đặc điểm của các lớp (class) từ tập dữ liệu huấn luyện, từ đó có thể xác định, phân lớp cho các dữ liệu kiểm tra.

Vì công ty chưa có sẵn dữ liệu lịch sử về những câu hỏi khách hàng thường đặt ra khi đến trang web của công ty, nên ở lần thử đầu tiên, sinh viên muốn tận dụng các hàm, thư viện đã có sẵn dữ liệu để tiết kiệm được thời gian soạn dữ liệu huấn luyện và nghiên cứu, điều chỉnh (fine-tune) các mô hình học sâu. Sinh viên chọn hàm `synset()` của thư viện `wordnet` để phát hiện các từ đồng nghĩa với từ chủ đề (`amount/salary`). Tuy nhiên danh sách từ đồng nghĩa của thư viện này không quá đa dạng, và có nhiều từ/cụm từ con người có thể hiểu chúng cùng nói về một chủ đề, tuy nhiên với hàm `synset()`, hiểu biết của chúng bị hạn hẹp và không thể hiểu các từ/cụm từ này cùng nghĩa với nhau.

Nhận thấy các thư viện NLTK có sẵn sẽ không đáp ứng được cụ thể nhu cầu xử lý của chatbot, sinh viên sẽ tiến hành lập tập dữ liệu huấn luyện gồm các câu hỏi có thể được gửi về từ khách hàng khi sử dụng chatbot và dán nhãn loại chủ đề cho các câu hỏi, và điều chỉnh (fine-tune) một mô hình học sâu đã được huấn luyện trước (`pre-trained deep-learning model`) để tiến hành phân lớp, xác định chủ đề tin nhắn.

Mẫu dữ liệu huấn luyện sẽ gồm 322 tin nhắn, với 111 tin nhắn với chủ đề ‘amount’, 111 tin nhắn với chủ đề ‘salary’, 100 tin nhắn với chủ đề ‘trivia’. Trước khi đưa vào huấn luyện, mẫu dữ liệu này sẽ được xử lý, làm sạch bằng hàm `clean_data()` đã được định nghĩa ở công việc 3. Sau khi huấn luyện, sinh viên sử dụng hàm `WordCloud()` trong thư viện `wordcloud` để biểu diễn các từ phổ biến của từng chủ đề, nhằm phát hiện, điều chỉnh dữ liệu trong trường hợp có 2 lớp (chủ đề) có quá nhiều từ phổ biến giống nhau (2 lớp có độ tương đồng cao).

Sau khi đã có mẫu dữ liệu huấn luyện, sinh viên tiến hành tìm hiểu và chọn một mô hình học sâu. Ở đây sinh viên sẽ chọn mô hình BERT.

Với các kiến thức đã được học về Máy học, Trí tuệ nhân tạo và mô hình học sâu, sinh viên tiến hành tinh chỉnh mô hình BERT để phù hợp với bài toán phân lớp (xác định chủ đề tin nhắn).

Mỗi mô hình học sâu phục vụ cho một tác vụ sẽ được lưu trong từng file riêng ở folder ‘fine-tuning_model’. Mô hình xác định chủ đề tin nhắn sẽ được lưu trong file ‘train_model_modifier.py’. Trước tiên, ta cài đặt các thư viện cần thiết, gồm torch, transformers, tensorflow qua câu lệnh “pip install torch tensorflow transformers”, và import các module cần thiết như torch, BertForSequenceClassification, BertTokenizer, tensorflow, và 1 số module cần thiết cho việc biểu diễn, đánh giá mô hình.

Với bài toán phân lớp hiện tại, mô hình gốc sẽ được sinh viên tinh chỉnh bằng cách thêm 1 lớp (layer) phân loại ở phía cuối mô hình, và kết quả trả về của mô hình chính là đầu vào của lớp phân loại này. Ta tạo 1 lớp tên UpdatePretrainedModel, trong lớp này ta định nghĩa 1 hàm tên train_and_save_model() gồm các tham số đầu vào:

- new_input_texts: tập hợp các tin nhắn đã được làm sạch
- new_labels: tập hợp các nhãn dán tương ứng với tin nhắn đầu vào
- save_path: tên file chứa mô hình sau khi đã tinh chỉnh
- num_classes: số lớp (số nhãn) trong phần phân phân lớp
- batch_size: số lượng mẫu dữ liệu trong một lần huấn luyện
- learning_rate: tốc độ học là một siêu tham số sử dụng trong việc huấn luyện các mạng nơ ron. Giá trị của nó là một số dương, thường nằm trong khoảng giữa 0 và 1. (trituenhantao, 2019)

Sau đó, ta khai báo tên mô hình ta sẽ sử dụng để tinh chỉnh là ‘bert-base-uncased’, và tải mô hình BERT đã huấn luyện cho việc phân lớp bằng cách gọi hàm BertForSequenceClassification.from_pretrained() với các tham số đầu vào gồm tên mô hình (đã khai báo ở phía trên) và số nhãn bằng số lớp của mẫu dữ liệu huấn luyện. Dữ liệu đầu vào sẽ được xử lý theo hàm tokenizer() của mô hình để biến đổi về dạng có thể đưa vào mô hình. Cả tin nhắn đầu vào đã được token và nhãn dữ liệu (đã được số hóa) sẽ được

chuyển đổi thành PyTorch tensor và kết hợp chúng vào 1 TensorDataset, và 'Dataloader' sẽ đưa TensorDataset vào mô hình, và giải quyết việc chọn số lượng mẫu dữ liệu và xáo trộn dữ liệu trong quá trình huấn luyện.

Tiếp theo, ta định nghĩa hàm mất mát (loss) 'CrossEntropyLoss' và hàm tối đa hóa (optimizer) 'Adam'. Hàm tối đa hóa sẽ cập nhật trọng số của mô hình dựa trên độ dốc (gradient) trong quá trình lan truyền ngược (backpropagation).

- Truyền ngược (back-propagation), là một từ viết tắt cho "backward propagation of errors" tức là "truyền ngược của sai số", là một phương pháp phổ biến để huấn luyện các mạng thần kinh nhân tạo được sử dụng kết hợp với một phương pháp tối ưu hóa như gradient descent. Phương pháp này tính toán gradient của hàm tổn thất với tất cả các trọng số có liên quan trong mạng nơ ron đó. Gradient này được đưa vào phương pháp tối ưu hóa, sử dụng nó để cập nhật các trọng số, để cực tiểu hóa hàm tổn thất. (ChatGPT, 2021)

Sau khi đã chuẩn bị xong các yếu tố đầu vào, vòng lặp huấn luyện sẽ bắt đầu. Ở đây số vòng lặp epoch trong quá trình huấn luyện sẽ là 200, và chỉ số mất mát sẽ được lưu lại mỗi 50 vòng để giám sát mô hình có bị overfitting hay không. Ở đầu epoch, chỉ số mất mát sẽ chỉnh về 0, và mô hình được đưa vào chế độ huấn luyện. Vòng lặp bên trong sẽ được dùng để duyệt qua các batch của mẫu dữ liệu. Với mỗi batch, độ dốc của hàm tối ưu hóa được chỉnh về 0, đầu ra của mô hình và chỉ số mất mát sẽ được tính toán. Sau đó, quá trình lan truyền ngược sẽ được thực hiện, trọng số của hàm tối ưu và tổng mất mát của epoch đó được cập nhật.

Sau khi đã hoàn thành xong việc huấn luyện với 200 epoch, ta dùng hàm .save() của torch để xuất file mô hình theo tên file đã truyền vào qua tham số save_path. Hàm tính chỉnh kết thúc và trả về tên file của mô hình được tính chỉnh.

Ở phần main, sau khi đã làm sạch dữ liệu huấn luyện và lưu dữ liệu mới vào 1 file excel, ta mở file excel đó dưới dạng bảng DataFrame và gọi hàm tinh chỉnh mô hình, truyền các tham số cần thiết vào hàm. Sau khi chạy xong hàm này, trong directory sẽ xuất hiện 1 file .path mới. Đây chính là file chứa mô hình đã được tinh chỉnh.

Bước tiếp theo sau khi đã huấn luyện mô hình thành công chính là đánh giá mô hình (evaluation). Ở đây sinh viên sử dụng ma trận nhầm lẫn (confusion matrix), các điểm số accuracy, precision, recall, F1.

Sinh viên sẽ sử dụng lại mẫu dữ liệu huấn luyện. Sau khi khai báo sử dụng mô hình tinh chỉnh, thực hiện các bước nhúng từ (word sinh viênbeddings) và tiến hành dự đoán nhãn của các tin nhắn. Kết quả dự đoán sẽ được lưu vào 1 cột mới trong bảng dữ liệu.

- Chỉ số accuracy được tính bằng tỷ lệ giữa số lượng dự đoán đúng (true positives) và tổng số mẫu trong tập dữ liệu. Trong lần thử đầu tiên này, độ dự đoán chính xác chỉ ở mức trung bình 66%.
- Ma trận nhầm lẫn (Confusion Matrix) là một công cụ quan trọng trong việc đánh giá hiệu suất của một mô hình phân loại (classification model) trong machine learning. Nó cung cấp cái nhìn toàn diện về cách mô hình dự đoán các nhóm (classes) và cung cấp thông tin về sự phân phối của các dự đoán so với thực tế. Các phần tử nằm trên đường chéo của ma trận thể hiện số quan sát được dự đoán đúng, các phần tử ngoài đường chéo chính chỉ số lượng phần tử bị nhầm sang lớp khác. Với mô hình này, lớp salary bị nhầm lẫn thành lớp amount rất nhiều, chỉ có 1/18 phần tử thuộc lớp salary được dán nhãn đúng, còn lại đều bị dán nhãn amount.

Điều này cho thấy, tập dữ liệu huấn luyện đang có vấn đề, cụ thể 2 lớp amount và salary hiện tại đang khá tương đồng, cần chỉnh sửa lại bộ dữ liệu để phân biệt rõ 2 lớp này hơn. Nhìn lại vào wordcloud của 2 lớp, Có thể thấy, lớp amount và lớp salary có nhiều từ phổ biến tương đồng với nhau (position, job, role), và trong khi ở lớp amount, các từ phổ

biến thể hiện được nội dung chủ đề (open, total, position: số lượng vị trí công việc đang mở), ở lớp salary, các từ phổ biến còn chung chung và chưa thể hiện được nội dung chủ đề (job, role, position,... không liên quan đến lương), vì vậy, dữ liệu của lớp salary cần được thay đổi. Và sau khi thay đổi, các từ phổ biến như average, standard, role,... có liên quan nhiều hơn đến chủ đề lương, nên dữ liệu mới này có thể chấp nhận được.

Khi tiến hành tinh chỉnh lại mô hình lần 2, kết quả đánh giá đã khả quan hơn. Cụ thể, độ dự đoán chính xác đã tăng lên 99,38%, lớp amount và lớp salary không còn bị nhầm lẫn nữa. Tuy lớp trivia có 2 trường hợp bị nhầm thành lớp amount nhưng không đáng kể, bỏ qua được. Mô hình tinh chỉnh lần 2 được chấp nhận và đem vào sử dụng.

Sau khi đã hoàn thành việc tinh chỉnh mô hình học sâu, sinh viên sẽ kết hợp 2 phương pháp xác định chủ đề tin nhắn thành 1 hàm là `topic_identifier()` với 2 tham số đầu vào là từ điển các chủ đề (vì khi tinh chỉnh mô hình, các nhãn đã được mã hóa, và kết quả dự đoán của mô hình trả về kiểu số, nên cần có 1 từ điển định nghĩa chủ đề này được mã hóa thành số nào, để kết quả trả về cuối cùng của tác vụ xác định chủ đề tin nhắn là tên chủ đề) và tin nhắn đã được làm sạch. Đầu tiên, ta sử dụng hàm `synsets()` của thư viện `wordnet` để tìm trong câu có từ nào đồng nghĩa với tên của 1 trong 3 chủ đề không. Trong trường hợp phát hiện có từ đồng nghĩa trong câu thì sẽ thay thế từ đó bằng từ chủ đề và trả về từ chủ đề cùng câu tin nhắn mới chứa từ chủ đề vừa được thay thế. Ngược lại, nếu không tìm thấy từ nào đồng nghĩa với bất kì từ chủ đề nào thì sẽ sử dụng mô hình tinh chỉnh. Hàm `topic_identifier()` trả về 1 object chứa từ chủ đề và câu tin nhắn.

Trong `chat_controller()`, các từ chủ đề được liệt kê trong 1 danh sách, và hàm `topic_identifier()` được gọi để xác định chủ đề tin nhắn. Trong trường hợp từ chủ đề là `trivia`, tin nhắn sẽ được lưu vào kho dữ liệu với lời phản hồi là “sorry”, lượt chat kết thúc. Ngược lại, hệ thống sẽ tiếp tục phân tích và cấu thành câu truy vấn SQL để đưa ra câu phản hồi cho khách hàng.

- Tin nhắn có nội dung không liên quan đến chủ đề tuyển dụng và công việc (trivia): thành công khi tin nhắn được phân loại đúng và đoạn tin nhắn được lưu vào database, lượt chat kết thúc với lời phản hồi “Sorry, the information has not been updated in our intelligence.”.
- Tin nhắn hỏi về lương hoặc về số lượng công việc còn trống: thành công khi tin nhắn được phân loại đúng

c. Những kết quả đạt được

Sinh viên đã tinh chỉnh thành công mô hình học sâu của BERT với bộ dữ liệu huấn luyện do sinh viên tự tạo ra, phát hiện được nguyên do khiến lần đầu tinh chỉnh mô hình trả về kết quả đánh giá thấp và khắc phục được vấn đề, giúp lần tinh chỉnh thứ 2 thành công và đạt được kết quả đánh giá tốt. Khi thử nghiệm với các tin nhắn ngoài bộ dữ liệu huấn luyện, kết quả trả về rất khả quan:

- What's the most fascinating documentary you've watched? → đúng
- i have been rejected a lot. what's the problem? → sai (amount)
- my bf broke up with me today. he cheated on meee. give me a plan for revenge! → đúng
- buy 1 get 1 today at starbucks. dont hesitate. Gooo → đúng
- The Sky is blue. What should I do? → đúng
- Can you provide the mass of open job positions? → đúng
- I'm interested in knowing the quantity of IT job openings in Ho Chi Minh. → đúng
- How many job openings are there in the volume total? → đúng
- Do you have information about the total IT job openings in HCM? → đúng
- What's the complete sum of job opportunities across the organization? → đúng
- What's the entirety of open job positions I can apply for? → đúng
- What is the average wage for someone with my level of experience? → đúng

- What is the average standard wage for employees with advanced experience in this department? → đúng
- Can you provide information about the standard bonus structure for performance-based earnings? → đúng
- Could you provide me with the average salary for a software manager in Ho Chi Minh City? → đúng
- Could you share the aggregate sum of open positions? → đúng
- Can you provide the sum total of open positions? → đúng
- Can you provide the entirety count of open job positions? → đúng
- Can you provide information about the standard bonus structure for employees with intermediate experience in this department? → đúng
- Can you explain the average salary progression for employees in this department? → đúng

6. Đề án 6: Cấu thành câu truy vấn SQL

a. Mục tiêu đề án

Sinh viên phân tích sâu vào ngữ nghĩa của các từ và phân loại từ vào các nhóm cấu thành câu truy vấn SQL, từ đó thiết lập câu truy vấn SQL thu được kết quả như mong muốn. Cuối cùng, sinh viên trích xuất kết quả truy vấn SQL và định dạng thành câu trả lời trả về cho khách hàng trên giao diện chatbot.

b. Nội dung công việc

Sau khi đã phân loại được chủ đề câu hỏi, ta sẽ phân tích các từ trong tin nhắn và phân loại chúng vào nhóm các yếu tố, thành phần của câu truy vấn SQL. Một câu truy vấn SQL sẽ gồm tối thiểu 2 thành phần:

- **SELECT**: chọn các cột, hoặc định dạng kết quả muốn trả về
- **FROM**: thao tác trên bảng nào

Cụ thể, trong tác vụ này, các câu hỏi sẽ liên quan đến thông tin việc làm, tuyển dụng, nên câu truy vấn sẽ thực hiện trên bảng jobs. Ngoài ra, nếu trong câu không nhắc gì tới yếu tố thời gian thì sẽ mặc định lấy dữ liệu của năm hiện tại (nếu tháng hiện tại lớn hơn hoặc bằng 8) hoặc năm trước (nếu tháng hiện tại ít hơn 8), và sẽ chỉ lấy các công việc hiện tại, không lấy các công việc đã xóa.

Các thao tác phân tích, phân loại và hoàn thiện câu truy vấn sẽ được thực hiện trong file ‘services/response_service.py’. Trong phần phân tích các từ và phân loại, ta sẽ xây dựng một hàm label_fields() đi tìm 4 yếu tố:

- **cal_on**: kết quả được tìm trên cột nào / phép tính nào được sử dụng → bổ sung cho phần **SELECT**
- **location**: điều kiện tìm kiếm địa điểm công việc → bổ sung cho phần **SELECT**, **WHERE**

- date: điều kiện tìm kiếm thời điểm tuyển dụng → bổ sung cho phần SELECT, WHERE
- open: xác định xem có cần điều kiện chọn ra những vị trí tuyển dụng đang mở hay không → bổ sung cho phần WHERE

Ngoại trừ chủ đề trivia, ta còn 2 chủ đề chính chưa có bước xử lý là amount và salary. Mỗi chủ đề sẽ có luồng xử lý riêng. Đầu tiên, ta sẽ tạo 1 vòng lặp:

Với từ chủ đề amount: hiện tại chỉ có chủ đề là số lượng công việc, tuy nhiên sinh viên sẽ khái quát cách xử lý. Cụ thể, từ amount chỉ số lượng, nhưng để xác định rõ số lượng của cái gì, thì từ chính cần tìm là từ “something” trong cụm “amount of something”. Ví dụ, “amount of open positions” thì từ chính sẽ là “open positions” hoặc “positions”. Có thể thấy, từ chính sẽ nằm sau từ chủ đề 1 đến 2 vị trí, nên để xác định từ chính với chủ đề này, ta sẽ tạo một vòng lặp duyệt các từ trong câu, và đặt 1 con số khoảng cách: nếu từ đang xét là danh từ, và trước từ đang xét 2 hoặc 3 vị trí là từ chủ đề, thì từ đang xét là từ chính, được lưu vào biến cal_on, và kết thúc vòng xét từ này, chuyển sang vòng xét từ tiếp theo.

Với chủ đề salary: sẽ có nhiều cách tính lương, như lương trung bình, mức lương cao nhất/thấp nhất,... Điểm chung của các từ chỉ phép tính này sẽ là tính từ, và theo cấu trúc ngữ pháp tiếng Anh, thì tính từ sẽ đứng trước danh từ. Ví dụ: “average salary” thì từ chính sẽ là “average”. Để tìm từ chính, ta sẽ tạo 1 vòng lặp duyệt các từ trong câu, và đặt một con số khoảng cách: nếu từ đang xét là tính từ, và sau từ đang xét 1 đến 2 vị trí là từ chủ đề thì từ đang xét là từ chính, được lưu vào biến cal_on, và kết thúc vòng xét từ này, chuyển sang vòng xét từ tiếp theo.

Trong trường hợp từ đang xét không phải là từ chính, thì ta tiếp tục xét đến trường hợp từ này có phải là từ chỉ địa danh (location) hay thời gian (date) hay không:

- Nếu trong từ đang xét có chữ “location” thì danh sách các địa danh sẽ bổ sung tên địa danh bằng cách tìm cặp khóa-giá trị trong từ điển NER đã lập trước đó, và lưu

giá trị của khóa là từ đang xét.

Ví dụ: từ đang xét là “location1”, trong từ điển NER có cặp {“location1”: “Ho Chi Minh”}, như vậy từ được thêm vào danh sách địa danh là [“Ho Chi Minh”].

- Tương tự với trường hợp từ đang xét có chữ “date”, danh sách được cập nhật là danh sách thời gian.

Sau khi vòng lặp kết thúc, nếu danh sách thời gian rỗng, có nghĩa trong tin nhắn không chỉ định điều kiện thời gian cụ thể:

- Với trường hợp chủ đề là amount: biến open sẽ có giá trị là True, có nghĩa sẽ chọn những vị trí đang mở tuyển dụng.
- Với trường hợp chủ đề là salary: biến open sẽ có giá trị là False, có nghĩa không có điều kiện về tình trạng nhận hồ sơ giữa các công việc.

Hàm label_fields() kết thúc và trả về 1 object chứa từ chính cal_on, danh sách địa danh location, danh sách thời gian date, và điều kiện trên trạng thái tuyển dụng.

Sau khi phân tích và xác định các yếu tố, ta sẽ phân tích sâu hơn yếu tố thời gian và biến đổi về kiểu dữ liệu phù hợp. Ví dụ:

- Cụm từ chỉ thời gian là “last month”, ta có thể biết được loại thời gian chính là tháng, và ta sẽ lấy dữ liệu của tháng trước;

Câu truy vấn SQL sẽ như sau:

```
WHERE MONTH(DATE(job_update)) = MONTH(CURDATE()) -1
```

- Hoặc với cụm từ “july 2023”, loại thời gian chính sẽ là tháng, cụ thể là tháng 7, và yếu tố phụ là năm 2023. Từ “july” sẽ được biến đổi thành “7”

Câu truy vấn SQL sẽ như sau:

```
WHERE (MONTH(DATE(job_update))) = 7 AND YEAR(DATE(job_update)) = 2023
```

Hiện tại ta sẽ quan tâm 4 loại thời gian là tuần, tháng, quý, năm. Với tuần và quý, ta có thể dễ dàng nhận dạng vì chúng thể hiện rõ qua các từ “week”, “quarter”, còn tháng và năm thường sẽ không được nêu rõ (“month”, “year”) mà gọi số hoặc tên tháng (“july”, “02”, “2023”), và ta cần chuyển đổi, dán nhãn để hệ thống có thể hiểu ý nghĩa của các từ, ký tự đó. Ta sẽ tạo một lớp `Time_Analyse()` chứa các hàm xử lý yếu tố thời gian. Trong đó, hàm `main_time_component()` sẽ tổng hợp các bước phân tích, và kết quả trả về của hàm là 1 object chứa các thông tin gồm loại thời gian chính, từ trong mẫu tin nhấn thể hiện loại thời gian chính, và danh sách các phần tử thời gian viết theo ngôn ngữ truy vấn SQL.

Đầu tiên, ta phân tích từng từ trong cụm thời gian bằng hàm `identify_date_type()`. Ta tạo 1 danh sách rỗng `result_list` dùng để chứa loại thời gian tương ứng với từng từ. Ta khởi tạo 1 vòng lặp duyệt từng từ trong cụm từ thời gian.

- Nếu từ đang xét là tên của một tháng (danh sách tên các tháng được trích ra từ hàm `.month_name()` của module `calendar`), từ sẽ được chuyển đổi sang dạng số bằng hàm `parser.parse()`, và 1 phần tử mới được thêm vào `result_list` dưới dạng `{"type": "month", "word": date_string, 'converted': parser.parse(date_string).month}`.
- Nếu từ đang xét là một con số, ta dùng hàm `.strftime('%B')` để chuyển dạng số thành dạng chuỗi (tên của tháng). Trong trường hợp hàm chuyển đổi thành công, một phần tử mới sẽ được thêm vào `result_list` dưới dạng `{"type": "month", "word": date_string, 'converted': date_string}`. Ngược lại, nếu hàm chuyển đổi xảy ra lỗi, ta sẽ kiểm tra xem số hiện tại có nằm trong khoảng 2000 đến 2100 không, nếu nằm trong khoảng đó thì số đó sẽ biểu hiện cho năm, và 1 phần tử mới được thêm vào `result_list` dưới dạng `{"type": "year", "word": year}`.

Loại thời gian chính sẽ là phần tử đầu tiên của danh sách `result_list`. Trong trường hợp `result_list` rỗng, loại thời gian chính sẽ rỗng.

Trong hàm `main_time_component()`, ta sẽ kiểm tra trong cụm từ chỉ thời gian có từ nào giống với 1 trong 4 loại thời gian chính không, nếu có sẽ lưu các từ đó vào 1 danh sách, và từ đầu tiên trong danh sách này sẽ là loại thời gian chính, lưu trong biến `maintime`; ngược lại, biến `maintime` sẽ là 1 chuỗi rỗng. Biến `timeword` cho biết từ nào trong cụm từ biểu hiện loại thời gian chính, sẽ có giá trị giống với biến `maintime`. Ví dụ: cụm từ chỉ thời gian có nội dung là “last month”. Sau khi kiểm tra, trong tin nhắn có từ “month”, nên `maintime=“month”`, `timeword = “month”`.

Nếu biến `maintime` rỗng, ta sẽ dùng hàm `identify_date_type()` để xác định `maintime` và `timeword`. Ví dụ: cụm từ chỉ thời gian có nội dung là “july 2023”, sau khi được hàm `identify_date_type()` xử lý, `maintime = “month”`, `timeword = “july”`

Sau khi đã xác định được loại thời gian chính `maintime`, ta dùng hàm `time_interpretion()` chuyển đổi, định dạng điều kiện thời gian theo ngôn ngữ truy vấn SQL cho các phần `SELECT`, `GROUPBY`, `ORDERBY`. Ví dụ: `maintime` là “year”, phần `SELECT` sẽ có thêm “`YEAR(DATE(job_update)) as YEAR_`”, phần `GROUPBY` sẽ có thêm “`YEAR(DATE(job_update))`”, phần `ORDERBY` sẽ có thêm “`YEAR(DATE(job_update)) DESC`”.

Hàm `main_time_component()` kết thúc và trả về 1 object chứa `maintime`, `timeword`, từ điển các điều kiện thời gian cho các phần câu truy vấn SQL, và điều kiện năm.

Quay trở lại với luồng xử lý chính, sau khi đã xác định các yếu tố trong hàm `label_fields()`, ta sẽ hoàn thiện câu truy vấn SQL bằng hàm `query_format()`. Đầu tiên, ta xử lý điều kiện năm. Kết quả xử lý các cụm từ chỉ thời gian sẽ được lưu trong biến `time_component_dict`. Nếu loại thời gian chính của mẫu tin nhắn là năm (“year”) thì điều kiện năm của câu truy vấn là giá trị của `timeword`. Ngược lại, nếu giá trị của khóa “year” trong `time_component_dict` khác rỗng, điều kiện năm của câu truy vấn là giá trị của khóa “year”. Các thành phần của câu truy vấn như `SELECT`, `GROUPBY`, `ORDERBY` sẽ được gán giá trị bởi các khóa tương ứng trong object `time_component_dict`. Nếu trong tin nhắn

không nhắc đến điều kiện năm, ta sẽ xét 2 trường hợp: nếu tháng hiện tại nhỏ hơn tháng 8, năm ta xem xét sẽ là năm ngoái, ngược lại, sẽ là năm nay.

Tiếp theo, ta sẽ định dạng câu điều kiện địa điểm WHERE, tìm kiếm trên 2 cột job_area_name và job_area_summary với từng từ khóa được gán nhãn là địa danh, lưu trong biến “location”. Với điều kiện trạng thái tuyển dụng được lưu trong biến open, các thành phần Having, Select trong câu truy vấn và điều kiện năm sẽ bổ sung thêm bộ lọc trên cột job_phase.

Tổng kết lại, ta đã xử lý và hoàn thiện được các thành phần của câu truy vấn SQL, gồm: SELECT, WHERE (thời gian, trạng thái tuyển dụng, địa điểm). Bước cuối cùng đó chính là ghép các thành phần này lại 1 câu truy vấn hoàn chỉnh tùy theo chủ đề yêu cầu của tin nhắn từ khách hàng đã được xác định bởi mô hình học sâu.

Nếu chủ đề là “amount”: hàm tính toán chính là hàm COUNT() để đếm số lượng phần tử thỏa các điều kiện tìm kiếm. Cấu trúc chung của câu truy vấn cho chủ đề này như sau: 'SELECT {select_add} COUNT({cal_on}) as Count_ FROM jobs WHERE is_hide = 0 AND {year} {location_filter} {groupby_add[:2]} {having} {orderby_add[:2]}'

Nếu chủ đề là “salary”: tùy vào loại lương mà khách hàng yêu cầu tìm kiếm (trung bình, tổng, lớn nhất, nhỏ nhất,...) được lưu trong biến cal_on mà phần SELECT sẽ tương ứng với loại lương đó. Ở đây ta sẽ làm mẫu cho loại lương trung bình: vì trong bảng jobs có 2 cột là job_minsalary và job_maxsalary, 1 số công việc sẽ không cung cấp lương tối thiểu hoặc lương tối đa hoặc cả 2, nên ta sẽ tính trung bình trên từng cột, chỉ lấy các công việc có cả lương tối thiểu và tối đa, sau đó sử dụng hàm CONCAT() để nối chuỗi 2 kết quả trung bình. Cấu trúc chung của câu truy vấn cho chủ đề này như sau: 'SELECT {select_add} CONCAT(ROUND(AVG(job_minsalary)), " - ", ROUND(AVG(job_maxsalary)), " (USD)") AS AVG_ FROM jobs WHERE is_hide = 0


```
AND {year} AND job_minsalary <> 0 AND job_maxsalary <> 0 {location_filter}
{groupby_add[:-2]} {having} {orderby_add[:-2]}
```

(Các từ được tô vàng là các biến dữ liệu đã được phân tích ở phía trên và truyền vào để tạo câu truy vấn hoàn chỉnh theo nhu cầu tìm kiếm của khách hàng.)

Hàm hình thành câu truy vấn SQL sẽ trả về 1 object chứa câu truy vấn, từ chính cal_on, danh sách điều kiện địa điểm và thời gian.

Ta tiếp tục tạo 1 hàm get_reponse() để thực hiện việc truy vấn trên cơ sở dữ liệu và chuyển đổi, định dạng từ kết quả truy vấn thành câu tin nhắn phản hồi sẽ hiển thị cho khách hàng trên giao diện chatbot. Các tham số đầu vào cho hàm này gồm câu truy vấn SQL từ hàm hình thành câu query query_sent() đã được giới thiệu ở trên, từ chính cal_on, chủ đề tin nhắn, danh sách điều kiện địa điểm và thời gian. Kết quả truy vấn SQL được thu thập qua hàm session.execute(text({câu truy vấn})) dưới dạng object.sqlalchemy. Để trích xuất được đúng giá trị cho câu phản hồi, ta chuyển đổi kết quả truy vấn về dạng bảng DataFrame, sau đó tùy vào chủ đề của tin nhắn, ta sẽ chọn cột có tên giống với chủ đề tin nhắn, và lấy giá trị của (các) ô thuộc cột đó (ở đây ta giả sử chỉ trả về 1 kết quả truy vấn, trường hợp groupby trả về nhiều dòng sẽ được nghiên cứu sau). Cấu trúc chung của 1 câu phản hồi sẽ như sau: “The {topic} {location} {date} is {giá trị kết quả truy vấn}.”

c. Những kết quả đạt được

Câu truy vấn SQL đã được hình thành giải đáp được gần 80% yêu cầu của tin nhắn khách hàng. Khi kết nối với cơ sở dữ liệu và thực hiện truy vấn, các câu truy vấn có nghĩa và trả về kết quả truy vấn đúng. Từ kết quả truy vấn, sinh viên đã biến đổi thành câu phản hồi với đầy đủ ngữ pháp, có nhắc lại chủ đề tin nhắn cũng như các điều kiện tìm kiếm về địa điểm, thời gian.

7. Đề án 7: Thiết kế giao diện người dùng cho chatbot và gọi API

a. Mục tiêu đề án

Sinh viên nắm được các kiến thức về lập trình web (HTML, CSS, Javascript, local storage), và với bản thảo thiết kế giao diện của chatbot cùng luồng hoạt động, kịch bản hội thoại của chatbot, các kết quả dự kiến của từng sự kiện khi khách hàng và chatbot tương tác với nhau, các thông tin của từ chatbot cần lưu trữ và cách truyền dữ liệu về backend đã được anh Hùng quản lý duyệt từ đầu, từ đó sinh viên làm quen với việc thiết kế web, nắm được cấu trúc trang web, ngôn ngữ lập trình và thiết kế thành công một sản phẩm. Sinh viên áp dụng các kiến thức được giới thiệu để thiết kế giao diện cho chatbot một cách trực quan, thân thiện với người dùng, các thao tác, nút bấm đơn giản, dễ sử dụng, có luồng hoạt động hợp lý, trôi chảy, tạo cho khách hàng cảm giác gần gũi, tự nhiên như đang nhắn tin với con người. Giao diện này phải liên kết được với hệ thống backend để xử lý tin nhắn và tương tác với database thông qua việc gọi API.

b. Nội dung công việc

Phần thiết kế giao diện chatbot sẽ được viết tiếp trong file thiết kế footer của trang pasona.vn, ngôn ngữ sử dụng chính là HTML, CSS và Javascript.

Phần đầu (header) của chatbot gồm có tên chatbot ở bên trái và 2 nút thu nhỏ, đóng cửa sổ ở phía tay phải. Trong phần thân của chatbot, đầu cuộc hội thoại sẽ là lời chào và biểu mẫu điền thông tin liên lạc của khách hàng được chứa trong cặp thẻ `<form></form>`, bao gồm các trường tên, mail và số điện thoại.

Các trường này đều bắt buộc phải điền đủ, ta sẽ sử dụng thuộc tính `required` trong tag `<input>` cho phép người dùng nhập thông tin, và trong cặp thẻ `<script></script>`, khi khách hàng chọn nút “Submit” và sự kiện được gọi, ta dùng hàm `.checkValidity()` để kiểm tra xem các trường thông tin bắt buộc đã được điền đầy đủ chưa: nếu 1 trường bị điền thiếu thì trình duyệt sẽ không chạy qua bước tiếp theo (nhờ hàm `.preventDefault()` và

.stopPropagation()) mà hiện thông báo yêu cầu khách hàng điền thông tin vào trường bị khuyết. Cho đến khi không còn trường bắt buộc nào bị bỏ trống nữa, hệ thống sẽ lấy giá trị của các ô nhập văn bản và lấy chúng làm tham số đầu vào cho hàm .submitUserInfo(). Hàm này sẽ có nhiệm vụ gọi API xử lý thông tin người dùng từ phía backend thông qua AJAX và cập nhật vào local storage mã khách hàng của khách hàng đang sử dụng chatbot.

AJAX viết tắt của "Asynchronous JavaScript and XML". Đây là một tập hợp các kỹ thuật phát triển web được sử dụng để tạo ra ứng dụng web bất đồng bộ. Một cách đơn giản, AJAX cho phép cập nhật một phần của trang web mà không cần phải tải lại toàn bộ trang. AJAX cho phép tương tác mượt mà và nhanh chóng trong ứng dụng web, vì chỉ có dữ liệu cần thiết được tải hoặc gửi đến máy chủ. Vì chỉ có một phần cụ thể của trang được cập nhật, điều này giúp giảm tải trên máy chủ, dẫn đến hiệu suất tốt hơn.

Sinh viên áp dụng AJAX trong việc gọi API từ phía backend và truyền các dữ liệu từ biểu mẫu. Các tham số cần khai báo gồm: phương thức (POST), đường dẫn (URL có endpoint là /userInfo), dữ liệu truyền về dưới dạng JSON, loại nội dung (contentType) là 'application/json'. Nếu backend xử lý thành công (200), dữ liệu trả về sẽ là mã khách hàng, và mã này sẽ được lưu vào local storage thông qua hàm .setItem(). Ngược lại, nếu việc xử lý có vấn đề, không thành công và trả về trạng thái 500, hệ thống sẽ trả về thông báo lỗi.

Sau khi đã xử lý thông tin của khách hàng, biểu mẫu thông tin liên lạc sẽ biến mất khỏi khung chat, thay thế bởi biểu mẫu tìm kiếm công việc, bằng cách tái sử dụng biểu mẫu tìm việc trên trang Job Page có sẵn theo giao diện mobile. Khi này, hệ thống bắt đầu lưu lại nội dung tin nhắn của cuộc hội thoại vào local storage.

Biểu mẫu tìm kiếm việc làm sẽ có 4 trường là từ khóa, lĩnh vực, mức lương và địa điểm, và không có một trường nào bắt buộc phải điền. Khi khách hàng nhấn nút "Search" ở cuối biểu mẫu, ta sẽ lấy token của biểu mẫu tìm kiếm gần nhất để xác thực yêu cầu. Sau đó, ta trích các giá trị có được từ các lựa chọn tìm kiếm của mỗi trường, để làm tham số

đầu vào cho hàm tìm kiếm, vì ở đây giá trị (value) và nội dung hiển thị trên giao diện (text) là khác nhau: value là mã các lựa chọn được lưu trong kho dữ liệu (ví dụ: “Option:B1872”), còn text là nội dung của các lựa chọn (ví dụ: “Ho Chi Minh”), khi truy vấn trong kho dữ liệu, ta truy vấn bằng mã (value).

Sau khi đã tổng hợp đủ các giá trị tìm kiếm, ta sử dụng .ajax() để thực hiện gọi API và truyền dữ liệu tìm kiếm. Nếu kết quả tìm kiếm khác rỗng, ta sẽ hiển thị các kết quả việc làm theo giao diện mobile, ngược lại, nếu kết quả tìm kiếm rỗng, ta sẽ trả về dòng tin nhắn “There is no job matched your request.”; đồng thời lịch sử tìm kiếm sẽ được lưu lại vào kho dữ liệu bằng cách gọi API có endpoints là “/jobsearch” lưu lịch sử tìm việc đã được sinh viên xây dựng trong phần backend. Giao diện cuộc trò chuyện sẽ được cập nhật vào local storage.

Tiếp theo, chatbot sẽ hỏi khách hàng còn muốn đặt câu hỏi nào không và đưa ra 3 lựa chọn: tìm việc, trò chuyện tự do với chatbot, kết thúc chat. Nếu khách hàng chọn tìm việc (searchjob), ta sẽ hiển thị biểu mẫu tìm việc mới; nếu khách hàng chọn trò chuyện tự do với chatbot, khung nhắn tin và nút “Send” sẽ được kích hoạt; nếu khách hàng chọn kết thúc chat, hệ thống sẽ hiển thị mục đánh giá, đồng thời vô hiệu hóa khung chat và nút “Send”.

Với việc trò chuyện tự do với chatbot, khách hàng nhập tin nhắn vào khung nhắn tin và gửi về hệ thống bằng cách nhấn phím Enter hoặc nút “Send”. Khi này, 1 sự kiện sẽ được gọi và ta sẽ tiếp tục dùng .ajax() để gọi API có endpoints là “/chat” xử lý tin nhắn. Nếu xử lý tin nhắn thành công và có được câu trả lời, chatbot sẽ hiển thị câu trả lời lên giao diện và tự động hỏi khách hàng còn muốn đặt câu hỏi nào không như khi trả kết quả tìm kiếm công việc.

Với việc đánh giá trải nghiệm sử dụng chatbot, chatbot sẽ hiển thị 5 ngôi sao cho khách hàng lựa chọn, theo mức độ từ rất tệ đến rất tốt theo chiều từ trái sang phải. Khi khách hàng chọn 1 ngôi sao, hệ thống gửi tin nhắn cảm ơn, đồng thời thông qua .ajax() gọi API có endpoints là “/rating_chatbot” để cập nhật đánh giá của khách hàng vào kho dữ liệu.

Với 2 nút thu nhỏ khung chat và đóng khung chat, nếu khách hàng chọn nút thu nhỏ, khi mở lên, lịch sử giao diện nhắn tin vẫn được lưu và giữ nguyên; nếu khách hàng chọn nút đóng khung chat, giá trị của nội dung lịch sử cuộc hội thoại và mã khách hàng sẽ bị xóa sạch và trở về dạng rỗng, khi khách hàng mở lại khung chat, 1 cuộc hội thoại mới sẽ bắt lại từ đầu với lời chào vào biểu mẫu thông tin liên lạc.

Khi khách hàng di chuyển sang 1 trang con khác, trình duyệt sẽ kiểm tra lịch sử giao diện cuộc trò chuyện trong local storage: nếu lịch sử rỗng, trình duyệt sẽ xem đây là khởi đầu của 1 cuộc trò chuyện mới, với sự xuất hiện của lời chào và biểu mẫu thông tin liên lạc; nếu lịch sử khác rỗng, trình duyệt sẽ hiển thị lại lịch sử giao diện đó trong khung chat.

c. Những kết quả đạt được

Sinh viên tự thiết kế thành công giao diện người dùng cho chatbot, biết cách dùng hàm .ajax() để truyền dữ liệu từ frontend về backend xử lý thông qua gọi API của backend. Ngoài ra, sinh viên còn biết sử dụng local storage để lưu trữ thông tin, dữ liệu, nhằm cung cấp cho người dùng sự tiện ích khi có thể chuyển trang mà không lo phải bắt đầu chat lại từ đầu, nhất là sau khi thực hiện tìm việc trong chatbot và chọn vào các đường dẫn để tham khảo thêm thông tin công việc.

CHƯƠNG 3: KẾT LUẬN

1. Những kết quả đạt được

a. Lý thuyết

Thông qua dự án chatbot, sinh viên vừa được củng cố, nâng cao khả năng lập trình back-end, áp dụng các kiến thức về xử lý ngôn ngữ tự nhiên, hệ thống cơ sở dữ liệu, phân tích và trực quan hóa dữ liệu; vừa được tiếp xúc và làm quen với các ngôn ngữ lập trình front-end, xây dựng và phát triển các trang web và ứng dụng web. Ba ngôn ngữ front-end chính mà sinh viên được giới thiệu và làm việc nhiều nhất là HTML, CSS và Javascript, cụ thể:

Với HTML (HyperText Markup Language), đây là ngôn ngữ đánh dấu sử dụng để tạo cấu trúc và giao diện của trang web. HTML định nghĩa các phần tử và cấu trúc của trang web như tiêu đề, đoạn văn bản, hình ảnh, liên kết, và nhiều thành phần khác. Sinh viên biết các sử dụng các thẻ (tag) và các thành phần, thuộc tính một cách linh động, sáng tạo, cách thiết lập sự kiện và luồng xử lý khi sự kiện được gọi, các phương thức HTTP và cơ chế hoạt động của từng phương thức. CSS được sử dụng để định dạng và trang trí giao diện của trang web. Nó xác định các quy tắc cho font chữ, màu sắc, khoảng cách, vị trí, và nhiều thuộc tính khác để tạo ra giao diện hấp dẫn và thẩm mỹ. Để thêm tính năng động và tương tác vào trang web, sinh viên sẽ dùng JavaScript, vì JavaScript cho phép lập trình viên thực hiện các hành động như xử lý sự kiện, thay đổi nội dung trang, và giao tiếp với người dùng.

Sinh viên còn được mở rộng kiến thức với Local Storage, một tính năng trong HTML5. Local Storage cung cấp khả năng lưu trữ dữ liệu trên trình duyệt của người dùng. Nó cho phép lưu trữ dữ liệu dưới dạng cặp key-value, tương tự như cách cookies hoạt động. Tuy nhiên, so với cookies, Local Storage cho phép lưu trữ lượng dữ liệu lớn hơn (từ vài MB đến vài chục MB), giúp ứng dụng có khả năng lưu trữ nhiều thông tin

hơn. Dữ liệu được lưu trữ trong Local Storage không được gửi kèm mỗi yêu cầu HTTP, giúp giảm tải mạng và tăng tốc độ tải trang. Vì dữ liệu nằm trên máy của người dùng, không cần phụ thuộc vào server nên nếu người dùng đóng trình duyệt và mở lại, dữ liệu vẫn được giữ nguyên. Đối với chatbot, Local Storage có thể được sử dụng để lưu trữ lịch sử trò chuyện của người dùng để họ có thể xem lại các cuộc trò chuyện trước đó, giữ lại các tùy chọn, cài đặt và lịch sử trò chuyện để cung cấp trải nghiệm người dùng tốt hơn.

Về phần back-end, sinh viên được tạo điều kiện để sử dụng Python làm ngôn ngữ chính cho phần thiết kế RESTful API, xử lý ngôn ngữ tự nhiên, xây dựng mô hình học sâu, tương tác với cơ sở dữ liệu thông qua câu truy vấn SQL. Để sử dụng các phương thức HTTP (như GET, POST, PUT, DELETE) và thực hiện các thao tác trên tài nguyên ở phía server thông qua các URL (Uniform Resource Locator), ta cần thiết kế và triển khai RESTful API. Sinh viên đã nghiên cứu và sử dụng thư viện Flask của Python để thiết kế RESTful API cho chatbot, định nghĩa các định tuyến tương ứng với từng sự kiện trong cuộc hội thoại của chatbot với khách hàng. Sinh viên vận dụng được các kiến thức về xử lý ngôn ngữ tự nhiên được giới thiệu trong môn học cùng tên ở trường để xây dựng 1 quy trình làm sạch tin nhắn của khách hàng gửi về, làm việc với các thư viện NLTK, Regex, Emoji, NER, POSTagger, wordnet ...

Trong quá trình nghiên cứu sản phẩm, sinh viên phát hiện ra bài toán phân lớp trong quá trình xử lý và tạo câu trả lời cho chatbot, từ đó xác định được mô hình học sâu sẽ sử dụng và tinh chỉnh là BERT cho bài toán phân lớp (BERT for classification) và tự xây dựng, đánh giá bộ dữ liệu huấn luyện cũng như độ hiệu quả của mô hình với bài toán hiện tại.

Cuối cùng, sinh viên tương tác và kết nối với cơ sở dữ liệu của công ty qua thư viện SQLAlchemy, sử dụng các dạng câu truy vấn từ đơn giản đến phức tạp để thu thập các thông tin, dữ liệu chuẩn bị cho câu phản hồi của chatbot.

b. Công nghệ, kỹ năng chuyên môn

Về kỹ năng chuyên môn, có 3 kỹ năng của một lập trình viên sinh viên được đào tạo và thực hành trong công việc, đó là quản lý và theo dõi các phiên bản chỉnh sửa của source code qua Git, thiết kế bố cục source code theo mô hình MVC, và tìm-phát hiện-chỉnh sửa lỗi debug qua try-catch.

Đầu tiên, sinh viên học được cách tổ chức, thiết kế bố cục source code một cách gọn gàng, rõ ràng và dễ quản lý, bảo trì hơn thông qua mô hình MVC, thay vì gom tất cả vào một file như trước đây. Mô hình MVC (Model-View-Controller) là một mô hình thiết kế phổ biến được sử dụng trong lập trình phần mềm. Mô hình này giúp tổ chức mã nguồn một cách rõ ràng và phân chia ứng dụng thành ba thành phần chính: Model, View và Controller. (ChatGPT, 2021)

a. Model:

- Chức năng: Model đại diện cho dữ liệu và logic của ứng dụng. Nó quản lý và xử lý thông tin, cũng như định nghĩa các quy tắc kinh doanh và quyền truy cập dữ liệu.
- Ví dụ: Trong một ứng dụng quản lý người dùng, Model có thể bao gồm các lớp hoặc đối tượng như User, Database, và các phương thức như create, read, update, delete (CRUD) để thao tác dữ liệu.

b. View:

- Chức năng: View là phần giao diện người dùng của ứng dụng. Nó hiển thị thông tin từ Model và cập nhật giao diện khi dữ liệu thay đổi. View không nên chứa bất kỳ logic kinh doanh nào.
- Ví dụ: Trong ứng dụng quản lý người dùng, View có thể là giao diện đăng nhập, trang quản lý người dùng, hoặc bảng danh sách người dùng.

c. Controller:

- Chức năng: Controller chịu trách nhiệm điều phối tương tác giữa người dùng, Model và View. Nó nhận lệnh từ người dùng thông qua View, xử lý các yêu cầu, cập nhật Model, và cập nhật View nếu cần thiết.
- Ví dụ: Trong ứng dụng quản lý người dùng, Controller sẽ xử lý các yêu cầu đăng nhập, thêm, sửa, xóa người dùng và cập nhật giao diện người dùng tương ứng.

Mô hình MVC đem lại rất nhiều lợi ích. Đầu tiên, mô hình giúp tách biệt dữ liệu, giao diện và logic điều khiển, giúp dễ dàng quản lý và phát triển mã nguồn; các thành phần của mô hình MVC có thể được sử dụng lại trong các phần khác của ứng dụng hoặc trong các ứng dụng khác. Cấu trúc rõ ràng của mô hình MVC làm cho việc duy trì và mở rộng ứng dụng trở nên dễ dàng hơn. Mô hình này còn cho phép các nhóm phát triển tập trung vào các phần khác nhau của ứng dụng mà không làm ảnh hưởng đến nhau. Cuối cùng, việc tách biệt logic ứng dụng giữa các thành phần khác nhau cho phép nhóm phát triển và thiết kế làm việc độc lập, tăng sự sáng tạo và hiệu suất.

Mô hình MVC đã trở thành một trong những mô hình thiết kế phổ biến nhất trong phát triển ứng dụng và được sử dụng rộng rãi trên nhiều nền tảng và ngôn ngữ lập trình khác nhau.

Trong lĩnh vực phát triển phần mềm, ngoài mô hình MVC (Model-View-Controller) phổ biến, còn tồn tại nhiều mô hình thiết kế khác mang lại cách tiếp cận và quản lý mã nguồn khác nhau. Dưới đây là một số mô hình tương tự mà các nhà phát triển thường sử dụng:

- MVVM (Model-View-ViewModel): MVVM được sử dụng phổ biến trong phát triển ứng dụng di động và desktop. Tương tự như MVC, nó cũng tách biệt các phần khác nhau của ứng dụng nhưng có một lớp trung gian gọi là ViewModel. ViewModel đại diện cho dữ liệu và logic hiển thị, giúp tách rõ ràng giữa dữ liệu và giao diện người dùng.

- MVP (Model-View-Presenter): MVP tách biệt rõ ràng hơn giữa việc xử lý logic ứng dụng và giao diện người dùng. Presenter chịu trách nhiệm điều khiển dữ liệu và giao tiếp với Model, sau đó cập nhật View tương ứng. Điều này thúc đẩy tính kiểm soát và kiểm thử dễ dàng.
- HMVC (Hierarchical Model-View-Controller): HMVC mở rộng ý tưởng của MVC bằng cách cho phép các thành phần con (sub-controllers và sub-views) tồn tại trong các module riêng biệt. Điều này giúp tạo ra các ứng dụng phức tạp và quy mô lớn hơn.
- VIPER (View-Interactor-Presenter-Entity-Router): VIPER được sử dụng phổ biến trong phát triển ứng dụng di động. Nó chia ứng dụng thành năm phần riêng biệt: View (hiển thị), Interactor (xử lý logic), Presenter (điều khiển), Entity (dữ liệu), và Router (điều hướng). Mô hình này tập trung vào sự phân chia trách nhiệm và độc lập của các thành phần.
- Redux: Redux là một kiến trúc quản lý trạng thái phổ biến trong ứng dụng web. Nó sử dụng một nguyên tắc "one-way data flow" và giữ trạng thái ứng dụng trong một trạng thái toàn cục (global state), giúp quản lý trạng thái ứng dụng dễ dàng hơn.

Mỗi mô hình thiết kế đề cập đến cách tiếp cận và tổ chức mã nguồn khác nhau, giúp nhà phát triển tối ưu hóa hiệu suất và quản lý ứng dụng một cách hiệu quả. Sự lựa chọn của mô hình phụ thuộc vào yêu cầu cụ thể của dự án và sở thích của nhà phát triển.

Kỹ năng chuyên môn thứ hai đó là làm việc, trao đổi source code thông qua Git. Công nghệ Git là một hệ thống quản lý phiên bản phân tán (DVCS - Distributed Version Control System) sử dụng để quản lý và theo dõi các phiên bản của mã nguồn. Đây là một công cụ mạnh mẽ và phổ biến trong cộng đồng phát triển phần mềm. (ChatGPT, 2021)

Các khái niệm cơ bản về Git bao gồm:

- Repository (Repo): Nơi lưu trữ toàn bộ dự án, bao gồm các tệp, thư mục và lịch sử thay đổi.

- Commit: Một "snapshot" của trạng thái hiện tại của mã nguồn. Mỗi commit có một mã định danh duy nhất.
- Branch: Một dòng phát triển độc lập. Các thay đổi trên một nhánh không ảnh hưởng đến các nhánh khác.
- Merge: Kết hợp các thay đổi từ một nhánh vào nhánh hiện tại.
- Pull Request (PR): Đề xuất sự thay đổi từ một nhánh (branch) vào một nhánh khác, thường được sử dụng để tích hợp các thay đổi từ các nhóm khác nhau.
- Remote: Một bản sao của kho lưu trữ trên máy chủ từ xa, thường được sử dụng để chia sẻ mã nguồn với các thành viên khác.

Những thao tác cơ bản khi làm việc với Git:

- Clone: Sao chép một repo từ kho lưu trữ từ xa về máy cục bộ để bạn có thể làm việc với nó.
- Commit: Lưu trạng thái hiện tại của mã nguồn vào repo cục bộ.
- Push: Đẩy các commit cục bộ lên repo từ xa.
- Pull: Lấy các thay đổi từ kho lưu trữ từ xa và áp dụng chúng vào mã nguồn cục bộ.
- Branching: Tạo và chuyển đổi giữa các nhánh khác nhau để phát triển độc lập.
- Merging: Kết hợp các thay đổi từ một nhánh vào nhánh hiện tại.
- Pull Request (PR): Đề xuất sự thay đổi từ một nhánh vào nhánh khác và yêu cầu xem xét và chấp nhận.

Kỹ năng chuyên môn thứ 3 đó là gỡ lỗi (debug), giúp lập trình viên tìm các lỗi trong một ứng dụng bằng các lệnh bên ngoài, do đó không có thay đổi nào đối với code thông qua hàm `breakpoint()` của module `PDB` - một module tích hợp bên trong Python (không cần phải cài đặt từ nguồn bên ngoài) và module `traceback` để hiển thị tên lỗi và vị trí lỗi. Khi Python gặp **`breakpoint()`**, nó sẽ tạm dừng chương trình và mở trình gỡ lỗi. Một khi chương trình dừng lại ở **`breakpoint()`**, sử dụng một trong các lệnh sau để kiểm tra và điều chỉnh chương trình:

- **c** (continue): Tiếp tục chạy cho đến khi gặp breakpoint tiếp theo.
- **n** (next): Thực thi đến khi gặp một dòng lệnh mới, sau đó dừng lại.
- **s** (step): Thực thi một dòng lệnh và vào bên trong các hàm được gọi (nếu có).
- **q** (quit): Thoát khỏi trình gỡ lỗi.
- **p <biến>** (print): In giá trị của biến.
- **l** (list): Hiển thị mã nguồn xung quanh vị trí hiện tại.
- **h** (help): Hiển thị trợ giúp về các lệnh trình gỡ lỗi.
- ...

Về công nghệ, RESTful API (Representational State Transfer) là một kiến trúc thiết kế giao diện lập trình ứng dụng (API) phổ biến trong phát triển phần mềm. Nó cung cấp một cách tiếp cận đơn giản và linh hoạt cho việc trao đổi dữ liệu giữa các ứng dụng và hệ thống khác nhau trên mạng. RESTful API không phụ thuộc vào nền tảng hoặc ngôn ngữ cụ thể nào, cho phép các ứng dụng hoạt động cùng nhau một cách hiệu quả. (ChatGPT, 2021)

Các nguyên tắc cơ bản của RESTful API bao gồm:

Một là trạng thái đại diện (State Representation): RESTful API được thiết kế để thể hiện trạng thái của nguồn tài nguyên thông qua các trạng thái đại diện. Trạng thái này có thể là thông tin về tài nguyên, được biểu thị dưới dạng JSON, XML hoặc các định dạng khác.

Hai là phương thức HTTP (HTTP Methods): RESTful API sử dụng các phương thức chuẩn của HTTP (GET, POST, PUT, DELETE) để thực hiện các thao tác khác nhau trên

tài nguyên. Điều này cho phép ứng dụng thực hiện các hành động như truy vấn, tạo mới, cập nhật và xóa dữ liệu.

Ba là các định danh độc lập (Resource Identification): Mỗi tài nguyên trong RESTful API được định danh bằng một URI (Uniform Resource Identifier). Điều này cho phép các ứng dụng truy cập và tương tác với các tài nguyên cụ thể thông qua các URL độc lập.

Bốn là không trạng thái (Stateless): RESTful API không lưu trạng thái của ứng dụng trên máy chủ. Mọi thông tin cần thiết để xử lý một yêu cầu phải được gửi kèm trong yêu cầu đó.

Năm là liên kết tài nguyên (Resource Linking): RESTful API cung cấp các cơ chế cho phép tài nguyên tham chiếu đến các tài nguyên khác. Điều này giúp xây dựng các mối quan hệ giữa các tài nguyên và tạo ra các dự án phức tạp.

Rất nhiều tiện ích mà RESTful API đem lại khiến nó trở nên cực kỳ phổ biến trong thiết kế, trong đó phải kể đến tính linh hoạt và độc lập của RESTful API cho phép các ứng dụng hoạt động độc lập với nhau, giúp tạo ra các hệ thống phân tán và quy mô lớn. Ngoài ra, kiến trúc RESTful cho phép các ứng dụng sử dụng lại các tài nguyên và endpoints, giúp tiết kiệm thời gian và công sức phát triển, đồng thời cung cấp một cách tiếp cận rõ ràng và linh hoạt cho việc quản lý tài nguyên và mở rộng các tính năng của ứng dụng. RESTful API có thể tích hợp một cách dễ dàng với các ứng dụng và dịch vụ khác, giúp mở rộng chức năng của ứng dụng với khả năng hoạt động nhanh chóng và hiệu quả, giúp tối ưu hóa hiệu suất ứng dụng.

RESTful API là một tiêu chuẩn thiết kế giao diện lập trình ứng dụng phổ biến, cung cấp một cách tiếp cận linh hoạt và độc lập với nền tảng. Bằng cách sử dụng các nguyên tắc cơ bản và các phương thức chuẩn của HTTP, RESTful API cho phép các ứng dụng tương tác và trao đổi dữ liệu một cách hiệu quả trên mạng. Điều này giúp tạo ra các ứng dụng linh hoạt, mở rộng và dễ dàng quản lý.

Flask API là một thư viện trong ngôn ngữ lập trình Python, được sử dụng để xây dựng và triển khai các API dễ dàng và hiệu quả. Flask API tận dụng cú pháp rõ ràng và sự linh hoạt của Python để cung cấp một cách nhanh chóng và tiết kiệm thời gian để tạo ra các API mạnh mẽ.

Một điểm mạnh của Flask API là khả năng định nghĩa các endpoints (điểm cuối) một cách dễ dàng. Các endpoints đại diện cho các tài nguyên cụ thể hoặc hành động mà API cung cấp. Flask cung cấp các decorators như “@app.route()” để định nghĩa các endpoints và gắn chúng với các hàm xử lý tương ứng. Flask API cung cấp các tiện ích mạnh mẽ khác nhau để xử lý các yêu cầu HTTP, quản lý trạng thái ứng dụng, và nhiều tính năng khác. Nó cung cấp một cách nhanh chóng và hiệu quả để xây dựng các API đáng tin cậy và mở rộng. Điều này làm cho Flask trở thành một trong những công cụ ưa thích của các nhà phát triển khi tạo ra các dịch vụ web dựa trên Python.

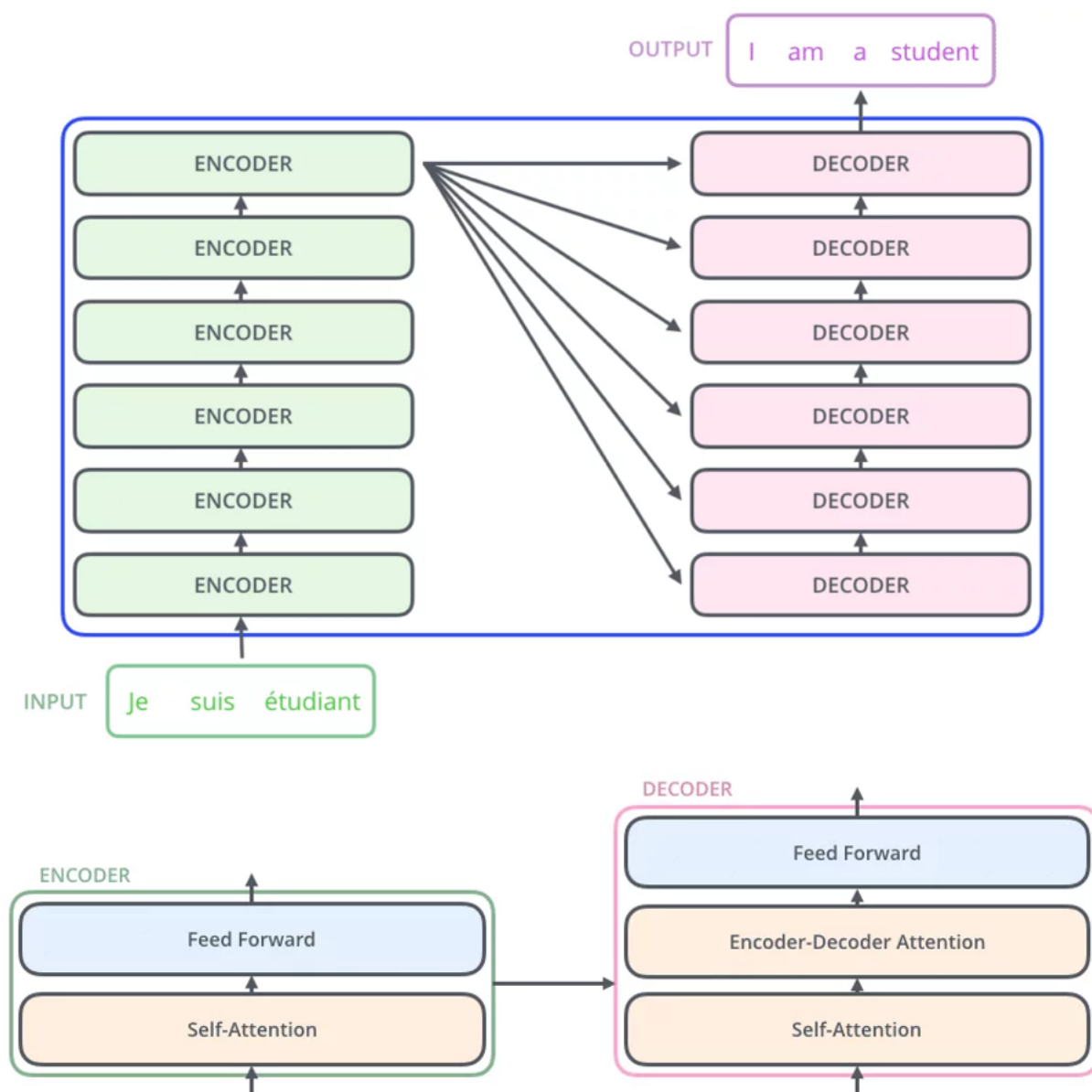
Một công nghệ khác sinh viên được tiếp cận và nghiên cứu đó là mô hình học sâu dành cho ngôn ngữ tự nhiên – BERT. BERT, viết tắt của Bidirectional Encoder Representations from Transformers, là một mô hình tiên tiến trong lĩnh vực Xử lý Ngôn ngữ Tự nhiên (NLP) được phát triển bởi Google. Nó nổi bật trong việc hiểu và xử lý ngôn ngữ nhờ khả năng nắm bắt thông tin ngữ cảnh từ cả hai hướng của một từ cụ thể. Phương pháp tiếp cận song hướng này là điều đặc biệt nổi bật của BERT so với các mô hình tiền nhiệm. (Quang, Viblo, 2018)

Mô hình BERT được dựa trên kiến trúc Transformer, mở đầu với khái niệm về cơ chế tự quan sát. Các cơ chế này cho phép nó đánh giá sự quan trọng của mỗi từ trong câu đối với tất cả các từ khác, từ đó tạo điều kiện cho việc hiểu ngữ cảnh toàn diện.

Kiến trúc Transformer (ChatGPT, 2021): Kiến trúc Transformer là một mô hình học sâu phát triển bởi Google Research, được giới thiệu trong bài báo năm 2017 mang tên

"Attention is All You Need". Đây là một trong những mô hình quan trọng trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP). Kiến trúc Transformer được thiết kế để xử lý các tác vụ NLP phức tạp như dịch máy, nhận diện ngôn ngữ tự nhiên, tóm tắt văn bản và nhiều tác vụ khác. Các đặc điểm quan trọng của kiến trúc Transformer bao gồm:

- Tự chú ý (Self-Attention): Đây là thành phần quan trọng của mô hình. Nó cho phép mô hình tập trung vào các phần quan trọng của đầu vào. Nhờ đó, nó có khả năng hiểu ngữ cảnh từ cả hai hướng và giúp cải thiện hiệu suất so với các mô hình trước đó.
- Đầu vào chuỗi đều nhau: Transformer không yêu cầu đầu vào có cùng độ dài như các mô hình trước. Thay vào đó, nó sử dụng ma trận đầu vào có kích thước tùy ý và sử dụng các cơ chế tự chú ý để xác định tầm quan trọng của từng phần tử.
- Kiến trúc mã hóa-giải mã (Encoder-Decoder): Transformer được sử dụng cho các tác vụ tạo chuỗi như dịch máy. Mô hình được chia thành phần mã hóa và giải mã, mỗi phần sử dụng một tập hợp riêng biệt của các lớp tự chú ý và mạng truyền thông.
- Khả năng song ngữ (Bidirectional): Transformer có khả năng hiểu ngữ cảnh từ cả hai hướng, điều này nghĩa là nó có thể nhìn thấy cả ngữ cảnh trước và sau một từ.
- Không có các tầng ẩn (No Recurrent Layers or Convolutional Layers): Không giống như các mô hình trước đó như RNN và CNN, Transformer không sử dụng các tầng ẩn. Thay vào đó, nó sử dụng tự chú ý.

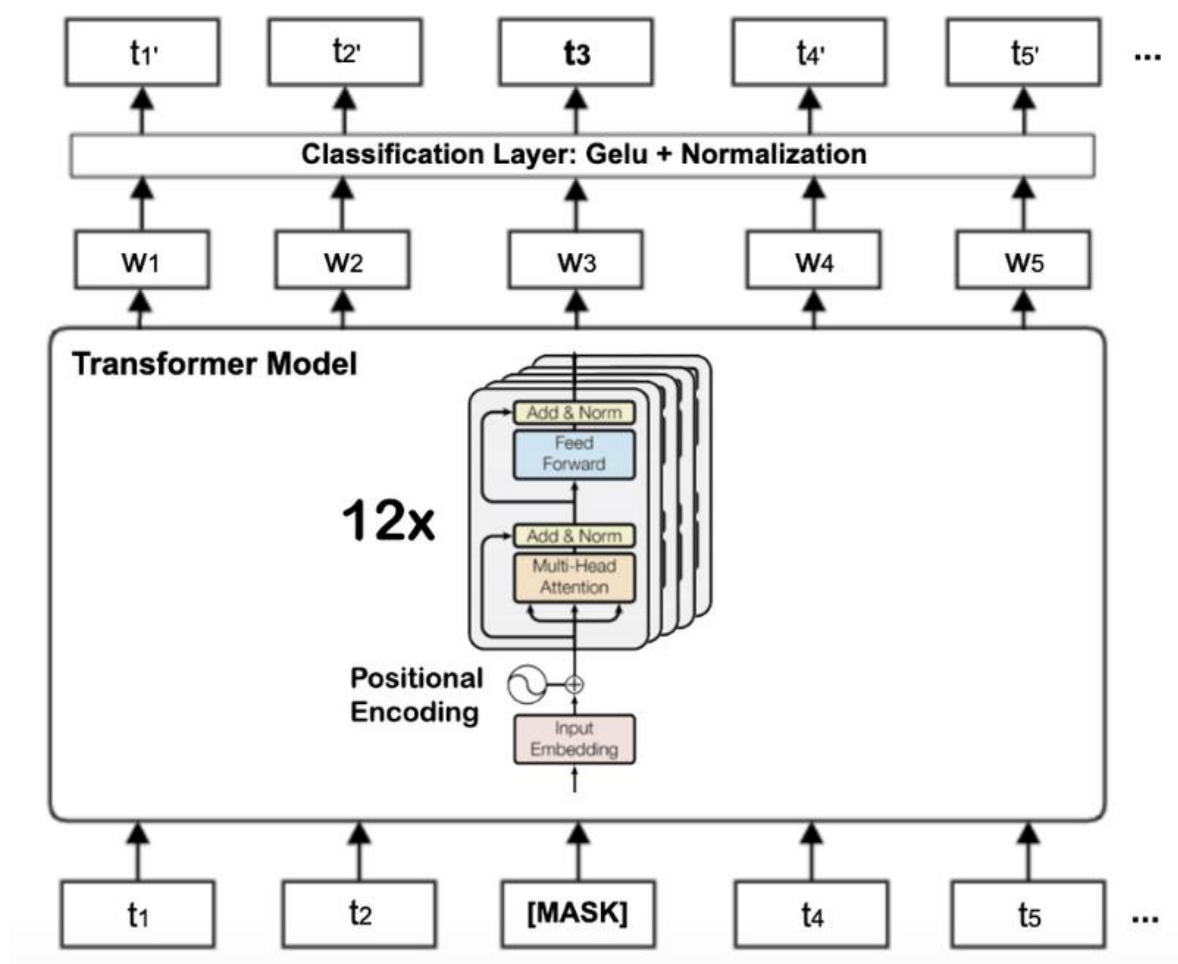


Hình 1. Kiến trúc Transformers

Cơ chế hoạt động của mô hình BERT bao gồm các tác vụ sau:

- Mã hóa từ (Tokenization): Câu hoặc đoạn văn bản đầu vào được chia thành các "token" nhỏ. Mỗi token có thể là một từ, một phần của từ hoặc một ký tự.

- Tạo ma trận nhúng (Embedding Matrix): Mỗi token được biểu diễn dưới dạng một vector số học trong không gian đa chiều. Những vector này tạo thành ma trận nhúng.
- Tạo lớp Self-Attention: Các token trong ma trận nhúng được đưa vào một lớp tự chú ý (self-attention layer). Đây là lớp quan trọng trong kiến trúc Transformer. Nó cho phép mô hình tập trung vào các phần quan trọng của văn bản và đồng thời loại bỏ sự phụ thuộc dài hạn.
- Mạng Feedforward: Ma trận đầu ra từ lớp tự chú ý được đưa vào một mạng feedforward để tạo ra các biểu diễn ngữ cảnh cuối cùng.
- Huấn luyện mô hình: Mô hình BERT được huấn luyện trên một tập dữ liệu lớn với các nhiệm vụ như dự đoán từ tiếp theo, dịch ngôn ngữ hoặc phân loại văn bản.
- Fine-tuning: Sau khi tiền huấn trên một tập dữ liệu lớn, BERT có thể được điều chỉnh cụ thể cho các nhiệm vụ như phân tích tâm trạng, trả lời câu hỏi, nhận diện thực thể đã được gán tên, vv.
- Sử dụng trong ứng dụng thực tế: Một khi đã được tiền huấn và điều chỉnh, BERT có thể được triển khai trong các ứng dụng thực tế như chatbot, tìm kiếm nâng cao, phân loại văn bản và nhiều ứng dụng khác liên quan đến xử lý ngôn ngữ tự nhiên.



Hình 2. Mô hình BERT

Khi so sánh mô hình BERT với các mô hình học sâu khác như Word2Vec, LSTM, GRU, CNN,..., BERT nổi trội hơn ở khía cạnh hiểu bối cảnh 2 chiều, và việc tinh chỉnh (fine-tune) đối với BERT không quá khó khăn, phức tạp, và với tác vụ phức tạp như thiết kế chatbot, BERT được ưu tiên nhất.

Khác với các mô hình truyền thống chỉ xử lý văn bản theo một hướng (từ trái sang phải hoặc ngược lại), BERT có thể hiểu cách từng từ bị ảnh hưởng bởi ngữ cảnh xung quanh toàn bộ. Điều này dẫn đến các biểu diễn ngôn ngữ phong phú hơn, tinh tế hơn, từ đó tạo điều kiện cho các dự đoán chính xác hơn và phù hợp với ngữ cảnh.

Mô hình cơ bản của BERT đã được tiền huấn trên một kho ngữ liệu lớn, cho phép nó nắm vững các sự tinh vi của ngôn ngữ trên nhiều lĩnh vực và ngữ cảnh. Sau đó, nó có thể được điều chỉnh cụ thể cho các nhiệm vụ như phân tích tâm trạng, trả lời câu hỏi và nhận diện thực thể đã được gán tên. Sự linh hoạt này biến BERT thành một công cụ cực kỳ đa năng trong các ứng dụng NLP.

Một trong những điều đột phá của BERT đó là sự quan tâm đặc biệt đến hiện tượng nhiều nghĩa, hoặc tình trạng từ có nhiều ý nghĩa. Bằng cách xem xét ngữ cảnh từ xuất hiện, BERT giỏi trong việc phân biệt ý nghĩa được dự đoán. Đây là một bước tiến lớn trong việc hiểu ngôn ngữ tự nhiên, tương tự như cách con người hiểu ngôn ngữ.

Tóm lại, mô hình cơ bản của BERT đại diện cho một bước tiến trong lĩnh vực NLP. Phương pháp tiếp cận song hướng và khả năng hiểu ngữ cảnh làm cho nó có khả năng hiểu văn bản một cách sâu sắc và chính xác. Với khả năng thích ứng với nhiều nhiệm vụ, BERT đã trở thành một điểm mốc quan trọng trong nghiên cứu và ứng dụng NLP hiện đại, làm thay đổi cách chúng ta tương tác và hiểu ngôn ngữ viết.

c. Kỹ năng mềm

Tuy dự án chatbot là dự án cá nhân, nhưng vì chatbot là một tính năng bổ sung của trang pasona.vn, nên sinh viên cần trao đổi, tìm hiểu về cấu trúc, mục đích, các thông tin và cơ sở dữ liệu từ người thiết kế trang web (anh Hiệp) để xây dựng chatbot phù hợp, tương thích với trang web hiện tại. Vì vậy, các kỹ năng giao tiếp, ghi chú, đặt câu hỏi và dự tính các tiêu chí cho câu trả lời là vô cùng quan trọng, để đảm bảo việc giao tiếp, trao đổi thông tin hiệu quả, tiết kiệm thời gian và đảm bảo chất lượng, tiến độ công việc. Kỹ năng giao tiếp này cũng đặc biệt quan trọng trong các buổi chia sẻ kiến thức chuyên môn của anh Hùng quản lý, vì khối lượng công việc của anh Hùng lớn hơn nhiều so với vị trí thực tập của sinh viên, thế nên để đảm bảo chất lượng các buổi chia sẻ, sinh viên sẽ chủ động hỏi danh sách các chủ đề, nội dung (agenda) anh sẽ chia sẻ, sinh viên dựa vào danh sách đó và tự đọc tài liệu trước ở nhà, ghi chú lại những ý chính cũng như các thắc mắc;

trong buổi chia sẻ, sinh viên bổ sung thêm các kiến thức mở rộng không có trong phần tự học trước đó, và chủ động nêu các thắc mắc cần anh Hùng giải đáp, để đảm bảo sinh viên nắm được các lý thuyết, kỹ năng chuyên môn được chia sẻ, sẵn sàng áp dụng vào dự án cá nhân mà không cần anh Hùng phải dành quá nhiều thời gian để chỉ từng bước một.

Một kỹ năng quan trọng khác sinh viên được hướng dẫn đó là kỹ năng lên kế hoạch công việc. Thay vì để quản lý lên sẵn kế hoạch thực hiện một dự án và các phần việc sẽ tham gia, sinh viên sẽ chủ động nghiên cứu công việc, dựa vào năng lực của bản thân mà ước lượng thời gian hoàn thành công việc, liệu có cần sự trợ giúp của ai không hay có thể tự làm một mình được. Việc tự lên kế hoạch cho công việc của bản thân không chỉ giúp sinh viên trở nên chủ động trong công việc, quản lý thời gian hiệu quả hơn mà thông qua việc tìm hiểu, nghiên cứu dự án và chia nhỏ đầu việc, sinh viên sẽ nắm rõ toàn bộ mục đích, các mốc quan trọng của dự án, khi làm việc nhóm, việc hiểu rõ này sẽ giúp các thành viên trao đổi dễ dàng, thấu hiểu công việc của nhau hơn, thay vì chỉ biết mỗi phần việc của bản thân mà không hiểu rõ dự án tham gia có mục tiêu, ý nghĩa gì.

Cuối mỗi ngày làm việc, sinh viên sẽ nộp lại cho anh Hùng quản lý một bản báo cáo ngắn về các việc đã làm trong hôm đó, cùng tiến độ thực hiện những đầu việc trong bảng kế hoạch đã định ra và dự định ngày tiếp theo sẽ làm gì. Việc báo cáo tiến độ công việc hằng ngày không chỉ giúp quản lý nắm được tình hình thực tập của sinh viên, mà còn tập cho sinh viên có thói quen thống kê lại những công việc đã thực hiện trong ngày, tận dụng tối đa bảng dự tính công việc đã lập ra, và lên kế hoạch, mục tiêu cho ngày tiếp theo.

2. Những thiếu sót, hạn chế cần cải thiện

Về phần kiến thức, chuyên môn, sinh viên cần mở rộng hiểu biết và thành thạo sử dụng thêm các framework thiết kế giao diện người dùng, để trong thời gian tới, sinh viên sẽ tự xây dựng một trang Dashboard cho chatbot và trang tìm việc (Job Page). Trong thực tế, việc có một công cụ, không gian để doanh nghiệp có thể quản lý tình hình hoạt động kinh doanh, khai thác các dữ liệu thu thập được từ các kênh giao tiếp, điểm tiếp xúc với

khách hàng là một điều vô cùng cần thiết và quan trọng, đặc biệt với sản phẩm chatbot, để trợ lý ảo này có thể thực sự giao tiếp một cách tự nhiên, giải đáp được nhiều yêu cầu từ tệp khách hàng của Pasona, việc thu thập lịch sử trò chuyện và hành vi tìm việc của họ trên trang web của công ty sẽ hỗ trợ rất nhiều cho doanh nghiệp trong việc nghiên cứu hành vi, thói quen, sở thích của khách hàng, biết được khách hàng quan tâm nhiều đến vấn đề gì khi khảo sát thị trường việc làm, đâu là những ngành nghề/lĩnh vực/vị trí/địa điểm có nhiều lượt tìm kiếm, quan tâm nhất,... để từ đó doanh nghiệp điều chỉnh các chiến dịch, hoạt động kinh doanh, quảng bá phù hợp với khách hàng mục tiêu, nâng cao sự hài lòng khi trải nghiệm dịch vụ mà trang web cung cấp.

Với ngành học khoa học dữ liệu của sinh viên, Dashboard là một công cụ mạnh mẽ cho phép sinh viên áp dụng được phần lớn các kiến thức đã được học ở trường vào thực tế: thu thập dữ liệu từ nhiều nguồn khác nhau (trang web, cơ sở dữ liệu...) và tiền xử lý chúng bằng cách sử dụng đa dạng ngôn ngữ truy vấn như Python, SQL; trực quan hóa dữ liệu thông qua việc áp dụng một cách đa dạng, hiệu quả các biểu đồ, biểu mẫu; áp dụng các phương pháp phân tích nghiệp vụ để làm việc với đối tượng sử dụng Dashboard nhằm nắm được các loại thông tin họ muốn quan sát, theo dõi để tạo nên 1 Dashboard “biết kể chuyện”: mạch lạc, rõ ràng, luồng thông tin từ bao quát đến chi tiết. Tuy rằng có rất nhiều công cụ, phần mềm hỗ trợ việc xây dựng Dashboard như Tableau, PowerBI, nhưng sẽ có những giới hạn nhất định hoặc phải trả phí để dùng được nhiều tính năng, hoặc có nguy cơ bị tiết lộ thông tin mật khi kết nối bên thứ ba (phần mềm) với cơ sở dữ liệu. Chính vì thế, việc tự thiết kế 1 trang Dashboard sẽ đem lại nhiều lợi ích cho doanh nghiệp hơn là sử dụng ứng dụng bên thứ ba.

Vì thế, trong giai đoạn thực tập, sinh viên sẽ nghiên cứu và tự xây dựng Dashboard mẫu, sau thời gian thực tập, khi dự án chatbot được thông qua, sinh viên sẽ trình bày Dashboard và làm việc tiếp với bộ phận PHR để hoàn thiện sản phẩm.

Anh Hùng quản lý đã giới thiệu cho sinh viên một thư viện cung cấp đa dạng các biểu đồ, đó là Vuetify 2, được xây dựng dành cho Vue.js. Vuetify được xây dựng dựa trên nguyên tắc thiết kế của Google - Material Design, cung cấp một giao diện người dùng sạch sẽ, mịn màng và hấp dẫn. Vuetify cung cấp một loạt các components giao diện người dùng sẵn có như buttons, cards, forms, dialogs, menus, v.v. Điều này giúp tiết kiệm thời gian phát triển và đảm bảo sự nhất quán trong giao diện người dùng. Vuetify 2 đi kèm với các thành phần được thiết kế sẵn để phản ứng với kích thước màn hình khác nhau, đảm bảo rằng ứng dụng của bạn sẽ hoạt động tốt trên các thiết bị từ điện thoại thông minh đến máy tính bảng và máy tính để bàn. Framework hỗ trợ chủ đề và sắc thái, cho phép bạn dễ dàng tùy chỉnh giao diện người dùng của mình để phù hợp với thương hiệu hoặc phong cách cụ thể của bạn. Vuetify cung cấp các components phù hợp với việc thiết kế các bảng điều khiển, bao gồm các components như cards, layouts, grids, charts, và nhiều hơn nữa. Vuetify là một dự án mã nguồn mở có một cộng đồng lớn, do đó bạn có thể tìm kiếm và nhận được sự giúp đỡ từ các nhà phát triển khác trong trường hợp bạn gặp vấn đề hoặc cần hướng dẫn. Dự án Vuetify luôn được cập nhật để cung cấp các tính năng mới, sửa lỗi và cải thiện hiệu suất.

Dashboard sẽ có cấu trúc như sau: 1 trang tổng quát (overview) (trang chủ); 1 trang về hoạt động của chatbot; 1 trang về tình hình sử dụng chức năng tìm việc (jobsearch); 3 trang dữ liệu của lịch sử tin nhắn của chatbot, lịch sử tìm kiếm công việc, thông tin khách hàng; các trang phụ như thông tin user, log in, log out, lỗi.

Với khách hàng mục tiêu phần lớn là người Việt Nam, trên thực tế, chatbot của sinh viên được kỳ vọng sẽ có phần xử lý tin nhắn tiếng Việt. Hiện tại, các thư viện NLP dành riêng cho tiếng Việt đã được nghiên cứu và giới thiệu, nổi bật không thể bỏ qua đó là PhoBERT.

PhoBERT là một mô hình ngôn ngữ tiếng Việt được phát triển bởi các nghiên cứu viên tại Viện Công nghệ Thông tin và Truyền thông (ICT) thuộc Đại học Quốc gia Hà Nội. Nhóm phát triển tập trung vào việc ứng dụng những tiến bộ trong lĩnh vực xử lý ngôn ngữ tự nhiên (NLP) để tạo ra một mô hình ngôn ngữ sâu có khả năng hiểu biết sâu về tiếng Việt. (ChatGPT, 2021)

(Quang, Viblo, 2020) Mô hình PhoBERT được xây dựng dựa trên kiến trúc BERT (Bidirectional Encoder Representations from Transformers), một trong những mô hình học sâu tiên tiến nhất trong lĩnh vực NLP. PhoBERT được huấn luyện trên một tập dữ liệu lớn bao gồm cả Vietnamese Wikipedia corpus và Vietnamese news corpus, đảm bảo rằng mô hình có sự hiểu biết sâu về ngôn ngữ Việt Nam. Bên cạnh đó, PhoBERT cũng sử dụng RDRSegmenter của VnCoreNLP để thực hiện phân tích từ cho dữ liệu đầu vào.

Với sự kết hợp giữa kiến thức về tiếng Việt và các tiến bộ trong công nghệ NLP, nhóm phát triển PhoBERT đã tạo ra một công cụ mạnh mẽ hỗ trợ nhiều ứng dụng trong việc xử lý ngôn ngữ tự nhiên cho tiếng Việt. Những ứng dụng phổ biến nhất của PhoBERT gồm tách từ (tokenization), nhúng từ (embedding), tinh chỉnh (finetune) cho các bài toán phân lớp (classification).

Khác với tiếng Anh, mỗi tiếng thường sẽ là 1 từ độc lập có nghĩa (apple, tree, consume), tiếng Việt có một cấu trúc từ ngữ phong phú với nhiều nguyên âm, dấu thanh và các ký tự đặc biệt khác, 1 từ tiếng Việt thường sẽ có 1 đến 3 tiếng, nếu tách riêng từng tiếng trong 1 từ thì sẽ đem lại những nghĩa khác nhau. Ví dụ, từ “học sinh” là danh từ, chỉ về 1 nhóm đối tượng từ 6-18 tuổi đi học ở trường, nếu tách từng tiếng sẽ có 2 từ “học” và “sinh”, cả 2 từ không còn là danh từ mà là động từ, chỉ về 2 hoạt động khác nhau: “học” chỉ hoạt động tiếp thu kiến thức, “sinh” chỉ hoạt động tạo ra 1 sự vật, sự sống nào đó. Nếu trước đây, ta dùng hàm tokenizer() của thư viện nltk để tách từ tiếng Việt sẽ không thực sự đưa về kết quả chính xác như ta kỳ vọng, nhưng với PhoBERT, PhoBERT Tokenizer sử dụng kiến trúc BPE (Byte-Pair Encoding) để tách các từ thành các subword. Điều này

giúp mô hình có thể hiểu được các từ mới hoặc từ vựng ít phổ biến. Kiến trúc BPE (Byte-Pair Encoding) là một phương pháp nén dữ liệu và mã hóa chuỗi ký tự. Trong ngữ cảnh của PhoBERT Tokenizer, BPE được sử dụng để tách các từ thành các đơn vị nhỏ hơn gọi là "subword". Thông qua việc tách từ thành các subword, mô hình có thể học được các thành phần cơ bản của từ và từ đó, đưa ra dự đoán về các từ mới hoặc từ vựng ít phổ biến.

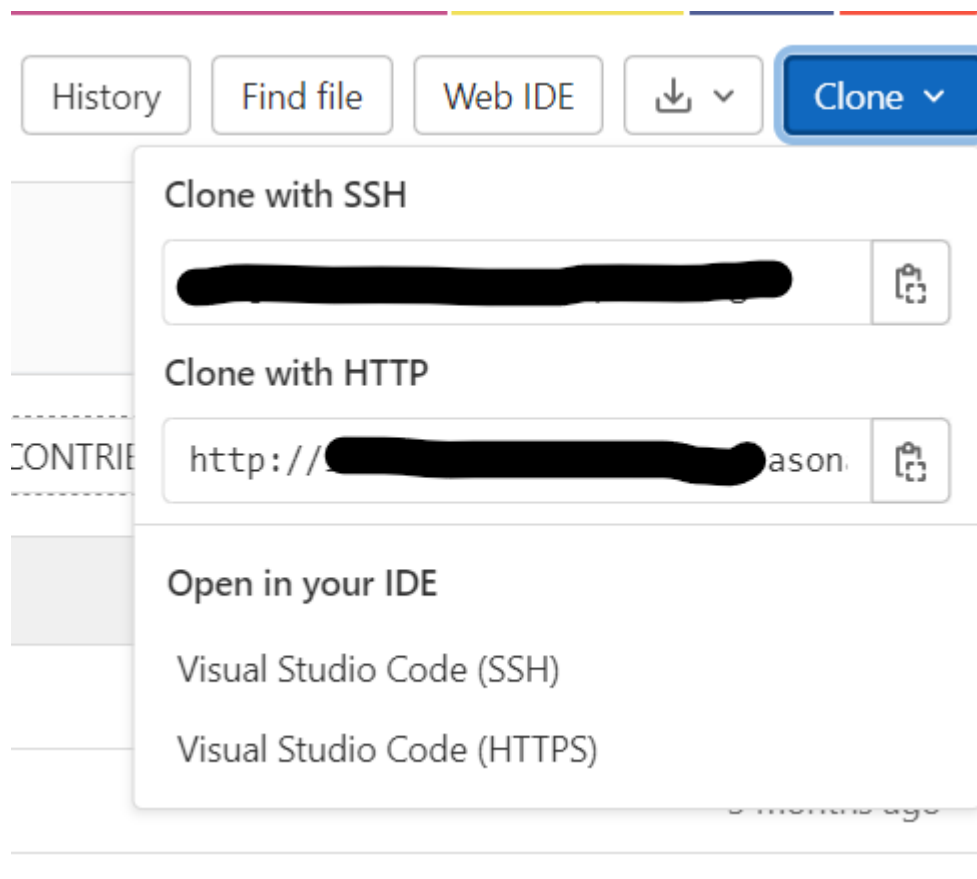
Việc cho phép người dùng tinh chỉnh mô hình (finetune) của PhoBERT khiến mô hình trở nên linh hoạt và dễ sử dụng hơn thông qua việc thay đổi Embedding Layer (lớp đầu tiên của mô hình, biểu diễn các từ dưới dạng vector), Hidden Layers, Output Layer, hàm mất mát (loss function) và thuật toán tối ưu hóa (optimizer). Ví dụ, với bài toán phân lớp (classification), ta thêm một lớp phân loại ở cuối và đầu ra của PhoBERT tinh chỉnh sẽ là đầu vào của lớp phân loại này. Đầu vào của PhoBERT tinh chỉnh cũng phải được thay đổi để phù hợp với nhiệm vụ phân lớp. Theo mô hình PhoBERT, văn bản đầu vào của mô hình phải được chuyển thành chuỗi token và được chèn thêm hai token [CLS] và [SEP]. Trong nhiệm vụ phân lớp, trạng thái ẩn tương ứng với token đặc biệt [CLS] là đại diện của toàn bộ câu được sử dụng cho các nhiệm vụ phân loại, khác với vector trạng thái ẩn tương ứng với token biểu diễn từ thông thường. Như vậy, khi cung cấp một câu đầu vào cho mô hình trong quá trình huấn luyện, đầu ra là một véc-tơ trạng thái ẩn tương ứng với token này. Lớp bổ sung được thêm ở trên bao gồm các nơ ron tuyến tính chưa được huấn luyện có kích thước [kích thước véc-tơ trạng thái ẩn, số ý định], có nghĩa là đầu ra của PhoBERT kết hợp với lớp phân loại là một vector gồm hai số đại diện cho điểm số để làm cơ sở phân loại câu. Vector này được đưa tiếp vào hàm mất mát cross-entropy để tính phân bố xác suất phân loại ý định. Mô hình PhoBERT sau khi tinh chỉnh được lưu thành định dạng chuẩn của Huggingface để đưa vào sử dụng trong ML-Chatbot. Cấu hình là bước cần thiết để các thành phần phối tự động tạo ra một mô hình học máy. Đối với các hệ thống chatbot sử dụng mô hình học máy thì cấu hình này là phần đóng gói toàn bộ các

phương pháp xử lý dữ liệu để tạo ra một mô hình học máy phù hợp nhất cho một bộ dữ liệu cụ thể.

Với dự án chatbot hiện tại, sinh viên dự định sẽ áp dụng mô hình PhoBERT để tách từ, nhúng từ và tinh chỉnh mô hình phân lớp để xác định chủ đề tin nhắn, các từ khóa chính trong tin nhắn và phân loại các từ khóa vào các trường tìm kiếm khác nhau (ngành nghề, địa điểm, công cụ, kỹ năng,...), qua đó có thể đem đến cho khách hàng thêm 1 lựa chọn sử dụng ngôn ngữ tiếng Việt thoải mái, thân thiện hơn.

CHƯƠNG 4: PHỤ LỤC KỸ THUẬT

Đề án 1: Quản lý source code bằng Git



(Giao diện lấy đường dẫn clone trên Gitlab)

```
PS C:\Users\sona_test> git clone http://[redacted]/pasona.git
remote: Enumerating objects: 4864, done.
remote: Counting objects: 100% (188/188), done.
remote: Compressing objects: 100% (107/107), done.
remote: Total 4864 (delta 87), reused 160 (delta 72), pack-reused 4676
Receiving objects: 100% (4864/4864), 823.97 MiB | 2.53 MiB/s, done.
Resolving deltas: 100% (3019/3019), done.
```

(Câu lệnh git clone trong terminal)

```
PS C:\[redacted]\pasona_test> cd pasona
PS C:\[redacted]\pasona_test\pasona> git status
Your branch is up to date with 'origin/develop'.
● nothing to commit, working tree clean
```

(Câu lệnh git status trong terminal)

```
PS C:\[redacted]\pasona_test\pasona> git checkout search-api
Updating files: 100% (69/69), done.
Switched to a new branch 'search-api'
● branch 'search-api' set up to track 'origin/search-api'.
PS C:\[redacted]\pasona_test\pasona> git status
On branch search-api
● Your branch is up to date with 'origin/search-api'.

nothing to commit, working tree clean
```

(Git checkout để di chuyển đến nhánh muốn làm việc)

```
PS C:\[redacted]\pasona> git add search-api/fine-tuning_model/train_model_identifier.py
PS C:\[redacted]\pasona> git status -uall
Refresh index: 100% (512/512), done.
On branch search-api
Your branch is up to date with 'origin/search-api'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   search-api/fine-tuning_model/train_model_identifier.py
```




(Câu lệnh git add để đưa file vào danh sách chờ)

```
PS C:\[redacted]\pasona> git commit -m "create fine-tuning_model folder to store fine-tuning files; add instruction on executing"
● [search-api fbeb960] create fine-tuning_model folder to store fine-tuning files; add instruction on executing topic fine-tuning model file.
  1 file changed, 213 insertions(+)
  create mode 100644 search-api/fine-tuning_model/train_model_identifier.py
○ PS C:\[redacted]\pasona>
```

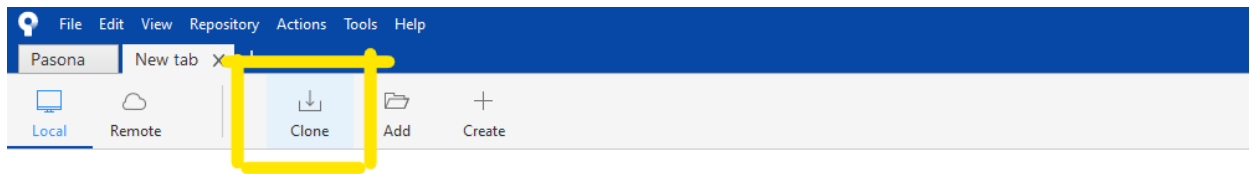
(Câu lệnh git commit để chuyển trạng thái sang staged kèm lời nhắn)

```
PS [redacted] pasona> git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 3.22 KiB | 3.22 MiB/s, done.
Total 5 (delta 2), reused 0 (delta 0), pack-reused 0
remote:
remote: To create a merge request for search-api, visit:
remote: [redacted] /pasona/-/merge_requests/new?merge_request%5Bsource_branch%5D=search-api
remote:
To [redacted]:pasona.git
   93cbdec..fbeb960 search-api -> search-api
PS [redacted] Pasona>
```

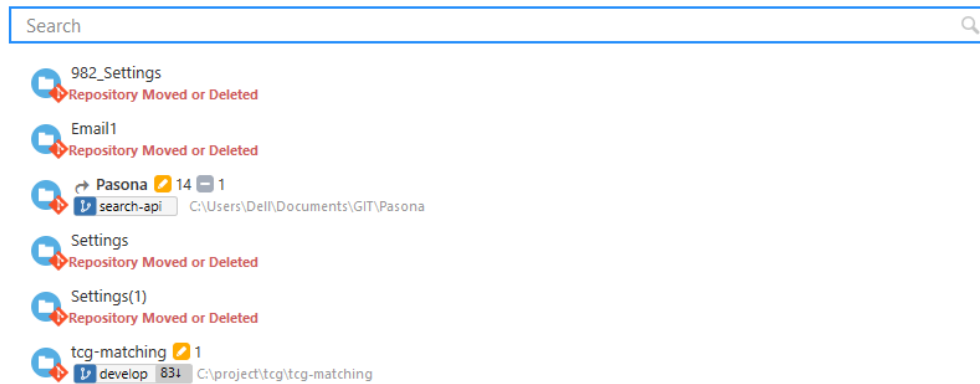
(Câu lệnh git push để đưa code từ local lên remote)

<div> create fine-tuning_model folder to store fine-tuning files; add instruction on... Doan Vu Minh Thanh authored 9 minutes ago</div> <div><input type="checkbox"/> </div>		
Name	Last commit	Last update
..		
 train_model_identifier.py	create fine-tuning_model folder to store fine-tuning files; add instru...	9 minutes ago

(Trạng thái của remote được cập nhật)



Local repositories



(Giao diện khởi động của SourceTree và nút chọn Clone)

Clone

Cloning is even easier if you set up a remote account

Source Path / URL:

Browse

Repository Type: ? No path / URL supplied

Destination Path:

Browse

Name:

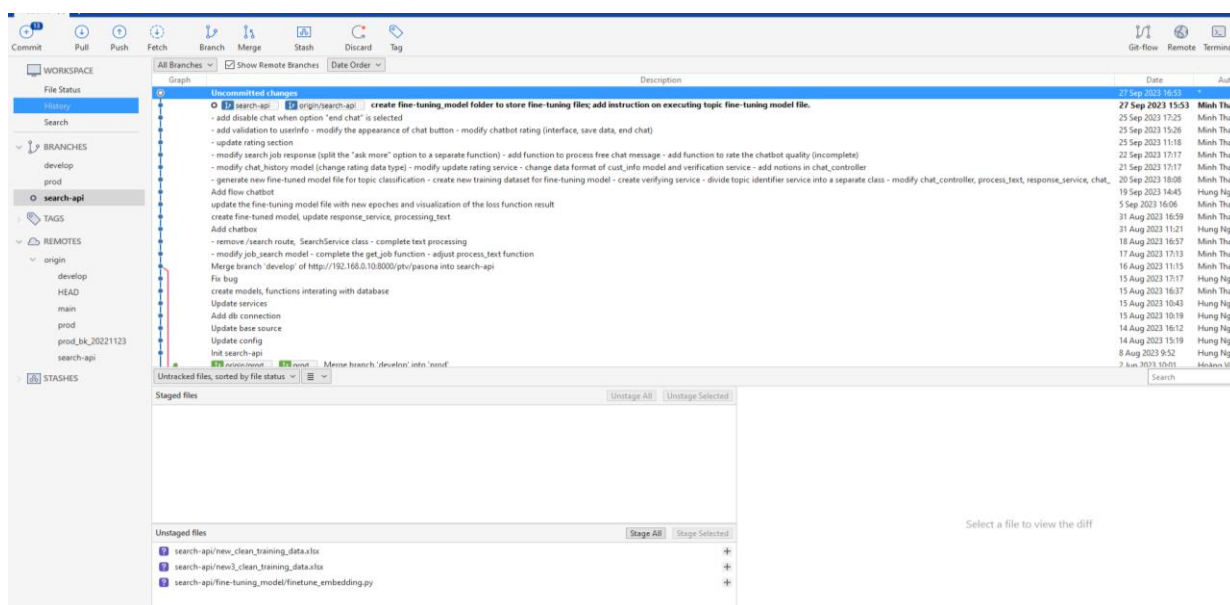
Local Folder:

[Root]

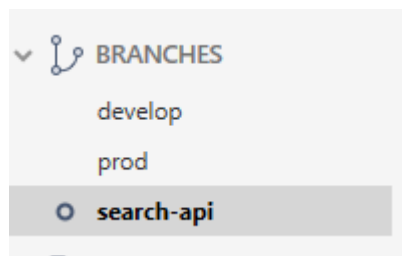
➤ Advanced Options

Clone

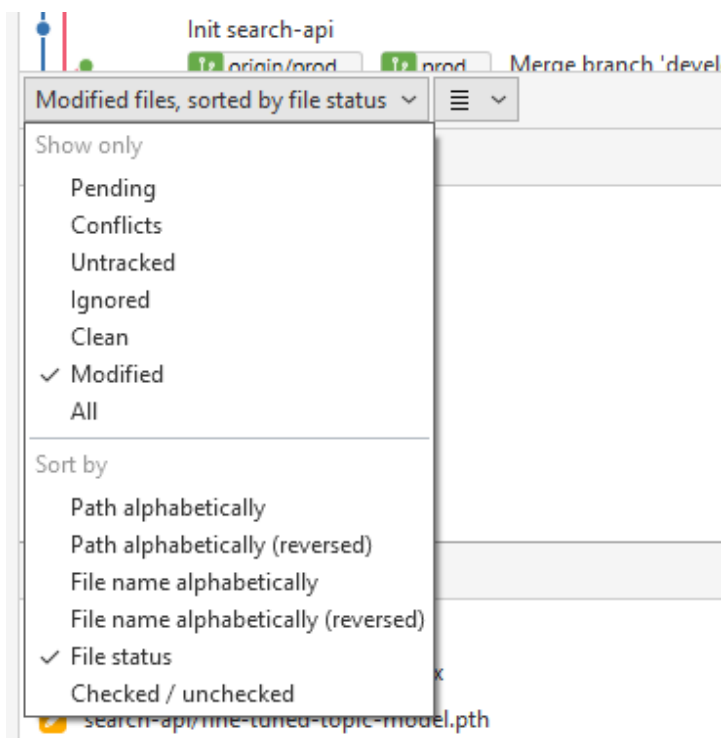
(Giao diện điền thông tin của Clone)



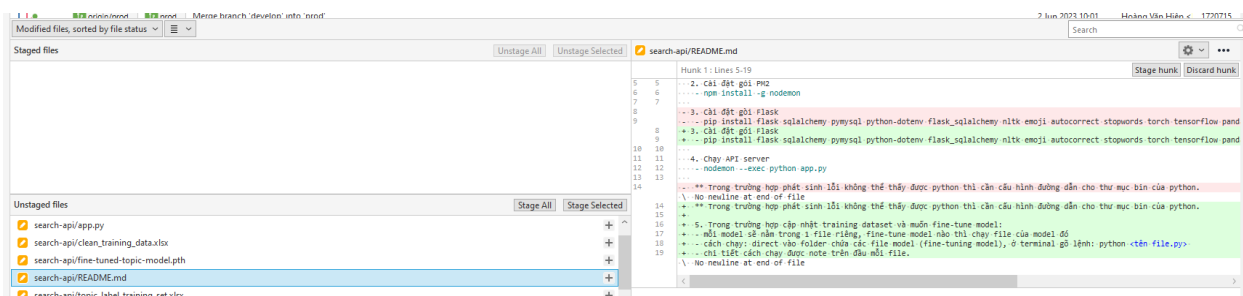
(Giao diện quản lý source code sau khi clone thành công)



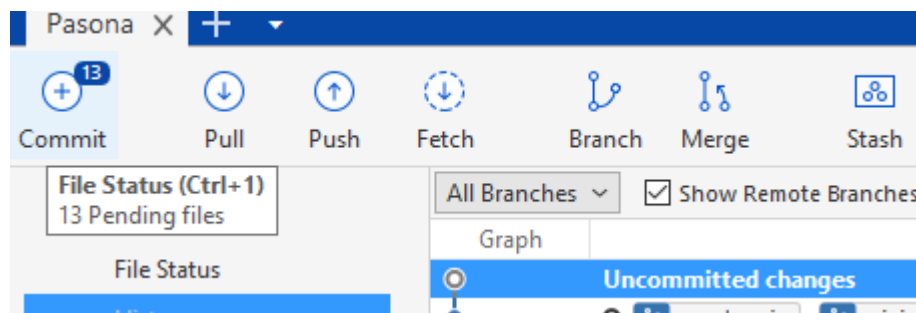
(Chọn đúng branches ở local để làm việc)



(Các trạng thái file trong SourceTree)



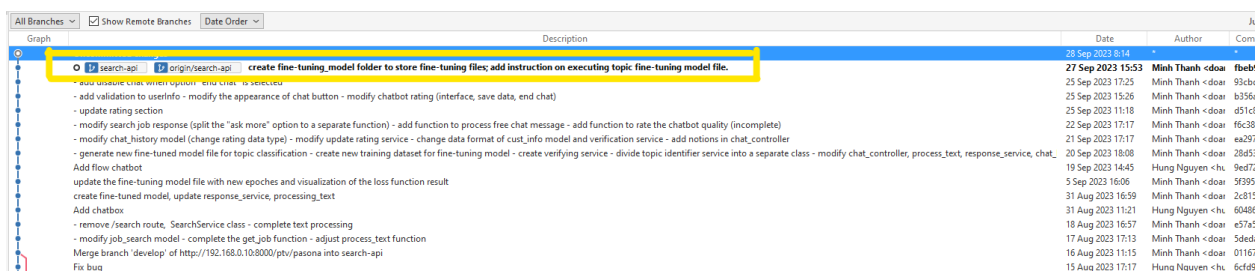
(Giao diện kiểm tra các thay đổi trong file và đẩy file sang trạng thái staged)



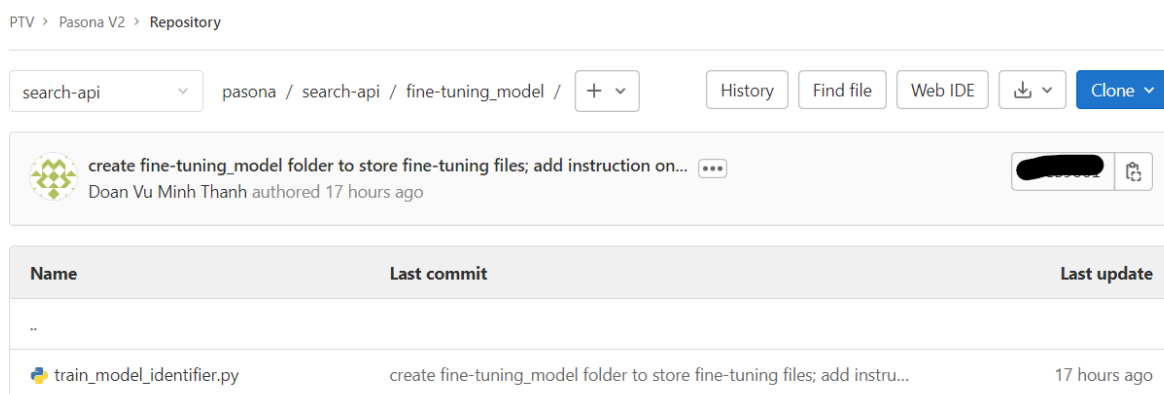
(Nút chọn commit)



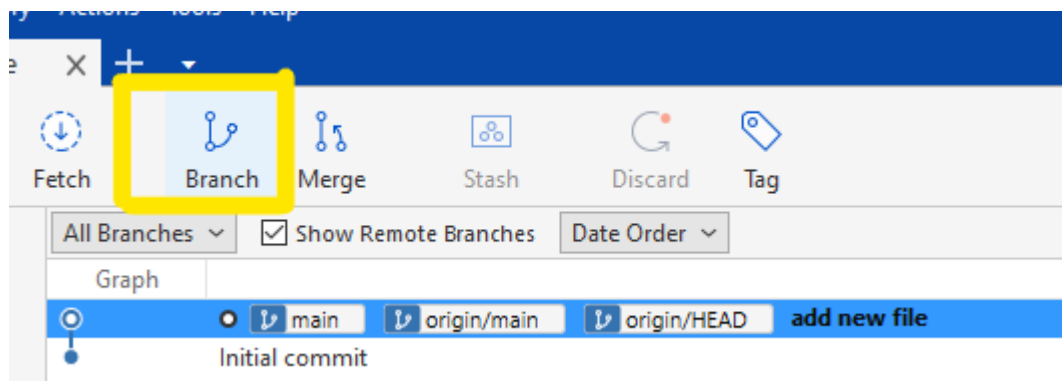
(Khung tin nhắn cho phần commit)



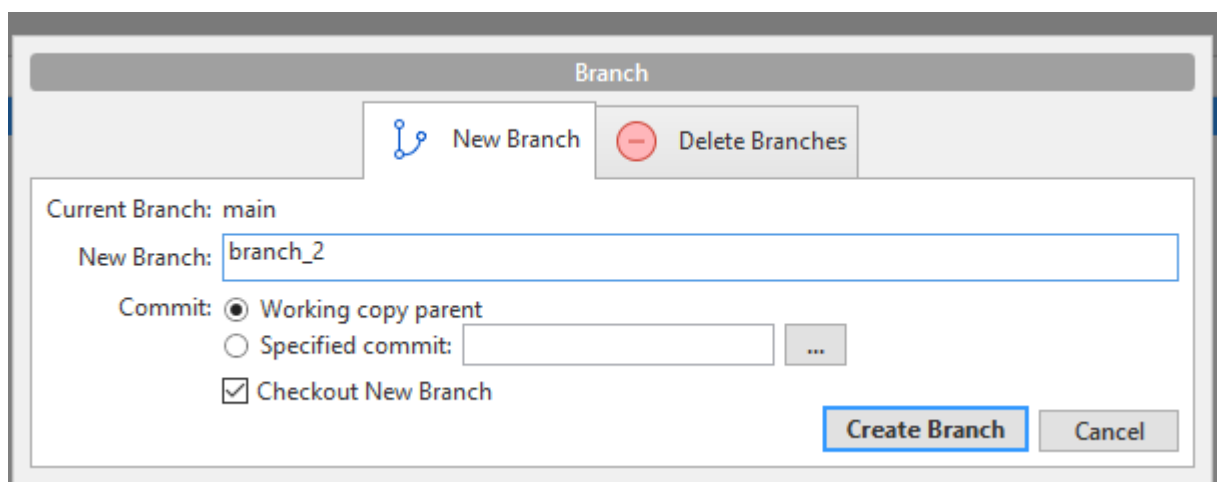
(Giao diện của SourceTree được cập nhật sau khi push các thay đổi lên remote)



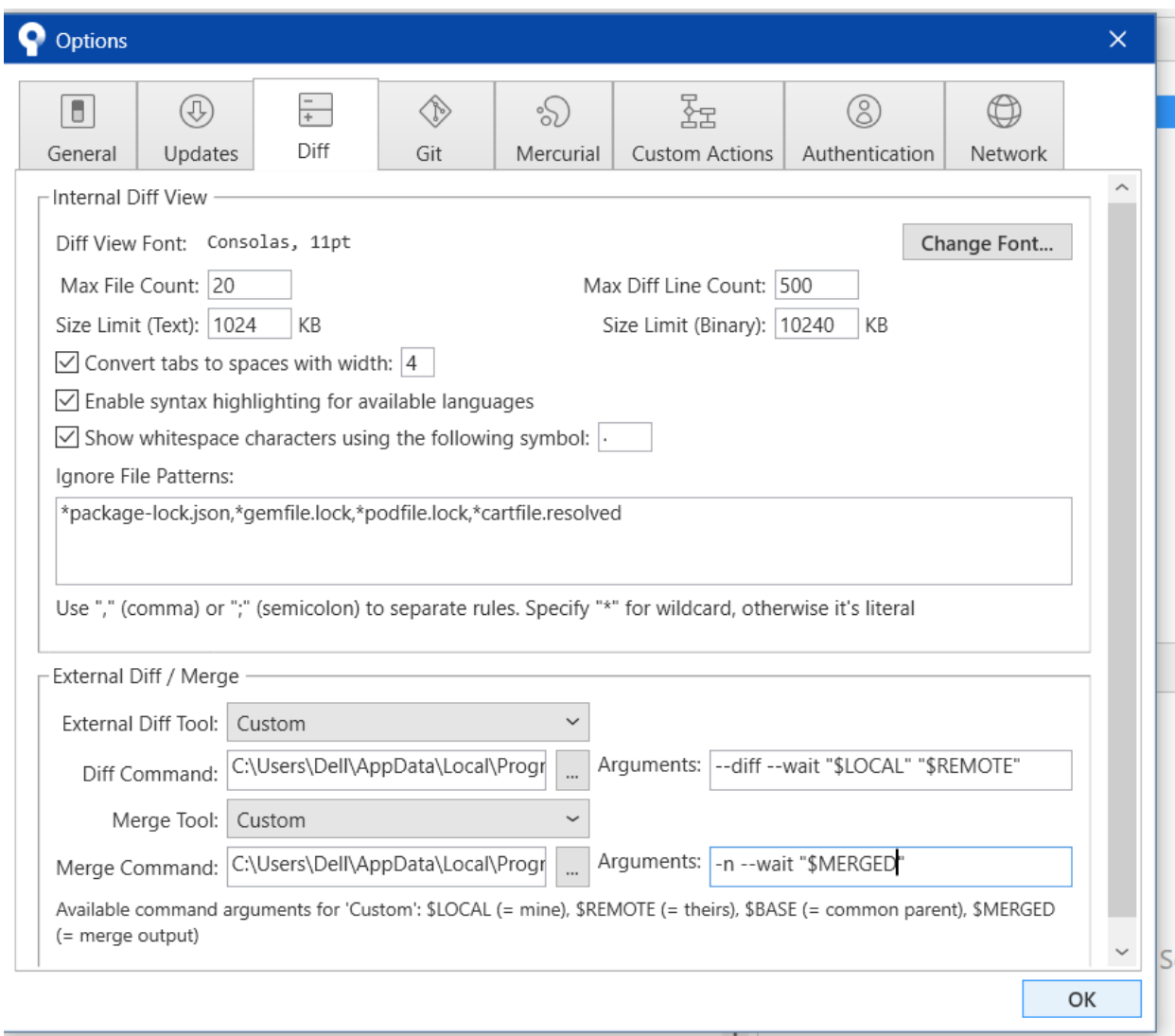
(Các thay đổi được hiển thị trên remote)



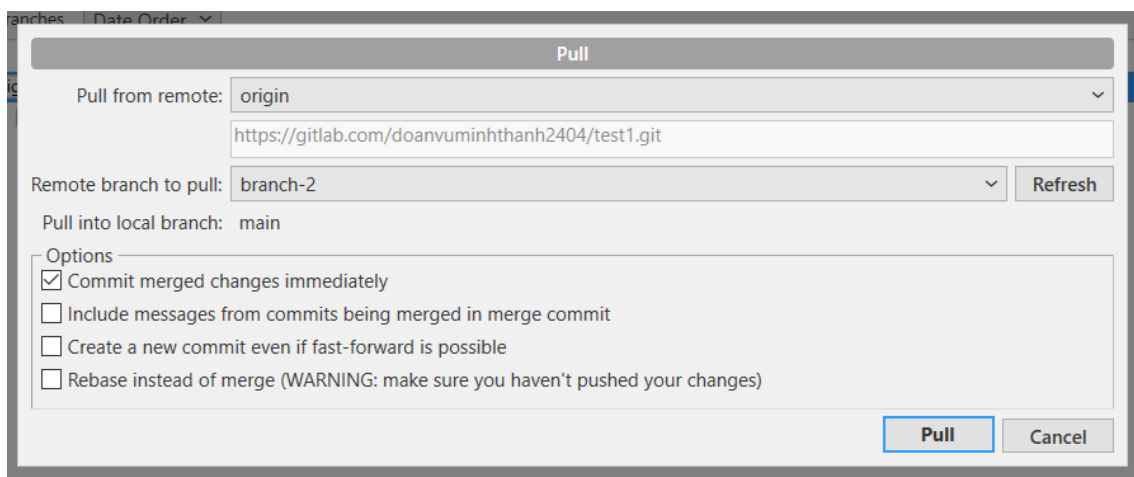
(Nút chọn tạo nhánh mới)



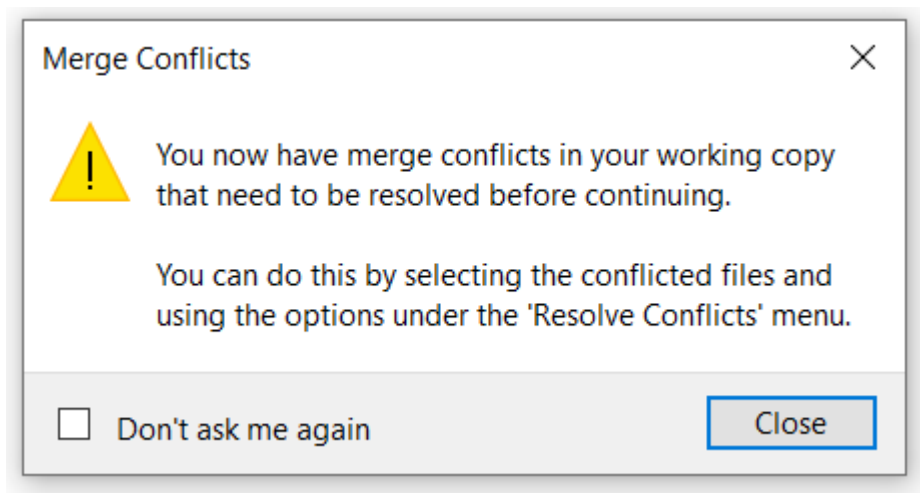
(Giao diện điền thông tin tạo nhánh mới)



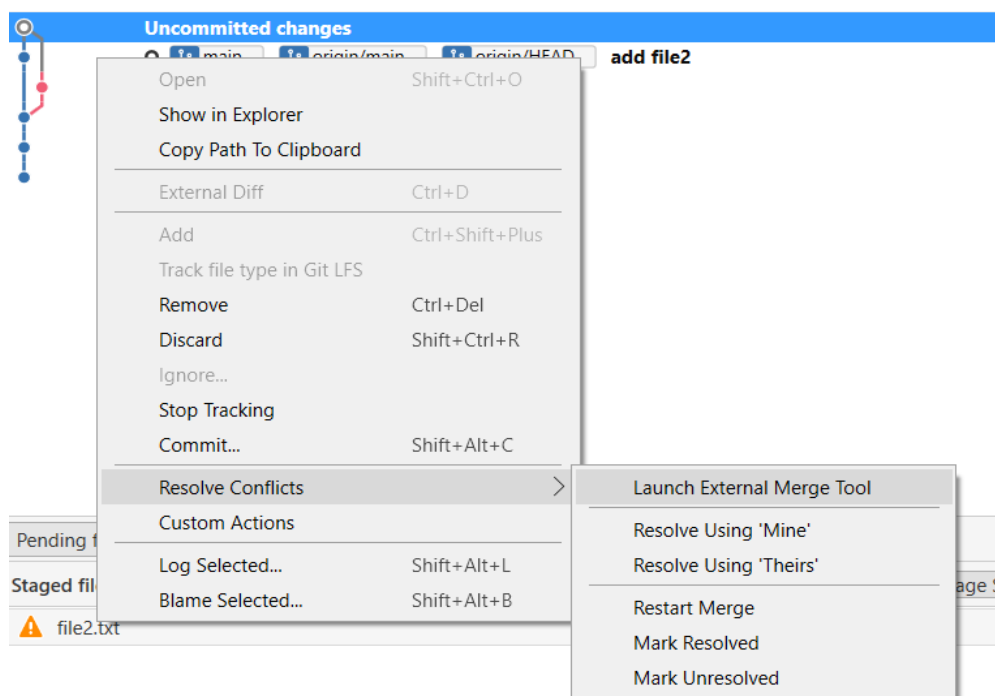
(Giao diện cấu hình cho phân xử lý mâu thuẫn-conflict code)



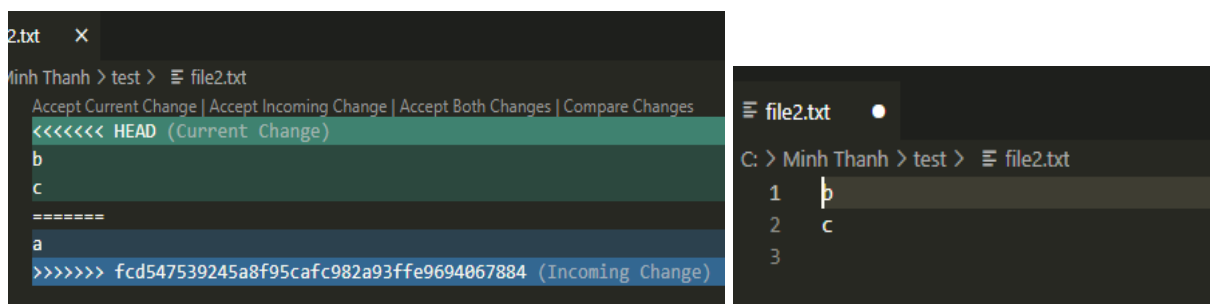
(Giao diện để sát nhập-merge nhánh)



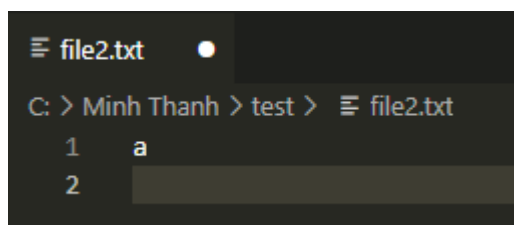
(Thông báo conflict)



(Lựa chọn xử lý conflict)



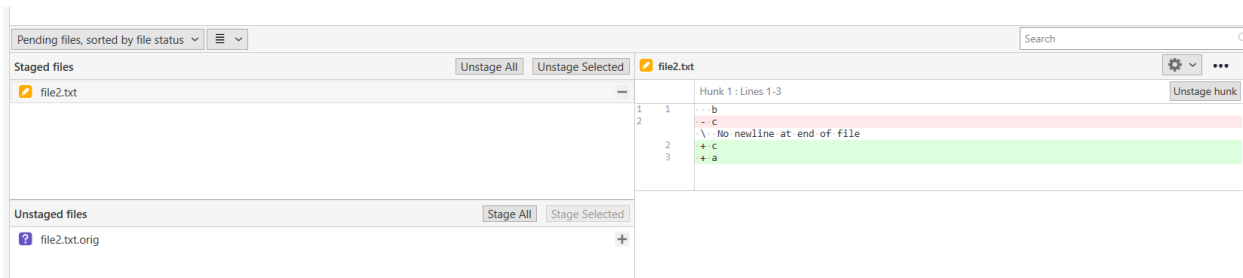
(Kết quả của giải quyết conflict với lựa chọn: Accept Current Change)



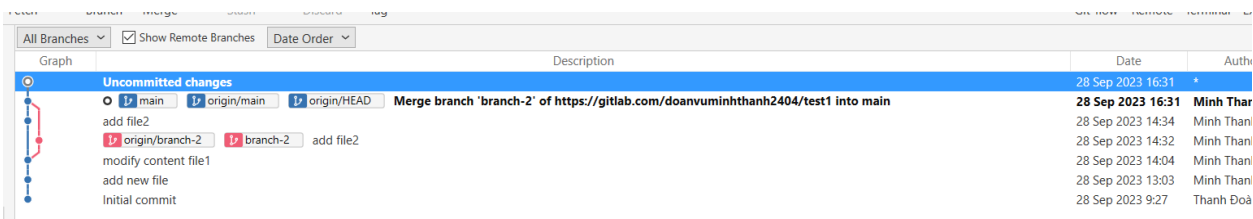
(Kết quả của lựa chọn giải quyết conflict: incoming change)

```
file2.txt
C: > Minh Thanh > test > file2.txt
1  b
2  c
3  a
4  |
```

(Kết quả của lựa chọn giải quyết conflict: both change)

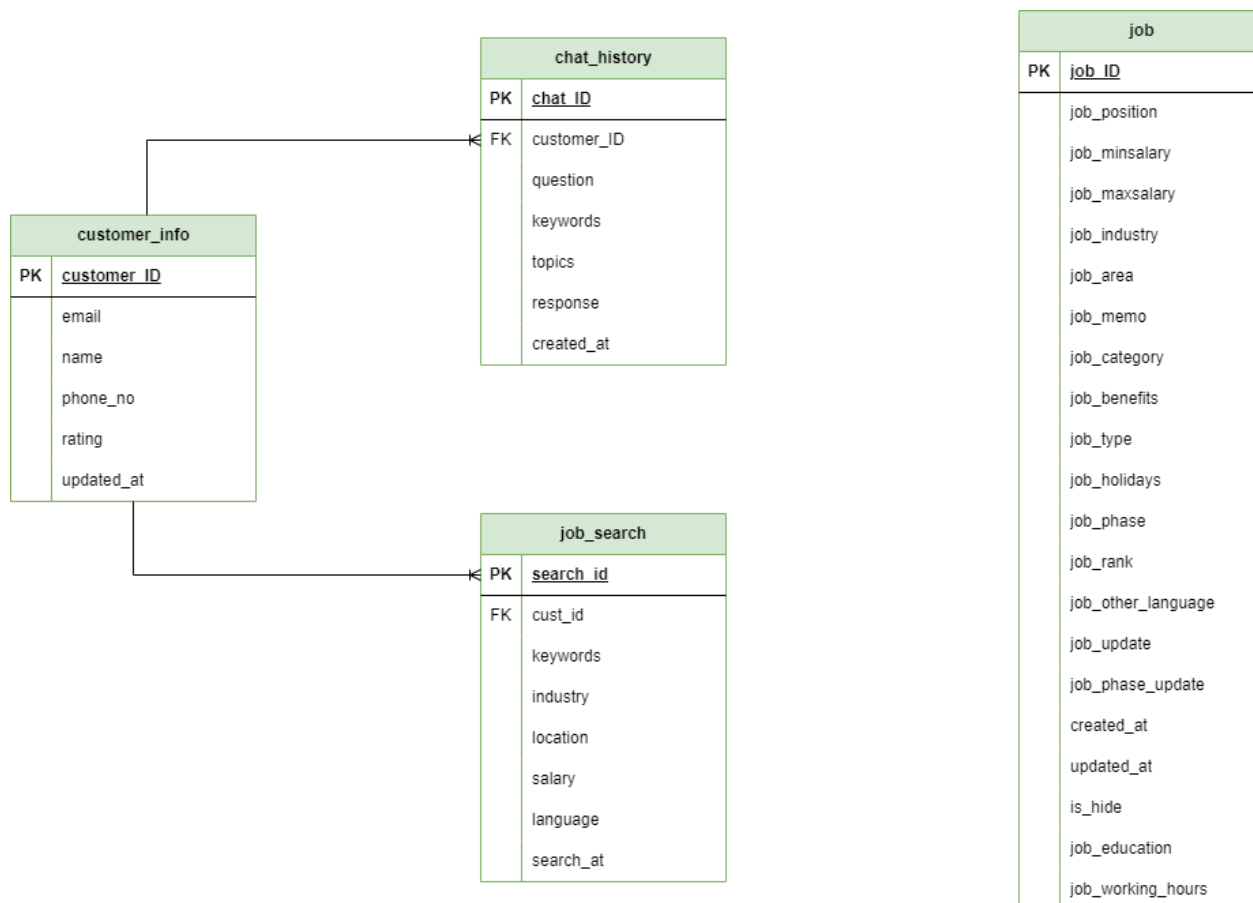


(Sau khi giải quyết conflict, chỉ đẩy file được xử lý conflict, bỏ qua file .orig)



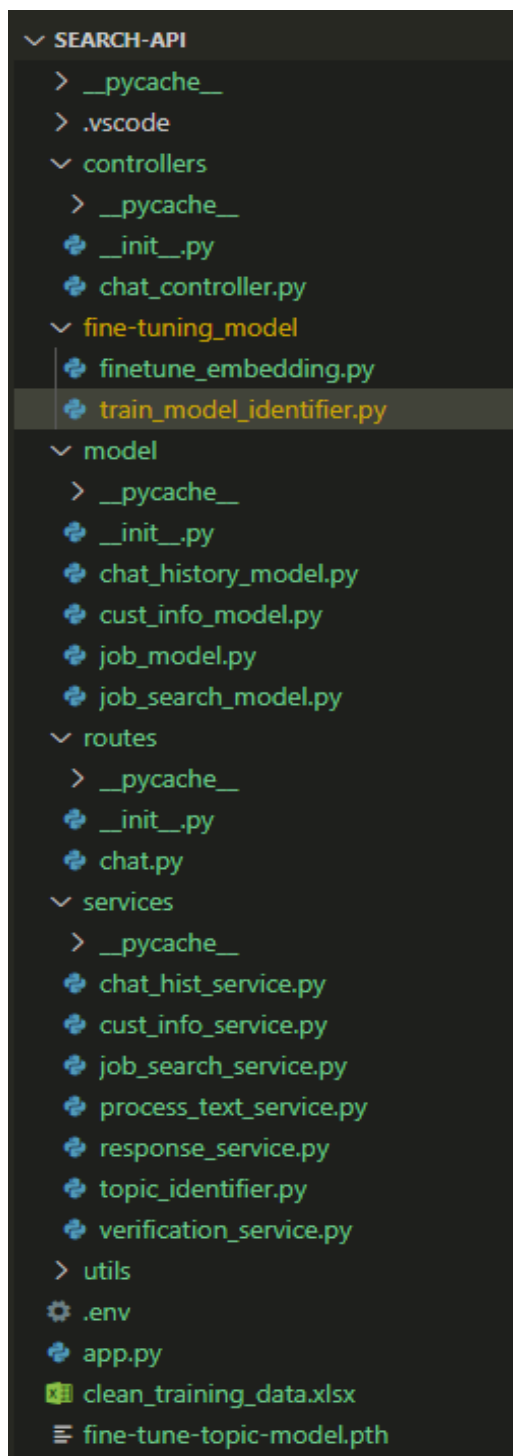
(Giao diện SourceTree sau khi merge branch thành công)

Đề án 2: Nhận nội dung thực tập và lên kế hoạch thực hiện dự án



(Mối quan hệ giữa các bảng trong cơ sở dữ liệu)

Đề án 3: Xây dựng mô hình thiết kế MVC và tạo API cho chatbot



(Tổ chức source code theo mô hình MVC)

```
from flask import Flask
from routes.chat import chat_bp
from utils.db import init_db
from dotenv import load_dotenv
import os
```

(tải các thư viện Flask để tạo API, các định tuyến trong file route.chat, load_dotenv để đọc các biến môi trường của cơ sở dữ liệu)

```
load_dotenv('.env')

app = Flask(__name__)

# Initialize database
app.config['DB_USER'] = os.getenv("DB_USER")
app.config['DB_PASS'] = os.getenv("DB_PASS")
app.config['DB_NAME'] = os.getenv("DB_NAME")
app.config['DB_HOST'] = os.getenv("DB_HOST")
app.config['DB_PORT'] = os.getenv("DB_PORT")
init_db(app)

# Register blueprint
app.register_blueprint(chat_bp)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=8000)
```

(đọc file .env và gán các biến môi trường để kết nối với cơ sở dữ liệu SQL của công ty, đăng ký blueprint, chọn số host và port)

Đề án 4: Tiền xử lý văn bản từ tin nhắn tự do của khách hàng

```
#dictionary consisting of the contraction and the actual value
Apos_dict={'s':" is","n't":" not","m':" am","ll':" will",
           "'d':" would","'ve":" have","'re":" are"}
#replace the contractions
for key,value in Apos_dict.items():
    if key in data:
        data=data.replace(key,value)
```

(thay thế các thể viết tắt thành từ gốc đầy đủ)

```
#remove emoji
data = emoji.replace_emoji(data, replace='')
#remove http, www, email, url
```

(loại bỏ các biểu tượng cảm xúc)

```
# Defining regex patterns.
replace_pattern = {
    "urlPattern"      : r"http\S+ | www\S+.",
    "userPattern"     : '@[^\s]+',
    "alphaPattern"    : "[^a-zA-Z0-9]"
}
for pattern in replace_pattern.keys(): #remove irrelevant characters
    data = re.sub(replace_pattern[pattern]," ", data)
```

(loại bỏ các đường dẫn, tên người dùng, các kí tự đặc biệt khác chữ, số)

```
#replace location abbreviations (HCM -> Ho Chi Minh, HN = Ha Noi)
abb_dict = {"hcm":"Ho Chi Minh", "hn":"Hanoi", "ha noi":"Hanoi", "vn":"Viet Nam"}
for key, value in abb_dict.items():
    if key in data.lower():
        before = data[:data.lower().find(key)]
        after = data[data.lower().find(key)+len(key):]
        data = before+value+after
```

(xử lý từ viết tắt địa điểm)

```
#One letter in a word should not be present more than twice in continuation
data = ''.join(''.join(s)[:2] for _, s in itertools.groupby(data))
```

(cắt bỏ các kí tự lặp lại liên tục trên 3 lần trong mỗi từ)

```
# NER identifier
nlp = spacy.load("en_core_web_lg")
ner = {}
doc = nlp(data)
for ent in doc.ents:
    #print(ent.text, ent.start_char, ent.end_char, ent.label_)
    ner[ent.text] = ent.label_
ner_dict = {}
location_count, date_count = location_count, date_count
for key, value in ner.items():
    if value == "GPE":
        data = data.replace(key, f'location{location_count}')
        ner_dict[f'location{location_count}'] = key
        location_count += 1
    elif value == "DATE":
        data = data.replace(key, f'date{date_count}')
        # if list(ner.keys)[list(ner.keys).index(key)-1] == "": print("")
        ner_dict[f'date{date_count}'] = key
        date_count += 1
```

(xác định tên địa danh và từ biểu hiện ngày tháng năm)

```
#exclude some abb words related to job
job_position_abb = ['IT', 'HR', 'BPO', 'QC', 'QC']
no = 1
abb_job = {}
for word in data.split():
    if word in job_position_abb:
        data = data.replace(word, f"job{no}")
        abb_job[f'job{no}'] = word
        no += 1
```

(xác định các từ viết tắt về kĩ năng, ngành nghề thông dụng)

```
#auto correct misspelled words
spell = Speller(Lang='en')
data = spell(data)
```

(tự sửa lỗi sai chính tả)

```
#return abb_job back
no = 1
for word in data.split():
    if word == 'job'+ str(no):
        data = data.replace(f'job{no}', abb_job[word])
        no += 1
```

(trả lại các từ viết tắt về ngành nghề, kỹ năng)

```
ner_dict = {}
location_count = 1
date_count = 1

for i in cleaned_text:
    # print(cleaned_text[cleaned_text.index(i)])
    grammar_vocab_result = raw_text_clean.grammar_vocab(i, location_count, date_count)
    cleaned_text[cleaned_text.index(i)] = grammar_vocab_result['data']
    ner_dict[grammar_vocab_result['data']] = grammar_vocab_result['ner_dict']
    location_count = grammar_vocab_result['location_count']
    date_count = grammar_vocab_result['date_count']
```

(3 công việc đầu của hàm tổng hợp clean_data())

```
def POSTagger(sent_clean):  
    """ ...  
    result = {}  
  
    for i in sent_clean:  
        wordsList = nltk.word_tokenize(i)  
        tagged = nltk.pos_tag(wordsList)  
        resultDictionary = dict((x, y) for x, y in tagged)  
        filteredDictionary = {}  
        wordnet_lemmatizer = WordNetLemmatizer()  
  
        for (key, value) in resultDictionary.items():  
            if value in ["JJR", "JJS", "JJ", "NN", "NNP", "NNS", 'IN', 'RB']:  
                filteredDictionary[wordnet_lemmatizer.lemmatize(key)] = value  
        filtersent = " ".join(s for s in list(filteredDictionary.keys()))  
  
        result[filtersent] = filteredDictionary  
  
    return result
```

(Xác định vị trí, loại từ của từng từ trong câu - POSTagger)

```
postag_result = raw_text_clean.POSTagger(cleaned_text)  
cleaned_text = list(postag_result.keys())  
#postag_ vari to store postagger dict  
postag_ = list(postag_result.values())
```

(Hàm xác định POSTag được gọi trong hàm tổng hợp clean_data())

```
def vocab(sent_clean):
    #remove stop words
    for i in sent_clean:

        #import english stopwords list from nltk
        stop = stopwords.words('english')
        excluded_words = ['each', 'and', "of", 'now']
        stop = [word for word in stop if word not in excluded_words]

        x_tokens = i.split()
        x_list=[]
        #remove stopwords
        for word in x_tokens:
            if word not in stop:
                x_list.append(word)

        sent_clean[sent_clean.index(i)] = " ".join(s for s in x_list)

    #lemmatize
    for i in sent_clean:
        wordnet_lemmatizer = WordNetLemmatizer()

        x = [wordnet_lemmatizer.lemmatize(n) for n in i.split()]
        sent_clean[sent_clean.index(i)] = " ".join(s for s in x)

    return sent_clean
```

(Loại bỏ các từ thừa – stopwords và chuyển từ về dạng gốc - lemmatization)

```
cleaned_text = raw_text_clean.vocab(cleaned_text)

old_key_ner = list(ner_dict.keys())

#replace old words by new words from postagger_
for i in range(len(old_key_ner)):
    ner_dict[cleaned_text[i]] = ner_dict.pop(old_key_ner[i])
ner_dict_all = {} #key: clean text #value: {'ner_dict': ner_dict[clean_text], 'filter_pos_tag': raw_text_clean.POSTagger(cleaned_text).value()}
for key, value in ner_dict.items():
    ner_dict_all.update(value)
filter_pos_dict = {}
for i in postag:
    filter_pos_dict.update(i)

data_pos = {"clean_text": " ".join(i for i in cleaned_text), "ner_dict": ner_dict_all, "filtered_dict": filter_pos_dict}

return data_pos
```

(Các hàm POSTagger và Lemma được gọi trong hàm tổng hợp clean_data(); kết quả hàm clean_data) trả về là 1 object gồm nội dung văn bản đã làm sạch, từ điển tên riêng, từ điển POSTag của các từ trong chuỗi văn bản đã được tiền xử lý)

```
def chat(): #receive message
    data = request.json

    try:
        #get data from client
        mes = data.get("Mes")
        cust_id = data.get('Cust_id')

        # # Process input data Start # #

        # data cleaning -> return a clean text
        data_pos = raw_text_clean.clean_data(mes) #dict: dict of POSTagger, string of clean sentences, dict of ner

        filteredDictionary = data_pos['filtered_dict'] #POSTagger dict
        ner = data_pos['ner_dict'] #NER dict
        clean_text = data_pos['clean_text'] #cleaned message

        # # Process input data End # #
```

(Hàm `clean_data()` được gọi trong file controllers)

Đề án 5: Ứng dụng mô hình học sâu để xác định chủ đề tin nhắn

```
1 from nltk.corpus import wordnet
2
3 def get_word_synonyms_from_sent(word, sent):
4     word_synonyms = []
5     for synset in wordnet.synsets(word):
6         for lemma in synset.lemma_names():
7             if lemma in sent and lemma != word:
8                 word_synonyms.append(lemma)
9     return word_synonyms
10
11
12 word = input('topic: ')
13 sent = input('Question: ')
14 word_synonyms = get_word_synonyms_from_sent(word, sent.split())
15 print ("WORD:", word)
16 print ("SENTENCE:", sent)
17 print ("SYNONYMS FOR '" + word.upper() + "' FOUND IN THE SENTENCE: " + ", ".join(word_synonyms))
18 sent1 = re.sub( " ", ".join(word_synonyms)", "salary", sent)
19 print(sent1)
```

```
topic: salary
Question: curious average earnings software manager location share infomation
WORD: salary
SENTENCE: curious average earnings software manager location share infomation
SYNONYMS FOR 'SALARY' FOUND IN THE SENTENCE: earnings
```

(Sử dụng thư viện có sẵn: trường hợp câu có từ đồng nghĩa với từ chủ đề)

```
topic: salary
Question: What is the typical compensation for employees in higher-level executive roles within the company?
WORD: salary
SENTENCE: What is the typical compensation for employees in higher-level executive roles within the company?
SYNONYMS FOR 'SALARY' FOUND IN THE SENTENCE:
```

(Sử dụng thư viện có sẵn: trường hợp câu không có từ đồng nghĩa với từ chủ đề)

```
salary: {'pay', 'remuneration', 'earnings', 'wage', 'salary'}
```

(Danh sách từ đồng nghĩa do thư viện wordnet cung cấp không đa dạng, phong phú)

1	question	label
12	what's the amount of open positions of IT in HCM? @ajfjas www.jdafaf.com @@@ (👉 :')	amount
14	How many vacant IT positions are there in Ho Chi Minh City?	amount
15	Could you tell me the current number of open IT job roles in HCM?	amount
16	What's the count of available IT positions in Ho Chi Minh?	amount
17	Do you have information about the total IT job openings in HCM?	amount
18	I'm curious about the quantity of unfilled IT jobs in Ho Chi Minh City.	amount
19	Can you provide me with the total number of IT job vacancies in HCM?	amount
20	What's the total count of open positions for IT roles in Ho Chi Minh?	amount
21	Could you share the number of available IT job positions in HCM?	amount
22	I'm interested in knowing the quantity of IT job openings in Ho Chi Minh.	amount
23	What's the current count of IT job vacancies in HCM?	amount
124	What's the quantity of open positions available in your company?	amount
125	Could you tell me the number of job openings you have right now?	amount
126	Do you know the total count of vacant positions in your organization?	amount
127	Can you provide the sum of open positions across different departments?	amount
128	How many job vacancies are there in total?	amount
129	What's the entirety of open job positions I can apply for?	amount
130	Could you share the aggregate number of job openings in your company?	amount
131	Do you have any information about the volume of available job positions?	amount
132	What's the figure for the total number of vacant jobs?	amount
133	How's the total amount of open positions distributed among departments?	amount
134	Can you give me the whole count of job vacancies?	amount
135	What's the grand total of open positions across the organization?	amount
136	Do you have the aggregation of vacant positions for different roles?	amount

Sheet1
Ready 111 of 322 records found Scroll Lock Accessibility: Good to go

(Các tin nhắn được dán nhãn amount)

1	question	label
2	Could you provide me with the average salary for a software manager in Ho Chi Minh City?	salary
3	What is the typical pay for a software manager in HCM?	salary
4	I'm curious about the average earnings of software managers in Ho Chi Minh City. Can you share that information?	salary
5	Can you give me an idea of the average compensation for software managers in HCM?	salary
6	What's the usual salary range for software managers in Ho Chi Minh City?	salary
7	I'd like to know the average pay scale for software managers in HCM. Could you help me with that?	salary
8	Could you shed some light on the average income of software managers in Ho Chi Minh City?	salary
9	What's the standard remuneration for a software manager in HCM?	salary
10	I'm researching software manager salaries in Ho Chi Minh City. What's the average figure?	salary
11	Can you provide me with information on the average wage of software managers in HCM?	salary
13	what's the average salari of software manager in HCM? @ueh www.facebook.com @@@ (👉 :')	salary
24	What is the average salary for this role?	salary
25	Can you give me an idea of the typical pay range?	salary
26	What can I expect in terms of compensation?	salary
27	Is there a standard wage range for this position?	salary
28	What is the average earnings for someone in this role?	salary
29	Can you provide information about average pay?	salary
30	How does the typical compensation package look for this position?	salary
31	Is there a standard wage scale for this role?	salary
32	What is the standard earnings range for this job?	salary
33	Can you tell me about the typical pay scale for this position?	salary
34	What is the average wage for someone with my level of experience?	salary
35	Can you provide details about the standard wage structure?	salary
36	How does the typical compensation compare to industry standards?	salary

Sheet1
Ready 111 of 322 records found Scroll Lock Accessibility: Good to go

(Các tin nhắn được dán nhãn salary)

1	question	label
224	How's the weather today?	trivia
225	Have you seen any good movies lately?	trivia
226	What's your favorite book?	trivia
227	Do you like to travel?	trivia
228	What's your go-to comfort food?	trivia
229	Have you tried any new recipes recently?	trivia
230	How's your day going?	trivia
231	Any exciting plans for the weekend?	trivia
232	Have you ever been skydiving?	trivia
233	What's your favorite hobby?	trivia
234	Do you enjoy hiking?	trivia
235	How do you like to spend your evenings?	trivia
236	Have you ever visited a foreign country?	trivia
237	What's the last concert you went to?	trivia
238	Are you a morning person or a night owl?	trivia
239	Do you have any pets?	trivia
240	What's your dream vacation destination?	trivia
241	Have you ever tried a new sport?	trivia
242	Do you like to cook or prefer eating out?	trivia
243	Are you a fan of live music?	trivia
244	Have you ever attended a music festival?	trivia
245	What's your favorite type of cuisine?	trivia
246	Have you ever tried painting or drawing?	trivia
247	Do you enjoy visiting art galleries?	trivia
248

Sheet1

Ready 100 of 322 records found Scroll Lock Accessibility: Good to go

(Các tin nhắn được dán nhãn trivia)

```
## Prepare training dataset START

# data preprocessing START
# import raw training dataset
topic_train_data = pd.read_excel('new3_clean_training_data.xlsx', sheet_name="Sheet1")
print(topic_train_data)
topic_train_data['clean_text'] = topic_train_data['question'].apply(lambda x: raw_text_clean.clean_data(x)['clean_text'])
topic_train_data['clean_text'].fillna('none', inplace=True)
#encode the label
label_encoding = {'amount':0, "salary": 1, "trivia": 2}
topic_train_data['label'] = topic_train_data['label'].map(label_encoding)

excel_file_path = 'new2_clean_training_data.xlsx'
# Export the DataFrame to an Excel file
topic_train_data.to_excel(excel_file_path, index=False)

# data preprocessing END

## Prepare training dataset END
```

(Làm sạch dữ liệu huấn luyện)

```
## Visualize - Generate the word cloud for specific class START

# import cleaned training dataset (change excel file name)
topic_train_data = pd.read_excel('new3_clean_training_data.xlsx', sheet_name="Sheet1")

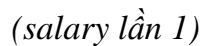
word_cloud_class = test_data[topic_train_data['label']==0]['clean_text']
#modify the label to your desired label name

wordcloud = WordCloud(width=800, height=400, background_color='white').generate(' '.join(word_cloud_class))

# Display the word cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()

## Visualization - Generate the word cloud for specific class END
```

Hàm biểu diễn WordCloud)





(trivia lần 1)

```
import torch
from transformers import BertForSequenceClassification, BertTokenizer
from torch.utils.data import DataLoader, TensorDataset
import pandas as pd
from services.process_text_service import raw_text_clean
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

(Import các hàm tiền xử lý dữ liệu, trực quan hóa dữ liệu, các thư viện chứa mô hình học sâu, các chỉ số đánh giá mô hình)

```
class UpdatePretrainedModel():  
  
    def train_and_save_model(new_input_texts, new_labels, save_path, num_classes, batch_size=32, learning_rate=1e-5):  
        # Load pre-trained BERT model and tokenizer  
        model_name = 'bert-base-uncased'  
  
        model = BertForSequenceClassification.from_pretrained(model_name, num_labels = num_classes)  
        # Load pre-trained weights into the modified model  
        # model.load_state_dict(model.state_dict())  
        tokenizer = BertTokenizer.from_pretrained(model_name)  
  
        # Tokenize new input_texts  
        new_tokenized_texts = [tokenizer.encode(text, add_special_tokens=True) for text in new_input_texts]  
  
        max_length = 128 # Adjust this based on your model's input requirements  
        new_tokenized_texts = [tokenizer.encode_plus(text, add_special_tokens=True, max_length=max_length, pad_to_max_length=True)['input_ids'] for text in new_input_texts]  
  
        # Convert to tensors and create a DataLoader  
        new_input_ids = torch.tensor(new_tokenized_texts)  
        new_labels = torch.tensor(new_labels)  
        new_dataset = TensorDataset(new_input_ids, new_labels)  
        new_dataloader = DataLoader(new_dataset, batch_size=32, shuffle=True)
```

(Khai báo mô hình gốc sẽ tinh chỉnh, tách từ, nhúng từ và thêm 1 lớp mới vào mô hình)

```
# Define loss function and optimizer  
criterion = torch.nn.CrossEntropyLoss()  
optimizer = torch.optim.Adam(model.parameters(), lr=1e-5)
```

(Định nghĩa hàm mất mát và hàm tối ưu)

```
# Initialize empty lists to store loss values
losses = []
# Fine-tuning loop with new data
num_epochs = 201
for epoch in range(num_epochs):
    running_loss = 0.0
    model.train()
    for batch_input_ids, batch_labels in new_dataloader:
        optimizer.zero_grad()
        outputs = model(input_ids=batch_input_ids, labels=batch_labels)
        loss = outputs.loss
        loss.backward()
        optimizer.step()

    #store loss value
    running_loss += loss.item()

    # Calculate average loss for the epoch
    if epoch % 50 == 0:
        epoch_loss = running_loss / len(new_dataloader)
        losses.append(epoch_loss)
        # Print and visualize the loss for each epoch
        print(f"Epoch [{epoch}/{num_epochs}] Loss: {epoch_loss}")
```

(tinh chỉnh mô hình với 200 epoch và cập nhật hàm mất mát)

```
# Save the model using torch.save()
torch.save(model.state_dict(), save_path)

return save_path
```

(Lưu mô hình đã được tinh chỉnh)


```
# fine-tuning model START
# import training dataset
topic_train_data = pd.read_excel(
    'new3_clean_training_data.xlsx', sheet_name="Sheet1")
# change excel file name to the cleaned training dataset

save_path = UpdatePretrainedModel.train_and_save_model(topic_train_data['clean_text'].tolist(
), topic_train_data['label'].tolist(), "fine-tuned-topic-model.pth", len(set(topic_train_data['label'].tolist())))
# modify the model path file name

# fine-tuning model END
```

(Hàm tinh chỉnh mô hình được gọi trong phần chính)

```
# Load test dataset (change excel file name)
test_data = pd.read_excel('new3_clean_training_data.xlsx', sheet_name = "Sheet1")

model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(set(test_data['label'].tolist())), from_tf=True)
model.load_state_dict(torch.load('fine-tune-topic-model.pth'))
model.eval()
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize and process text
input_texts = test_data['clean_text'].tolist()
encoded_inputs = tokenizer(input_texts, padding=True, truncation=True, return_tensors='pt')
input_ids = encoded_inputs['input_ids']
attention_mask = encoded_inputs['attention_mask']

# Get predictions
with torch.no_grad():
    outputs = model(input_ids, attention_mask=attention_mask)
    predicted_labels = torch.argmax(outputs.logits, dim=1).tolist()

# Add predicted labels to the DataFrame
test_data['predicted_label'] = predicted_labels

print(test_data)
```

(Đánh giá mô hình lần 1)

```
Accuracy: 0.66
Confusion matrix:
[[18.  0.  0.]
 [17.  1.  0.]
 [ 0.  0. 14.]]
Class 0: Precision = 0.51, Recall = 1.00, F1-Score = 0.68
Class 1: Precision = 1.00, Recall = 0.06, F1-Score = 0.11
Class 2: Precision = 1.00, Recall = 1.00, F1-Score = 1.00
Precision: 0.8380952380952381
Recall: 0.6851851851851851
```

(Kết quả đánh giá mô hình tinh chỉnh lần 1)



(dữ liệu nhóm salary chỉnh sửa lần 1)

```
[322 rows x 4 columns]
Accuracy: 0.9937888198757764
Confusion matrix:
[[111.  0.  0.]
 [ 0. 111.  0.]
 [ 2.   0. 98.]]
Class 0: Precision = 0.98, Recall = 1.00, F1-Score = 0.99
Class 1: Precision = 1.00, Recall = 1.00, F1-Score = 1.00
Class 2: Precision = 1.00, Recall = 0.98, F1-Score = 0.99
Precision: 0.9941002949852508
Recall: 0.9933333333333333
```

(Kết quả đánh giá mô hình tinh chỉnh lần 2)


```
def topic_identifier(class_dict, input_sent):
    """ ...
    #use wordnet.synsets to find the topic
    predicted_topic = ""
    for i in list(class_dict.values()):
        if Topic.get_word_synonyms_from_sent(i, input_sent)['topic'] != "":
            predicted_topic = Topic.get_word_synonyms_from_sent(i, input_sent)['topic']
            input_sent = Topic.get_word_synonyms_from_sent(i, input_sent)['sent']
            break

    if predicted_topic == "":
        #if the result returns null, try BERT
        model = BertForSequenceClassification.from_pretrained('bert-base-uncased', num_labels=len(list(class_dict.keys())), from_tf=True)
        model.load_state_dict(torch.load('fine-tune-topic-model.pth'))
        model.eval()
        tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

        # input_text = "Can you give me an idea of the average compensation for software managers in HCM?"
        encoded_input = tokenizer.encode_plus(input_sent, add_special_tokens=True, return_tensors='pt')
        # output = model(**encoded_input)
        # Perform topic classification
        with torch.no_grad():
            outputs = model(**encoded_input)

        # Get the predicted label (topic)
        predicted_label = torch.argmax(outputs.logits).item()

        # Get the topic based on the predicted label
        predicted_topic = class_dict[predicted_label]

    return {'topic':predicted_topic.strip(), 'label_sent': input_sent}
```

(luồng làm việc xác định chủ đề tin nhắn)

Đề án 6: Cấu thành câu truy vấn SQL

```
def convert_month(month_int):
    """ ...
    month_int = dt.strptime(month_int, '%m')
    month_name = month_int.strftime('%B')
    return month_name

def identify_date_type(word):
    """ ...
    month_names = list(calendar.month_name[1:]) # Get a list of month names
    result_list = []

    for date_string in word.split():
        if date_string.capitalize() in month_names: #identify month
            result_list.append({"type": "month", "word": date_string, 'converted': parser.parse(date_string).month})
        elif date_string.isdigit():
            try:
                month_name = Time_Analyse.convert_month(date_string)
                if month_name.capitalize() in month_names: #identify month
                    result_list.append({"type": "month", "word": month_name, 'converted': date_string})
            except ValueError:
                year = int(date_string)
                if (2000 <= year) & (year <= 2100): #identify year
                    result_list.append({"type": "year", "word": year})

    year = ""
    for i in result_list:
        for key, value in i.items():
            if value == "year": #extract year data
                year = i['word']

    if not result_list:
        return {'main_time': '', 'year': year}
    else:
        return {'main_time': result_list[0], 'year': year}
```

(Hàm phân tích yếu tố thời gian)

```
def time_interpretation(select_list, groupby_list, orderby_list):
    """ """
    def list_to_sent(list, common_phrase, num_omission, as_col = False):
        """ """
        sent = ""
        for i in list:
            if as_col:
                sent += f"{i}{common_phrase} AS {i.capitalize()}_, "
            else:
                sent += f"{i}{common_phrase}"
        sent = sent[:-num_omission]
        return sent

    select_component = list_to_sent(select_list, "(DATE(job_update))", 2, True)
    groupby_component = list_to_sent(groupby_list, "(DATE(job_update))", ", 2)
    orderby_component = list_to_sent(orderby_list, "(DATE(job_update)) DESC, ", 2)

    return {"select": select_component, "groupBy": groupby_component, "orderBy": orderby_component}

match maintime:
    case "year":
        return {'maintime': maintime, 'timeword': time_word, \
                'time_component_dict': Time_Analyse.time_interpretation(['YEAR'], ['YEAR'], ['YEAR']), \
                'year': Time_Analyse.identify_date_type(word)['year']}
    case "quarter":
        return {'maintime': maintime, 'timeword': time_word, \
                'time_component_dict': Time_Analyse.time_interpretation(['QUARTER', 'YEAR'], ['QUARTER'], ['YEAR', 'QUARTER']), \
                'year': Time_Analyse.identify_date_type(word)['year']}
    case "month":
        return {'maintime': maintime, 'timeword': time_word, \
                'time_component_dict': Time_Analyse.time_interpretation(['MONTH', 'YEAR'], ['MONTH'], ['MONTH']), \
                'year': Time_Analyse.identify_date_type(word)['year']}
    case "week":
        return {'maintime': maintime, 'timeword': time_word, \
                'time_component_dict': Time_Analyse.time_interpretation(['WEEK', 'MONTH', 'YEAR'], ['WEEK'], ['WEEK']), \
                'year': Time_Analyse.identify_date_type(word)['year']}
```

(Hàm thiết lập điều kiện thời gian trong câu truy vấn SQL)

```
def query_format(topic, cal_on, location, date, open=True):
    """ ...

    select_add = ""
    groupby_add = "GROUP BY "
    orderby_add = "ORDER BY "

    # # Determine the year START
    year = ""
    if date:
        for i in date:
            time_component_dict = Time_Analyse.main_time_component(i) # analyse time phrase

            if time_component_dict['maintime'] == 'year':
                year = f"YEAR DATE(job_update)={time_component_dict['timeword']} AND"
                having = f"HAVING Year_ = {time_component_dict['timeword']}"
            elif time_component_dict['year'] != "":
                year = f"YEAR DATE(job_update)={time_component_dict['year']} AND"
                having = f"HAVING Year_ = {time_component_dict['year']}"

            select_add += (
                time_component_dict['time_component_dict']['select']+", "
            )
            groupby_add += (
                time_component_dict['time_component_dict']['groupby']+", "
            )
            orderby_add += (
                time_component_dict['time_component_dict']['orderby']+", "
            )

    if year == "":
        if dt.now().month <= 8:
            year = f"YEAR DATE(job_update)={dt.now().year -1}"
            having = f"HAVING Year_ = {dt.now().year -1}"
        else:
            year = f"YEAR DATE(job_update)={dt.now().year}"
            having = f"HAVING Year_ = {dt.now().year}"

    # # Determine the year END
```

(Bổ sung các thành phần trong câu truy vấn với thông tin về thời gian)

```
# # Generate location condition phrase START

location_filter = ""

if Location:
    for i in Location:
        location_filter += f'(job_area_name LIKE "% {i} %" OR job_area_summary LIKE "% {i} %") OR '
        location_filter = f'AND ({location_filter[:-4]})'

# # Generate location condition phrase END

# condition = Query.generate_linked_keywords(keyword) #generate keyword condition phrase

# Add job_phase condition phrases if the message directs "open position/vacancy"
if open:
    having += ' AND job_phase = "Open"'
    select_add += "job_phase, "
    year += ' AND job_phase = "Open" '
```

(Câu truy vấn bổ sung thêm điều kiện địa điểm)

```
# # Complete the query statement START
match topic:
    case "amount":
        if cal_on == "":
            cal_on = "job_posititon"
        sql = f'SELECT {select_add}COUNT({cal_on}) as Count_ FROM jobs WHERE is_hide = 0 \
            AND {year} {location_filter} {groupby_add[:-2]} {having} {orderby_add[:-2]}'
    case "salary":
        match cal_on:
            case "average" | "standard" | "":
                cal_on = 'average'
                sql = f'SELECT {select_add}CONCAT(ROUND(AVG(job_minsalary)), " - ", ROUND(AVG(job_maxsalary)), " (USD)") AS AVG_ \
                    FROM jobs WHERE is_hide = 0 AND {year} AND job_minsalary <> 0 AND job_maxsalary <> 0 \
                    {location_filter} {groupby_add[:-2]} {having} {orderby_add[:-2]}'
```

(Hoàn thiện câu truy vấn cho từng chủ đề tin nhắn)

```
def get_reponse(query_sent, cal_on, topic, location, date):
    """ ...
    #call API

    result = db.session.execute(text(query_sent)) #execute the query
    df = pd.DataFrame.from_dict(result). astype(str) #convert to dataframe, convert data type of columns to string

    if location != "":
        location = ' in ' + location #formatting location phrase for the response

    def extract_value(column_name, topic, location, date):
        """
        - generate response displayed to user
        - params:
            + column_name (str): column whose value will be displayed
            + topic (str): main topic of the message
            + location (str)
            + date (str)
        -return: response (str)

        """
        values = df[column_name].tolist()
        response = f'The {topic.lower()} {location} {date} is '
        if len(values) == 1:
            response += f' { " ".join(values)}.'
        elif len(values) < 1:
            print("in process of updating.")
        return response

    try:
        match topic:
            case 'amount':
                response = extract_value('Count_', " ".join([topic, "of job positions"]), location, date)
            case 'salary':
                response = extract_value('AVG_', " ".join([cal_on, topic]), location, date)

    except Exception as e:
        print(f"An error of type {type(e)} occurred: {e} {traceback.extract_tb(e.__traceback__)[0].lineno}")
        response = 'Sorry, this information has not been updated in our system...'

    return " ".join(response.split())
```

(Hàm trích xuất kết quả truy vấn SQL và chuyển đổi thành câu phản hồi cho khách hàng)

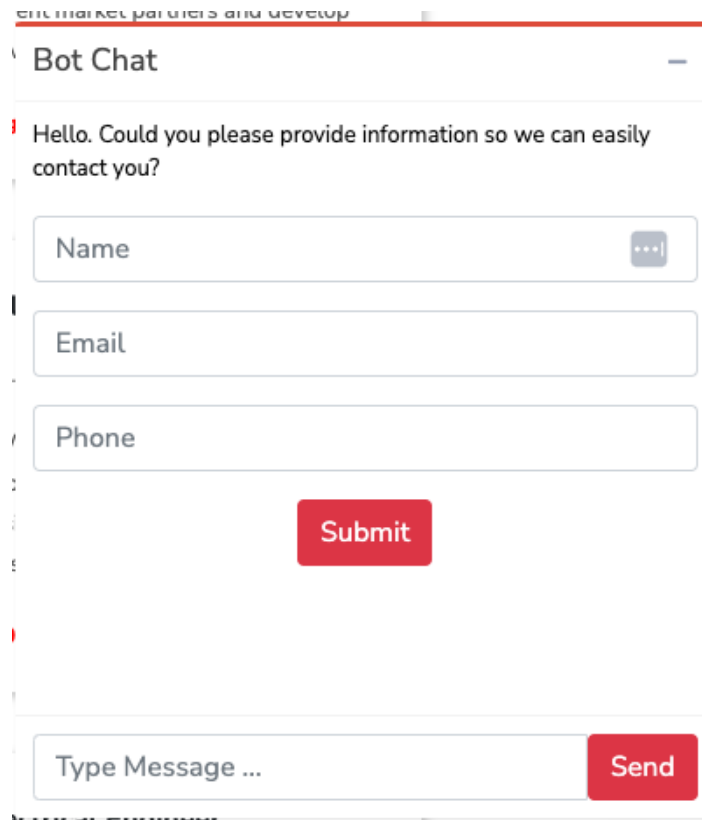
Đề án 7: Thiết kế giao diện người dùng cho chatbot và gọi API



(Header của chatbot)

```
<div class="box-body"> <!-- display chat content/history -->
  <div class="direct-chat-messages">
    <div class="chat-userInfo">
      <p class="text-black" style="color: black; font-size:small">Hello. Could you please provide
        information so we can easily contact you?</p>
      <form class="contact-info" action="{{route('saveData')}}" method="POST">
        <div class="form-group">
          <input type="text" name='U_Name' class="form-control" aria-describedby="Name" placeholder="Name" id='U_Name' required>
        </div>
        <div class="form-group">
          <input type="email" name='U_Email' class="form-control" aria-describedby="Email" placeholder="Email" id='U_Email' required>
        </div>
        <div class="form-group">
          <input type="phone" name='U_Phone' class="form-control" aria-describedby="Phone" placeholder="Phone" id='U_Phone' required>
        </div>
        <div class="text-center">
          <button class="btn btn-danger btn-flat" id="submitUserInfo">Submit</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

(Thiết kế biểu mẫu thông tin liên lạc của khách hàng)



The screenshot displays a web interface for a 'Bot Chat'. At the top, a red header bar contains the text 'ent market partners and develop'. Below this, the title 'Bot Chat' is centered. A message from the bot reads: 'Hello. Could you please provide information so we can easily contact you?'. The form consists of three input fields: 'Name' (with a dropdown arrow icon), 'Email', and 'Phone'. A red 'Submit' button is positioned below these fields. At the bottom of the chat window, there is a text input field labeled 'Type Message ...' and a red 'Send' button.

(Giao diện biểu mẫu thông tin liên lạc)


```
function submitUserInfo(name, email, phone) {  
  $.ajax({  
    type: 'POST',  
    url: `http://192.168.0.228:8000/userInfo`,  
    data: JSON.stringify({  
      Name: name,  
      Email: email,  
      Phone: phone  
    }),  
    contentType: "application/json",  
    success: function(result) {  
      cust_id = result.data  
      console.log(cust_id)  
      localStorage.setItem('cust_id', cust_id)  
    },  
    error: function(jqXHR, ajaxOptions, thrownError) {  
    },  
  });  
}
```

(Hàm .ajax() gọi API backend để xử lý thông tin liên lạc của khách hàng và lưu mã khách hàng vào local storage)

```
// Cust_info update/insert START

$(document).on('submit', '.contact-info', function(e) {
    if (!this.checkValidity()) {
        e.preventDefault();
        e.stopPropagation();
    } else {
        let name = $('#U_Name').val()
        let email = $('#U_Email').val()
        let phone = $('#U_Phone').val()

        console.log(name, email, phone)

        submitUserInfo(name, email, phone);
        // The value of cust_id has been updated at this point
        console.log(localStorage.getItem('cust_id'));

        $('.chat-userInfo').toggleClass('chat-hide')


        $('.direct-chat-messages').append(searchfrom)

        save_mes_hist()
    }
})

// Cust_info update/insert END
```

(sau khi xử lý thông tin liên lạc của khách hàng, chatbot sẽ gửi tiếp biểu mẫu tìm việc, đồng thời, lịch sử giao diện chat bắt đầu được ghi lại vào local storage)

Bot Chat



Hello. Please enter the necessary information to start looking for a job.

Search Job

Location

Industry

Salary

SEARCH

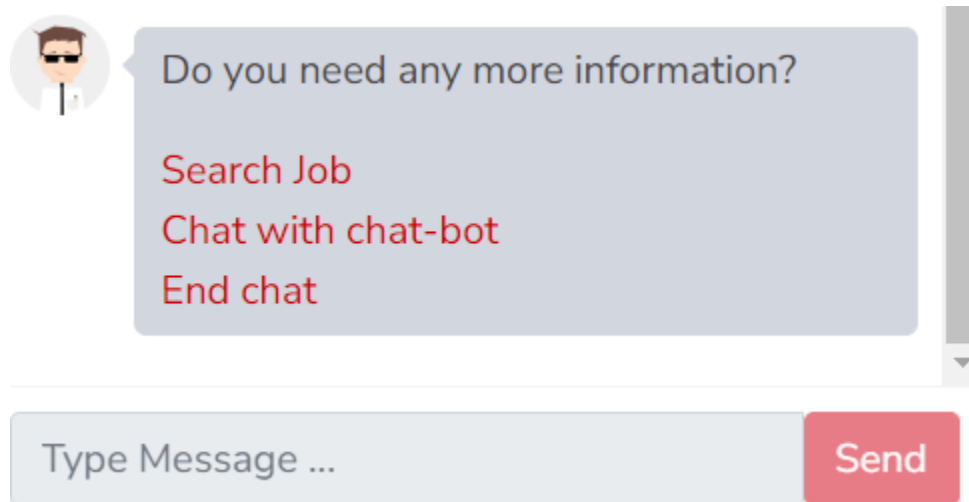
Type Message ...

Send

(Giao diện của biểu mẫu tìm việc)

```
$.ajax({
  type: type,
  url: ajaxUrl,
  data: formData,
  dataType: "json",
  success: function(result) {
    if (result["msg"] == "success") {
      let jobs = result["jobsChat"]
      if (result["count"] > 0) {
        $('.direct-chat-messages').append(
          `<div class="direct-chat-msg">
            
            <div class="direct-chat-text">
              Here is the corresponding job information:<br />
              ${jobs}
            </div>
          </div>`
        )
        // save search job history
      } else {
        $('.direct-chat-messages').append(
          `<div class="direct-chat-msg">
            
            <div class="direct-chat-text">
              There is no job matched your request.
            </div>
          </div>`
        )
      }
    }
    searchJobHistory(localStorage.getItem('cust_id'), jobName.trim(), jobIndustry.trim(), "", jobLocation.trim(), jobSalary.trim())
    bot_ans()
  }
})
```

(Hàm .ajax() gọi API xử lý yêu cầu tìm việc và hiển thị kết quả tìm kiếm, lịch sử tìm kiếm được lưu lại)



(Sau mỗi câu hỏi đáp từ chatbot, 3 lựa chọn tác vụ sẽ tự động hiện lên)

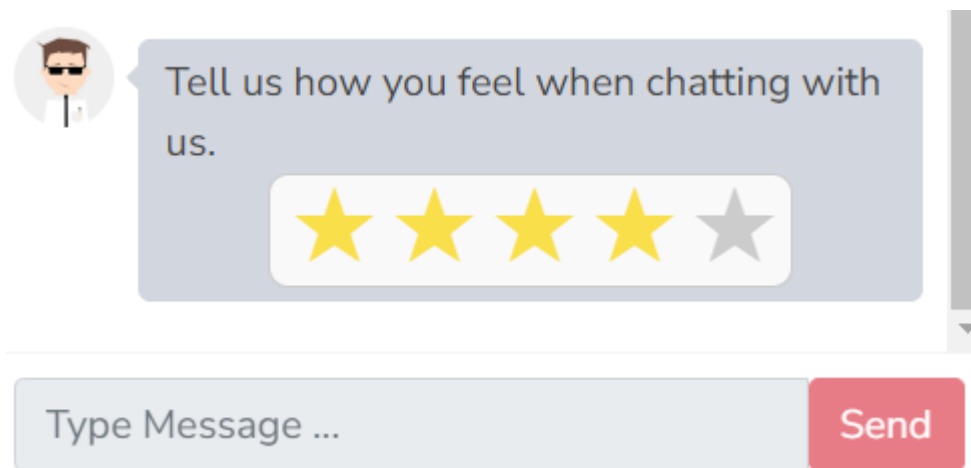
```
// Send button
$(document).on('click', '#btn-chat-send', async function() {
    $('.chat-freechat').toggleClass('chat-hide')

    let mes = $('#input-chat-mess').val()
    $('.direct-chat-messages').append( ...
    ...
    </div>
    </div>`)

    let cust_id = localStorage.getItem('cust_id');
    console.log(cust_id)
    console.log(mes)

    if (mes && cust_id) {
        await analyzeFreeChat(mes, cust_id)
        $('#input-chat-mess').val("")
    } else {
        console.log(cust_id)
    }
})
```

(Hàm gọi API để xử lý tin nhắn tự do của khách hàng)



(Giao diện đánh giá trải nghiệm chatbot)

```
function clear_mes_hist() {  
  let hist = localStorage.getItem('cust_mes')  
  localStorage.setItem('cust_mes', '')  
  localStorage.setItem('cust_id', '')  
  if (hist)  
    $('.direct-chat-messages').html('<div class="chat-userInfo">...'  
}
```

(Hàm xóa lịch sử chat trong local storage khi khách hàng chọn đóng cửa sổ chatbot)

CHƯƠNG 5: NHẬT KÝ CÔNG VIỆC

Ngày bắt đầu	Ngày kết thúc	Người tham gia	Vai trò của sinh viên	Tên công việc
07/08/2023	07/08/2023	Nguyễn Hoàng Hiệp	Thực tập sinh	Tìm hiểu về trang web Pasona.vn
08/08/2023	08/08/2023	Nguyễn Hoàng Hiệp	Thực tập sinh	Quản lý source code bằng Git và SourceTree
09/08/2023	10/08/2023	Nguyễn Phi Hùng	Thực tập sinh	Nhận nội dung dự án cá nhân và lên bảng dự tính công việc (Estimation)
11/08/2023	14/08/2023	Minh Thanh	Người thực hiện chính	Phác thảo cách thức hoạt động của chatbot
15/08/2023	15/08/2023	Minh Thanh	Người thực hiện chính	Vẽ lưu đồ dữ liệu
15/08/2023	15/08/2023	Minh Thanh	Người thực hiện chính	thu thập các mẫu thiết kế chatbot tham khảo
16/08/2023	16/08/2023	Nguyễn	Thực tập	Mô hình MVC là gì?

		Phi Hùng	sinh	
16/08/2023	16/08/2023	Nguyễn Phi Hùng	Thực tập sinh	Web API, Flask fundamental
17/08/2023	17/08/2023	Minh Thanh	Người thực hiện chính	Định nghĩa các route và các model
18/08/2023	18/08/2023	Nguyễn Phi Hùng	Thực tập sinh	Giới thiệu phần mềm test ARC, Postman
21/08/2023	21/08/2023	Minh Thanh	Người thực hiện chính	Hàm lưu/cập nhật thông tin liên lạc khách hàng
22/08/2023	22/08/2023	Minh Thanh	Người thực hiện chính	Lưu lịch sử tìm kiếm công việc
23/08/2023	23/08/2023	Minh Thanh	Người thực hiện chính	Lưu dữ liệu đánh giá chatbot từ khách hàng
24/08/2023	28/08/2023	Minh Thanh	Người thực hiện chính	Hàm làm sạch tin nhắn thô được gửi từ khách hàng
28/08/2023	28/09/2023	Minh Thanh	Người thực hiện chính	Dựng mô hình xác định chủ đề của tin nhắn
05/09/2023	05/09/2023	Minh	Người thực	Chuẩn bị dữ liệu huấn luyện mô

		Thanh	hiện chính	hình
06/09/2023	13/09/2023	Minh Thanh	Người thực hiện chính	Huấn luyện mô hình với các chỉ số khác nhau, đánh giá độ hiệu quả của mô hình qua các lần thử
14/09/2023	14/09/2023	Minh Thanh	Người thực hiện chính	Chỉnh sửa dữ liệu huấn luyện
14/09/2023	18/09/2023	Minh Thanh	Người thực hiện chính	Huấn luyện mô hình với dữ liệu mới (thành công)
18/09/2023	19/09/2023	Minh Thanh	Người thực hiện chính	Xác định các thành phần tạo một câu query SQL, tạo 1 câu query hoàn chỉnh, truy vấn dữ liệu bằng SQL-Alchemy, định dạng câu phản hồi
20/09/2023	24/09/2023	Nguyễn Phi Hùng	Thực tập sinh	HTML, CSS, Javascript, JQuery, Ajax Fundamental
24/09/2023	02/10/2023	Minh Thanh	Người thực hiện chính	Thiết kế giao diện (UI) cho chatbot, gọi API cho các sự kiện
03/10/2023	06/10/2023	Minh Thanh	Người thực hiện chính	Gọi API cho các sự kiện

07/10/2023	10/10/2023	Nguyễn Phi Hùng	Thực tập sinh	Test các dạng câu hỏi, luồng chạy của chatbot, kết quả lưu dữ liệu vào kho dữ liệu; Góp ý, đưa thêm các trường hợp mở rộng, hướng nghiên cứu phát triển chatbot, cải thiện giao diện
11/10/2023	13/10/2023	Minh Thanh	Người thực hiện chính	Biên soạn tài liệu hướng dẫn cách sử dụng chatbot và đề cương source code

TÀI LIỆU THAM KHẢO

ChatGPT. (2021). *OpenAI*. Retrieved from ChatGPT: <https://chat.openai.com/>

PasonaVN. (2022). *Pasona*. Retrieved from About us: <https://pasona.vn/about-us>

Quang, P. H. (2018). *Viblo*. Retrieved from Hiểu hơn về BERT: Bước nhảy lớn của Google: <https://viblo.asia/p/hieu-hon-ve-bert-buoc-nhay-lon-cua-google-eW65GANOZDO>

Quang, P. H. (2020). *Viblo*. Retrieved from BERT, RoBERTa, PhoBERT, BERTweet: Ứng dụng state-of-the-art pre-trained model cho bài toán phân loại văn bản: <https://viblo.asia/p/bert-roberta-phobert-bertweet-ung-dung-state-of-the-art-pre-trained-model-cho-bai-toan-phan-loai-van-ban-4P856PEWZY3>

trituenhantao. (2019). *trituenhantao.io*. Retrieved from Hướng dẫn Fine-Tuning BERT với PyTorch: <https://trituenhantao.io/lap-trinh/huong-dan-fine-tuning-bert-voi-pytorch/>

PHÊ DUYỆT VÀ NHẬN XÉT CỦA CÔNG TY VÀ NGƯỜI HƯỚNG DẪN ĐẠI DIỆN DOANH NGHIỆP

OVERALL EVALUATION:

Consider the ratings for all of the personal and performance elements above (Đánh giá chung cho tất cả các yếu tố trên thang điểm 10).

Unsatisfactory (Không đạt yêu cầu)			Poor (Kém)		Average (Trung bình)		Good (Khá)		Outstanding (Giỏi)	
0	1	2	3	4	5	6	7	8	9	10

Strengths of intern (Ưu điểm của thực tập viên):


- Trình độ học hỏi cao: Thực tập viên thể hiện tinh thần học hỏi tích cực, luôn nắm bắt kiến thức mới và không ngại ngại hỏi khi khó khăn.
- Hòa đồng, giao tiếp tốt với đồng nghiệp: Thực tập viên hòa đồng và có khả năng giao tiếp tốt với đồng nghiệp giúp tạo ra môi trường làm việc thân thiện và hiệu quả.
- Biết cách đặt vấn đề cần giải quyết: Thực tập viên biết cách đặt ra câu hỏi và vấn đề cần giải quyết một cách rành rọt, giúp tăng hiệu suất và tập trung vào mục tiêu công việc.
- Kỹ năng xử lý lỗi tốt: Thực tập viên thể hiện khả năng xử lý lỗi tốt, giúp nhanh chóng khôi phục sự cố trong quá trình làm việc.

Areas for improvement (Những điều thực tập viên cần cải thiện):

- Cần cải thiện khả năng tính toán thời gian công việc: Thực tập viên cần cải thiện khả năng ước tính thời gian công việc một cách chính xác hơn để đảm bảo hoàn thành nhiệm vụ đúng hạn và quản lý thời gian hiệu quả hơn.

Date: 16/10/2023

Supervisor's Signature


Nguyễn Phi Hùng



Please send this form to us (Vui lòng gửi bản đánh giá này đến):

Khoa CNTT Kinh Doanh – Đại học Kinh tế TP.HCM

Lầu 9 – 279 Nguyễn Tri Phương, Phường 5, Quận 10, TP.HCM

Email: bjt@ueh.edu.vn . Tel: +84-283-526-5816



Tu Thi Son
Director