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

LUẬN VĂN NGÀNH
KHOA HỌC MÁY TÍNH

Đề tài

Trợ lý số tư vấn khóa học tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ
Can Tho University Software Center Digital Counselor

Sinh viên thực hiện: Nguyễn Hưng
Mã số: B1709536
Khóa: 43

Cần Thơ, 12/2021

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

LUẬN VĂN NGÀNH
KHOA HỌC MÁY TÍNH

Đề tài

Trợ lý số tư vấn khóa học tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ
Can Tho University Software Center Digital Counselor

Giáo viên hướng dẫn:
TS. Lưu Tiến Đạo
TS. Trương Xuân Việt

Sinh viên thực hiện:
Nguyễn Hưng

Mã số: B1709536
Khóa: 43

Cần Thơ, 12/2021

NHẬN XÉT CỦA GIẢNG VIÊN

LỜI CẢM ƠN

Lời đầu tiên, em xin chân thành cảm ơn thầy Lưu Tiến Đạo và thầy Trương Xuân Việt, thầy đã tận tình hướng dẫn, cung cấp những kiến thức quý báu cho em trong suốt quá trình thực hiện luận văn tốt nghiệp. Cảm ơn thầy đã tạo điều kiện và cho em cơ hội để thực hiện luận văn này và áp dụng vào thực tế như một giải pháp giúp ích cho các tư vấn viên. Cảm ơn sự tâm huyết của thầy trong quá trình trình giảng dạy, ngọn lửa nhiệt huyết của thầy truyền lại sẽ là một hành trang cho em vững bước trên con đường mưu cầu tri thức.

Em xin gửi lời cảm ơn chân thành đến các Thầy cô Khoa Công nghệ thông tin và Truyền thông Trường Đại học Cần Thơ. Qua sự tận tâm trong quá trình giảng dạy cũng như qua các buổi học của quý thầy cô đã cung cấp cho em rất nhiều kiến thức nền tảng vững chắc để tiếp tục nghiên cứu và tự học hỏi sau này.

Em xin cảm ơn Trung tâm Công nghệ phần mềm Đại học Cần Thơ đã tin tưởng và hỗ trợ em trong suốt thời gian thực hiện đề tài này.

Cuối cùng, em xin gửi lời cảm ơn đến gia đình, người thân và những người bạn thường xuyên quan tâm, động viên, chia sẻ kinh nghiệm để giúp em hoàn thành tốt luận văn này.

Mặc dù đã cố gắng và nỗ lực rất nhiều trong quá trình thực hiện luận văn. Tuy nhiên, do kiến thức và kĩ năng còn hạn chế nên khó tránh những sai sót. Em rất mong được sự thông cảm, góp ý của quý thầy cô để luận văn được hoàn thiện hơn.

Cần Thơ, ngày tháng năm 2021

Người viết

Nguyễn Hưng

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

DANH MỤC TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Diễn giải
AI	Artificial Intelligence	Trí tuệ nhân tạo
BOW	Bag of Words	Mô hình túi đựng từ
CDD	Conversation Driven Development	Phát triển theo hướng hội thoại
CRF	Conditional Random Fields	Mô hình xác suất trường điều kiện ngẫu nhiên
DIET	Dual Intent and Entity Transformer	Một kiến trúc SOTA trong lĩnh vực chatbot
DM	Dialogue Management	Quản lý hội thoại
FFNN	Feed Forward Neural Network	Mạng neural truyền thẳng
FSA	Finite State Automata	Mô hình dựa trên máy trạng thái hữu hạn
HMM	Hidden Markov Models	Mô hình Markov ẩn
LSTM	Long short-term memory	Mạng cải tiến giải quyết vấn đề phụ thuộc quá dài
MVC	Model-View-Controller	Mẫu kiến trúc phần mềm
NLG	Natural Language Generation	Thành phần sinh ngôn ngữ
NLP	Natural Language Processing	Xử lý ngôn ngữ tự nhiên
NLU	Natural Language Understanding	Hiểu ngôn ngữ tự nhiên
PE	Position Encoding	Kỹ thuật đánh dấu vị trí của từ trong câu
RNN	Recurrent Neural Network	Mạng Neural hồi quy
SOTA	State-Of-The-Art	Mức độ phát triển cao nhất của một công nghệ nào đó đạt được trong một khoảng thời gian nhất định
SPA	Single Page Application	Trang web mà ở đó người dùng có thể truy cập nhiều trang con mà không ảnh hưởng gì đến trang gốc.
SVM	Support Vector Machine	Máy vector hỗ trợ
TCP	Transmission Control Protocol	Giao thức điều khiển truyền tải
UI	User Interface	Giao diện người dùng
	Pretrained Model	Mô hình huấn luyện sẵn

TÓM TẮT

Hiện nay dịch vụ tư vấn, giải đáp thắc mắc trực tuyến về các khóa học và dịch vụ tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ là một phần quan trọng trong hoạt động chiêu sinh của trung tâm. Tuy nhiên không phải lúc nào tư vấn viên cũng trực tuyến. Trong luận văn này, tôi đề xuất giải pháp Trợ lý số tư vấn khóa học tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ. Hệ thống tư vấn này có khả năng tiếp nhận và trả lời một cách tự động 24/7 câu hỏi từ người dùng. Sau đó phân lớp ý định, trích xuất thực thể, từ đó trả về cho người dùng câu trả lời phù hợp nhất. Luận văn cũng giải quyết một thách thức lớn đối với hệ thống chatbot đó là phân lớp đa ý định khi mà trong một câu nói của người dùng có thể mang nhiều hơn một ý định. Hệ thống sử dụng Rasa để xây dựng chatbot cùng với việc tùy chỉnh tham số, bổ sung thành phần mới phù hợp với miền dữ liệu mà chatbot hỗ trợ. Hiện tại Trợ lý số có thể tư vấn 23 khóa học tại Trung tâm Công nghệ phần mềm và một số vấn đề liên quan khác. Kết quả thử nghiệm cho kết quả khả quan khi đạt F1 Score là 0.94. Độ chính xác này sẽ còn được cải thiện theo thời gian khi số lượng câu hỏi đủ lớn. Vì vậy, giải pháp đề xuất này sẽ mở ra một hướng đi đáng được quan tâm trong việc tư vấn, giải đáp thắc mắc tại trung tâm.

Từ khóa: chatbot Tiếng Việt, kiến trúc DIET, cải tiến các thành phần Rasa, đường ống Rasa.

ABSTRACT

Currently, the counseling service and answering online questions on courses at the Can Tho University Software Center is an important part of the enrollment activity. However, the problem is that consultants are not always ready. In my dissertation, I proposed a digital assistant for courses at the Can Tho University Software Center. This consulting system is capable of automatically receiving and answering questions 24/7 from users. It then categorizes intent, extracts the entity, and then returns the user the most appropriate response. Dissertation also addresses a major challenge to chatbot systems which is multipurpose categorization when users have more than one intent. The subject uses Rasa to build the chatbot along with customizing the parameters, adding new components to match the data domain that the chatbot supports. Currently the digital assistant can consult 23 courses at the Can Tho University Software Center and other related issues. The test result was a F1 score of 0.94. This accuracy also improves over time when the number of questions is sufficiently large. Therefore, the proposed solution will usher in a remarkable way in consultancy and answer at center.

Keywords: Vietnamese chatbot, DIET architecture, improving Rasa components, Rasa pipeline.

PHẦN GIỚI THIỆU

1. Đặt vấn đề

Văn hóa là một đặc điểm xã hội đặc trưng của con người. Văn hóa là tập hợp các hoạt động và sáng tạo phi sinh học của con người. Nó được truyền tải qua hướng dẫn và học hỏi. Con người tiếp thu văn hóa từ cha mẹ, người thân, những người xung quanh và từ toàn nhân loại. Quá trình kế thừa văn hóa giúp con người truyền tải những kinh nghiệm tích lũy trong suốt các thế hệ. Thích nghi văn hóa hiệu quả hơn hẳn thích ứng sinh học vì nó nhanh hơn, có thể thích ứng và tích lũy qua các thế hệ. Một chế độ được gọi là tiến hóa siêu hữa cơ.

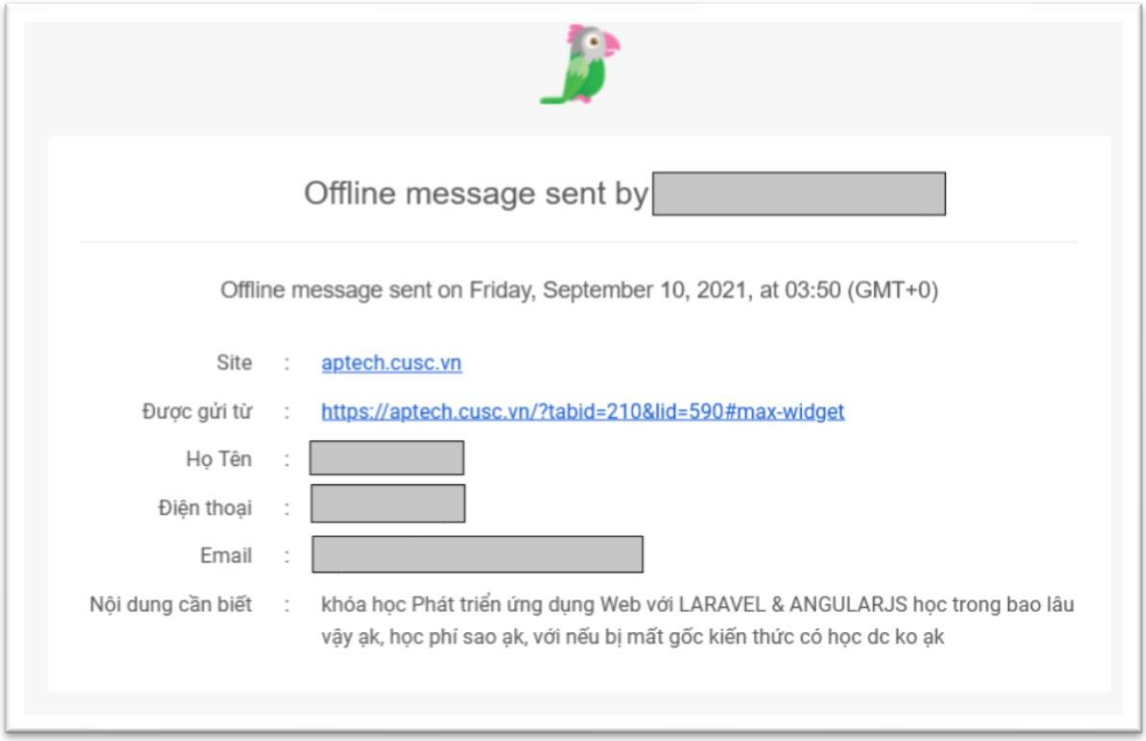
Quá trình phát triển của Internet đã thúc đẩy quá trình tiến hóa siêu hữu cơ khi giờ đây chúng ta không cần chia sẻ kiến thức và kinh nghiệm đến từng cá thể mà thay vào đó chỉ cần chia sẻ với máy tính và lượng thông tin đó sẽ được chia sẻ với tất cả mọi người trên khắp thế giới thông qua nhiều hình thức khác nhau: văn bản, âm thanh, hình ảnh, …. Trong suốt thập kỉ vừa qua nhờ có sự phát triển vượt bậc trong lĩnh vực Trí tuệ nhân tạo nói chung và Xử lý ngôn ngữ tự nhiên nói riêng việc trao đổi thông tin một cách "người" hơn đã trở thành hiện thực khi giờ đây con người đã có thể giao tiếp với máy bằng ngôn ngữ tự nhiên thông qua văn bản hay thậm chí là giọng nói. Chatbot được xây dựng dựa trên nền tảng như thế, chatbot giúp con người quản lý các cuộc trò chuyện, tương tác với người dùng thông qua ngôn ngữ tự nhiên.

Theo báo cáo của Grand View Research năm 2021[1], thị trường chatbot toàn cầu dự kiến sẽ đạt 2.5 triệu USD vào năm 2028. Thị trường chatbot sẽ phát triển mạnh mẽ trong các lĩnh vực tài chính, chăm sóc khách hàng. Chatbot đã giúp các doanh nghiệp giảm đáng kể chi phí vận hành (các kênh tư vấn trực tuyến) cũng như mang lại giá trị to lớn trong trải nghiệm khách hàng.

Chatbot là một bước tiến đột phá trong cách con người tương tác với dữ liệu. Chatbot mang đến trải nghiệm dễ dàng, nhanh chóng và tiện lợi trong việc tương tác giữa người và máy. Giờ đây người dùng không cần phải chờ đợi khi các tư vấn viên bận hoặc không trực tuyến. Thay vào đó chỉ cần một thiết bị có kết nối Internet, người dùng đã có thể tương tác với chatbot và nhận được sự hỗ trợ kịp thời, hiệu quả như tương tác với người thật. Tại thời điểm hiện tại chatbot đã không còn xa lạ với người dùng khi hiện nay có rất nhiều lĩnh vực triển khai và phát triển thành công chatbot như: du lịch, nhà hàng, mua sắm, …. Nhưng chưa có nhiều trợ lý ảo hỗ trợ tư vấn các khóa học và dịch vụ cụ thể nào đó. Vì đây là một miền dữ liệu đóng (Ví dụ mỗi trung tâm có thể có các khóa học, dịch vụ và học phí khác nhau). Hình i dưới đây là một tin nhắn được gửi đến hệ thống tư vấn

[1] https://www.grandviewresearch.com/press-release/global-chatbot-market

trực tuyến tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ vào thời điểm các tư vấn viên không trực tuyến.



Hình i: Câu hỏi từ người dùng được gửi đến hệ thống tư vấn trực tuyến

Xuất phát từ nhu cầu thiết yếu đã trình bày ở trên, tôi thực hiện đề tài luận văn "Trợ lý số tư vấn khóa học tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ" bằng việc nghiên cứu và áp dụng Rasa framework [1] vào việc xây dựng chatbot hỗ trợ tư vấn khóa học, dịch vụ hiện có tại Trung tâm công nghệ phần mềm Đại học Cần Thơ. Một trong những thách thức trong việc xây dựng chatbot là có thể xử lý các câu hỏi đa ý định từ người dùng. Vì trong ngôn ngữ giao tiếp thường ngày, một câu nói có thể chứa nhiều hơn một ý định như câu hỏi mà người dùng đặt ra ở Hình i. Việc chatbot có thể hiểu và trả lời được câu hỏi loại này sẽ giúp việc tương tác giữa người dùng và chatbot sẽ tự nhiên hơn. Bên cạnh đó nhằm hỗ trợ cho việc nhập và truy xuất dữ liệu học phí của chatbot, đề tài đã xây dựng một module website Quản lý học phí dựa trên mô hình Model-View-Controller (MVC).

2. Lịch sử giải quyết vấn đề

Đã có rất nhiều đề tài nghiên cứu trong lĩnh vực chatbot điển hình như:

- "Chatbot cho sinh viên Công nghệ thông tin" (Đỗ Thanh Nghị, Hoàng Tùng) [2] đây là một hệ thống trả lời tự động các câu hỏi liên quan đến môi trường học tập, phương pháp học tập bậc đại học, kĩ năng nghề nghiệp, xu hướng công nghệ dành riêng cho sinh viên ngành Công nghệ

thông tin. Hệ thống đã mang đến cho sinh viên khối ngành Công nghệ thông tin một nguồn thông tin dồi dào, hữu ích và dễ dàng tiếp cận đối với các vấn đề liên quan đến ngành học.

- "Hệ thống hỗ trợ tư vấn tuyển sinh" (Nguyễn Thái Nghe, Trương Quốc Định) [3] đây là một hệ thống phân loại câu hỏi và gửi đến các chuyên gia, dựa trên các câu hỏi đã được phân loại, các chuyên gia sẽ gửi câu trả lời về cho người dùng. Đề tài đã mở ra hướng đi mới trong hỗ trợ tư vấn tuyển sinh một cách tự động.
- "Chatbot bán hàng tự động iBotSale" (Đỗ Viết Mạnh, Trần Đức Nghĩa, Nguyễn Việt Anh, Hà Thị Hồng Vân, 2020) [4] đây là một hệ thống hỗ trợ trả lời được các câu hỏi liên quan đến sản phẩm mà người dùng quan tâm như: giá cả, kích thước, số lượng, …. Đề tài đã đưa ra giải pháp bán hàng tự động thông qua chatbot một cách tự nhiên hơn thay vì sử dụng mô hình truyền thống Menu/Button.
- "Codebot" (Vương Lê Minh Nguyên, Lương Công Tâm, Nguyễn Viết Hưng, Nguyễn Đỗ Thái Nguyên, Lương Trần Hy Hiến, Lương Trần Ngọc Khiết, Phan Thị Trinh) [5] đây là một hệ thống chatbot hỗ trợ giáo dục, chatbot có khả năng trả lời các câu hỏi liên quan đến lập trình C++ và Python bằng tiếng Việt. Đề tài đã giải quyết khó khăn đối với giới trẻ Việt Nam khi tiếp cận với ngôn ngữ lập trình. Khi các nguồn tài liệu đa số đều bằng tiếng Anh.
- "Chatbot hỗ trợ người dùng ngân hàng" (Nguyễn Tất Tiến) [6] đây là một hệ thống hỗ trợ tư vấn các câu hỏi liên quan đến ngân hàng như: kiểm tra số dư, vay tiền, gửi tiền, …. Đề tài đã mở ra hướng tiếp cận mới về cách thức giao dịch giữa người dùng và ngân hàng. Khi giờ đây người dùng có thể thực hiện giao dịch bằng ngôn ngữ tự nhiên.

Bên cạnh đó cũng đã có rất nhiều chatbot được triển khai ở nhiều lĩnh vực khác nhau như tư vấn khách hàng, đặt lịch khách sạn, …. Một số chatbot có thể kể đến như:

- *Poncho*: chatbot được thiết kế dùng riêng cho lĩnh vực thời tiết. Ngoài dự báo thời tiết chúng còn gửi cảnh báo khi thời tiết xấu với sự đồng ý của người dùng.
- *Mitsuku*: chatbot được xây dựng với mục đích tán gẫu, nói chuyện phiếm với người dùng.
- *Yeshi:* là một chatbot đại diện cho các cô gái trẻ ở Ethiopia, những người phải đi bộ 2.5 giờ mỗi ngày để tìm nước sạch. Khi ai đó trò chuyện với bot, Yeshi sẽ gửi hình ảnh, video, âm thanh và bản đồ để tạo trải nghiệm

cũng như cảm xúc sâu sắc giúp người người dùng hiểu được thực tế khắc nghiệt của người Ethiopia trong cuộc khủng hoảng nước là như thế nào.

Chatbot còn được sử dụng như một trợ lý ảo cá nhân. Điển hình như những trợ lý ảo: Siri, Cortana, ... Đã giúp người dùng đặt báo thức, truy vấn thông tin, đặt xe, ... một cách vô cùng tiện lợi như tương tác trực tiếp với con người.

3. Mục tiêu đề tài

3.1. Mục tiêu chung

Mục tiêu chung của đề tài là xây dựng được hệ thống tư vấn, giải đáp các câu hỏi liên quan đến các khóa học, dịch vụ đang được cung cấp tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ. Bên cạnh đó, hệ thống được xây dựng có khả năng phân loại và trả lời các câu hỏi đa ý định từ người dùng. Hệ thống cũng được xây dựng để quản trị viên dễ dàng quản lý các câu hỏi, câu trả lời thông qua giao diện web và file csv. Hệ thống sẽ được tích hợp vào hệ thống tư vấn trực tuyến hiện có ở trung tâm.

3.2. Mục tiêu cụ thể

- Tìm hiểu các thành phần của chatbot
- Thiết kế cơ sở dữ liệu lưu trữ học phí phục vụ việc trả lời của chatbot
- Thiết kế module website Quản lý học phí theo mô hình MVC
- Xây dựng ứng dụng chat bằng ReactJS
- Xây dựng mô hình phân loại ý định và trích xuất thực thể dựa trên Rasa framework

4. Đối tượng và phạm vi nghiên cứu

- Đối tượng nghiên cứu: tổng hợp những câu hỏi mà người dùng thường quan tâm khi trao đổi trực tuyến với tư vấn viên.
- Phạm vi nghiên cứu: khóa học ngắn hạn, khóa học dài hạn và các dịch vụ đang được cung cấp tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ

5. Phương pháp nghiên cứu

Lý thuyết:

- Tìm hiểu các khóa học và dịch vụ hiện đang có tại trung tâm
- Tìm hiểu các lý thuyết và kĩ thuật cơ bản trong Xử lý ngôn ngữ tự nhiên
- Tìm hiểu kiến trúc DIET hiện đang được sử dụng trên Rasa framework
- Tìm hiểu các pre-trained model như PhoBert, Fasttext, BPE nhằm áp dụng, đánh giá và tìm ra cấu hình phù hợp nhất cho miền dữ liệu mà chatbot đang xử lý
- Tìm hiểu mô hình MVC để xây dựng module website Quản lý học phí
- Tìm hiểu ReactJS để xây dựng ứng dụng chat thời gian thực

Thực hành:

- Tiếp nhận dữ liệu được cung cấp từ trung tâm
- Thu thập thêm dữ liệu về các khóa học
- Thu thập thêm dữ liệu từ người dùng
- Xây dựng cơ sở dữ liệu lưu trữ học phí của các khóa học
- Xây dựng mô hình và đánh giá độ chính xác
- Xây dựng module website Quản lý học phí
- Xây dựng giao diện chat thời gian thực giữa người dùng và bot

6. Kết quả đạt được

- Xây dựng được hệ thống chatbot hỗ trợ trả lời các câu hỏi liên quan đến 23 khóa học hiện có tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ
- Xử lý câu hỏi đa ý định từ người dùng
- Xây dựng module website Quản lý học phí hỗ trợ quản lý học phí cũng như là nguồn dữ liệu để chatbot trả lời các câu hỏi liên quan đến học phí
- Xây dựng ứng dụng chat trực tuyến giữa người dùng và chatbot

7. Bố cục luận văn

Phần giới thiệu

Giới thiệu tổng quát về đề tài

Phần nội dung

Chương 1: Mô tả bài toán

Chương 2: Thiết kế và cài đặt

Chương 3: Kiểm thử hệ thống và đánh giá độ chính xác

Phần kết luận

Trình bày kết quả đạt được và hướng phát triển

CHƯƠNG 1: MÔ TẢ BÀI TOÁN

1.1. Mô tả chi tiết bài toán

Chatbot là một chương trình máy tính tương tác với người dùng bằng ngôn ngữ tự nhiên dưới dạng văn bản hoặc giọng nói. Chatbot ngày nay được ứng dụng trong nhiều lĩnh vực khác nhau như: chăm sóc khách hàng, thương mại điện tử, đặt bàn nhà hàng, …. Để xây dựng được hệ thống chatbot cần tìm hiểu cách thức mà một hệ thống chatbot hoạt động là như thế nào.

Chatbot có thể chia thành 2 loại chính:

- Hệ thống chatbot phục vụ trên một miền dữ liệu mở (*open domain*): đây là mô hình trả lời tự động cho phép người dùng có thể trao đổi với chatbot ở bất kì lĩnh vực nào. Nhưng vì số lượng chủ đề mà người dùng hướng đến là không giới hạn nên việc có thể trả lời hợp lí các câu hỏi mà người dùng đặt ra cần đến rất nhiều hiểu biết và đây là một vấn đề khó khăn.
- Hệ thống chatbot phục vụ trên miền dữ liệu đóng (*closed domain*): các câu hỏi và trả lời được giới hạn ở một lĩnh vực cụ thể nào đó. Người dùng có thể hỏi những câu ngoài lĩnh vực mà chatbot có thể trả lời. Tùy tình huống hệ thống sẽ kéo người dùng về lại chủ đề chính hoặc nếu không thể xử lý sẽ chuyển người dùng tương tác trực tiếp với tư vấn viên.

Với bài toán tập trung vào miền đóng như trợ lý ảo tư vấn các khóa học, đề tài sẽ tự tạo tập dữ liệu huấn luyện riêng. Điều này cũng đảm bảo chatbot có thời gian huấn luyện ngắn và độ chính xác cao hơn.

Để xây dựng được chatbot có khả năng hiểu và trả lời bằng ngôn ngữ tự nhiên, trước hết cần phải nghiên cứu cách máy tính có thể hiểu ngôn ngữ tự nhiên. Tìm hiểu cách xây dựng chatbot bằng Rasa framework. Sau đó tiếp nhận dữ liệu được cung cấp bởi Trung tâm Công nghệ phần mềm Đại học Cần Thơ và thu thập thêm dữ liệu từ nhiều nguồn khác nhau nhằm xây dựng cơ sở dữ liệu học phí và cách chatbot trả lời với từng ý định cụ thể.

Tiếp đến sẽ phân tích và thiết kế ứng dụng chat và module website Quản lý học phí. Ứng dụng chat sẽ cho phép người người dùng tương tác với chatbot trong thời gian thực. Hệ thống chatbot nhận tin nhắn từ người dùng gửi đến. Tiến hành mã hóa, phân loại ý định và dựa vào ý định đã xác định được sẽ trả về câu trả lời phù hợp cho người dùng. Đối với module website Quản lý học phí cho phép quản trị viên có thể dễ dàng quản lý học phí của từng khóa học. Dữ liệu về học phí sẽ được hệ thống chatbot truy vấn để trả lời các câu hỏi có liên quan đến các vấn đề học phí mà người dùng quan tâm.

1.2. Vấn đề và giải pháp liên quan đến bài toán

1.2.1. Vấn đề

Các vấn đề chính để xây dựng hệ thống chatbot bao gồm:

- Hiểu được câu hỏi của người dùng và đưa ra câu trả lời chính xác
- Thu thập bộ câu hỏi đủ lớn để đạt độ chính xác tối ưu
- Xây dựng Cơ sở dữ liệu (CSDL) và module website Quản lý học phí nhằm phục vụ việc trả lời các câu hỏi liên quan đến học phí
- Xây dựng ứng dụng chat trực tuyến

1.2.2. Giải pháp

- Sử dụng RASA framework để xử lý vấn đề hiểu ngôn ngữ tự nhiên và phát sinh câu trả lời
- Sử dụng ngôn ngữ lập trình PHP và mô hình MVC để xây dựng module website Quản lý học phí
- Sử dụng ReactJS để xây dựng ứng dụng chat thời gian thực giữa người dùng và bot
- Sử dụng Rasa X để quản lý các tin nhắn đến từ người dùng, từ đó đánh giá và gán nhãn nhằm cải thiện hiệu suất của chatbot

1.2.3. Các thư viện và công cụ hỗ trợ

1.2.3.1. ReactJS

ReactJS là một thư viện JavaScript có tính hiệu quả và linh hoạt để xây dựng các thành phần giao diện người dùng (UI) có thể sử dụng lại. ReactJS giúp phân chia các UI phức tạp thành các thành phần nhỏ (*được gọi là component*). Nó được tạo ra bởi Jordan Walke, một kỹ sư phần mềm tại Facebook. ReactJS ban đầu được phát triển và duy trì bởi Facebook và sau đó được sử dụng trong các sản phẩm của mình như WhatsApp & Instagram.

ReactJS được dùng để xây dựng các ứng dụng *single page application* (*SPA*). Một trong những điểm hấp dẫn của ReacJS là nó không chỉ được xây dựng bên phía clients mà còn sử dụng được bên phía server.

1.2.3.2. Rasa

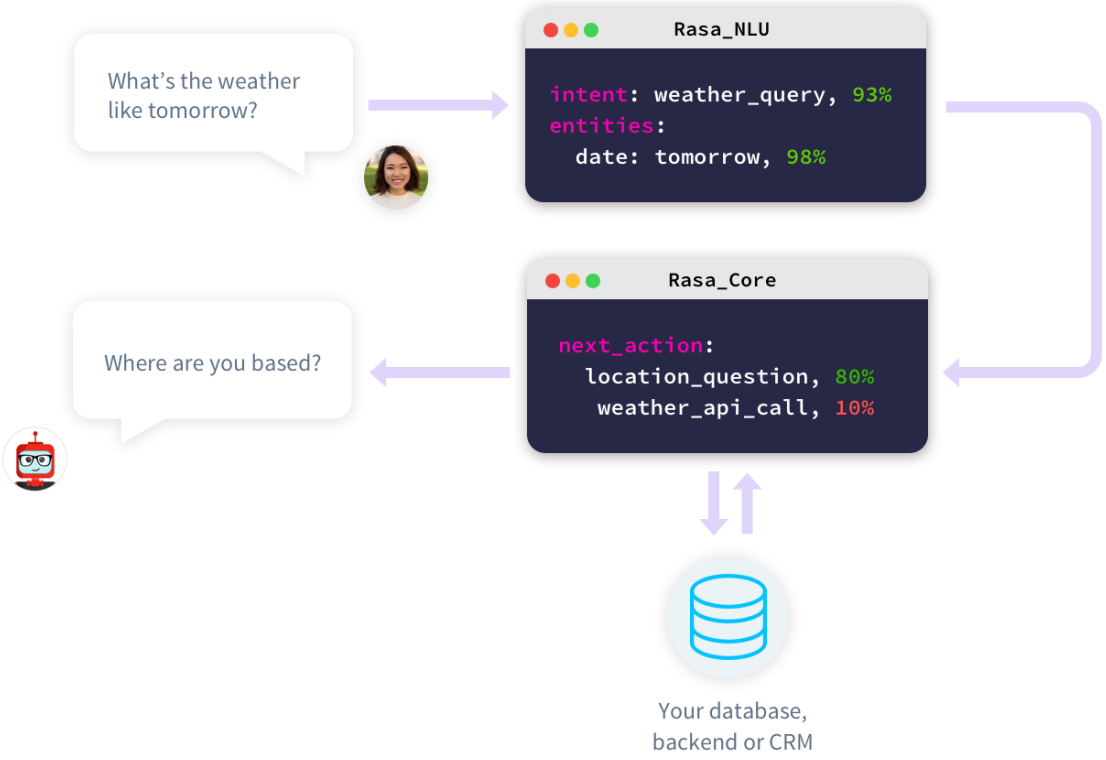
Rasa là một nền tảng mở nguồn mở hỗ trợ xây dựng chatbot một cách nhanh chóng và hiệu quả. Rasa bao gồm các thành phần chính sau:

- **Rasa NLU** - một thư viện để hiểu ngôn ngữ tự nhiên (NLU) thực hiện phân loại ý định và trích xuất thực thể từ đầu vào của người dùng và giúp bot hiểu những gì người dùng đang nói.
- **Rasa Core** - một khung chatbot với quản lý hội thoại , lấy đầu vào có cấu trúc từ NLU. NLU và Core là độc lập và người ta có thể sử dụng NLU mà không cần Core, và ngược lại.
- **Rasa X**: công cụ giúp chúng ta xây dựng, cải thiện và triển khai model chatbot vừa tạo.

CHƯƠNG 2: THIẾT KẾ VÀ CÀI ĐẶT

2.1. Tổng quan hệ thống chatbot

2.1.1. Kiến trúc tổng thể của Rasa



Hình 2.1: Kiến trúc tổng quát của Rasa

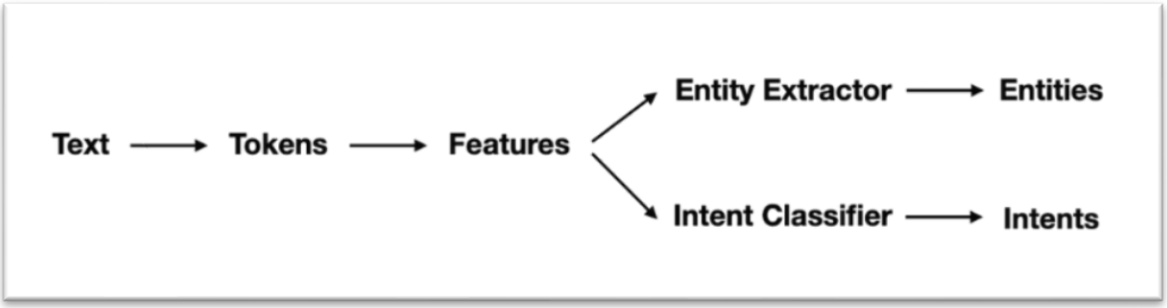
Hình 2.1 mô tả các thành phần cốt lõi của Rasa [7] bao gồm:

- **Rasa NLU** (NLU - Natural Language Understanding): đây là thành phần rất quan trọng trong hệ thống chatbot, chịu trách nhiệm phân loại ý định, trích xuất thực thể.
- **Rasa Core** (DM - Dialogue Management): đây là nơi thực hiện quản lý luồng hội thoại. Dựa vào các ý định và thực thể đã xác định ở thành phần NLU quyết định hành động tiếp theo cũng như tin nhắn sẽ phản hồi cho người dùng.

2.1.2. Các thành phần cơ bản của một hệ thống chatbot

2.1.2.1. Thành phần NLU

Đây là thành phần quan trọng nhất của một hệ thống chatbot. Việc đầu tiên cũng là cốt yếu nhất của một hệ thống chatbot cần phải làm đó là hiểu được ý định của người dùng. Chỉ khi phân loại được ý định của người dùng thì mới có thể đưa ra câu trả lời phù hợp.



Hình 2.2: Quá trình xử lý một tin nhắn đầu vào từ người dùng

Thành phần NLU xử lý tin nhắn người dùng gửi đến bằng chuỗi các hành động nối tiếp nhau một cách tuần tự theo hình ống (pipeline), với kết quả của đầu ra của hành động trước đó là đầu vào của hành động tiếp theo. Hình 2.2 ở trên mô tả pipeline mà thành phần Rasa NLU đang áp dụng.

2.1.2.1.a. Tách từ (Tokenizer)

Tách từ là một quá trình nhằm xác định ranh giới giữa các từ. Việc chia một câu văn bản thành các từ riêng lẻ được gọi là các words (hoặc tokens). Mục tiêu của việc tách từ là để khử tính nhập nhằng về ngữ nghĩa của văn bản. Tùy vào từng ngôn ngữ có những đặc điểm khác nhau sẽ có cách tách từ khác nhau. Dựa theo đặc điểm của ngôn ngữ tự nhiên mà ngôn ngữ được phân thành các loại như sau:

- *Ngôn ngữ hòa kết*: đặc điểm của ngôn ngữ này là từ sẽ biến đổi hình thái để thay đổi ý nghĩa và ngữ pháp. Trong tiếng Anh từ "cats" sẽ là hình thái số nhiều của "cat". Một số ngôn ngữ đại diện cho loại này như: tiếng Anh, tiếng Nga, tiếng Hy Lạp, …
- *Ngôn ngữ cô lập*: đặc điểm là mỗi từ độc lập không biến đổi hình thái, nếu muốn thay đổi ý nghĩa thì có thể thêm từ ghép với từ gốc. Vị trí của từ trong câu cũng sẽ mang đến sắc thái ý nghĩa khác nhau. Ví dụ: "cửa trước" và "trước cửa" sẽ mang ý nghĩa hoàn toàn khác nhau. Một số ngôn ngữ đại diện cho loại này như: tiếng Việt, tiếng Trung,…
- *Ngôn ngữ chắp dính*: Nằm giữa ngôn ngữ cô lập và ngôn ngữ hoà kết, đặc điểm của ngôn ngữ chắp dính là từ gốc không biến đổi tuy nhiên từ phái sinh của nó được cấu thành bằng cách gắn thêm phụ tố vào. Một số ngôn ngữ đại diện cho loại này bao gồm: tiếng Hàn, tiếng Nhật, …
- *Ngôn ngữ đa tổng hợp*: Đặc điểm là từ không tách riêng với câu hay có hiện tượng 1 từ mà mang nghĩa của cả 1 câu. Ngôn ngữ đại diện cho loại này là: tiếng Chinuk, tiếng Yupik, …

Đối với ngôn ngữ tiếng Việt ranh giới từ không chỉ là những khoảng trắng mà còn là là sự kết hợp giữa các từ với nhau tạo nên từ đơn hoặc từ ghép.

Ví dụ: đối với câu "Trường đại học Cần Thơ" sau khi tách từ sẽ trở thành "Trường đại_học Cần_Thơ"

Hiện nay có nhiều thư viện hỗ trợ việc tách từ cho tiếng Việt như: underthesea, VnCoreNLP, …

2.1.2.1.b. Trích xuất đặc trưng (Featurizer)

Để phân loại được ý định của người dùng ta cần mô hình hóa ngôn ngữ tức là biểu diễn ngôn ngữ dưới dạng vector số học mà máy có thể hiểu được. Phương pháp phổ biến nhất hiện nay là word embedding (nhúng từ). Tập nhúng từ là tên chung cho một tập hợp các mô hình ngôn ngữ và phương pháp học đặc trưng trong xử lý ngôn ngữ tự nhiên, nơi các từ hoặc cụm từ được ánh xạ đến một vector số thực. Một số phương pháp biểu diễn phổ biến như Word2Vec, GloVe, FastText, …

Trong Rasa sau khi tách từ, các từ sẽ được trích xuất đặc trưng. Rasa có 2 loại vector biểu diễn đặc trưng cho một từ bao gồm:

- *Sparse features*: loại bỏ các số 0 trong vector và chỉ lưu trữ các vị trí của các phần tử khác 0 và giá trị tương ứng. Bằng cách này, có thể tiết kiệm rất nhiều bộ nhớ máy tính.
- *Dense features*: ngược lại với sparse features các số 0 vẫn được lưu trữ. Vector đặc trưng này thường được sinh ra từ các pre-trained embedding.

2.1.2.1.c. Phân lớp ý định (Intent classification)

Sau khi mô hình hóa ngôn ngữ thì việc xác định ý định người dùng dựa trên câu hỏi của người dùng được gọi là phân loại ý định hay phân lớp văn bản. Ở bước này có thể dùng một số kĩ thuật như: Decision Tree, Vector Support Machine (SVM), Recurrent Neural Network (RNN), Long-Short Term Memory (LSTM), …. Hầu hết các chatbot hiện nay đều ứng dụng mô hình deep learning như RNN hay LSTM để phân loại ý định người dùng. Thách thức lớn nhất cho các chatbot ở bước này là xác định đa ý định (multi intent) trong một câu nói của người dùng. Vì trong ngôn ngữ giao tiếp thường ngày, một câu nói có thể chứa nhiều hơn một ý định. Ví dụ nếu người dùng hỏi "*Khóa học Chuyên gia Trí tuệ nhân tạo có học phí bao nhiêu và lịch học cụ thể như thế nào*" thì chatbot phải xác định được 2 ý định từ người dùng đó là "*học phí*" và "*lịch học*" trong câu nói trên. Việc chatbot có thể hiểu và trả lời được câu hỏi loại này sẽ giúp việc tương tác giữa người dùng và chatbot sẽ tự nhiên hơn.

Đối với yêu cầu của đề tài một miền dữ liệu đóng, số lượng ý định của người dùng nằm trong một tập hữu hạn những ý định đã được định nghĩa sẵn,

có liên quan đến lĩnh vực mà chatbot hỗ trợ. Vì vậy, bài toán phân lớp ý định người dùng có thể qui về bài toán phân lớp văn bản. Với đầu vào là tin nhắn của người dùng, hệ thống phân lớp sẽ xác định ý định tương ứng với câu đó trong tập các ý định đã được định nghĩa trước. Tuy nhiên vì trong ngôn ngữ tự nhiên mỗi ý định sẽ có cách diễn đạt khác nhau. Vì vậy cần phải thu thập tập dữ liệu bao gồm nhiều cách diễn đạt khác nhau cho mỗi ý định.

Ví dụ: đối với ý định hỏi học phí khóa học Lập trình viên quốc tế sẽ có nhiều cách hỏi khác nhau như sau:

- Học phí khóa học Lập trình viên quốc tế bao nhiêu vậy ạ?
- Học phí khóa này bao nhiêu vậy ạ?
- Ngành này đăng kí học tốn bao nhiêu tiền vậy ạ?

Có thể thấy việc thu thập dữ liệu cho bài toán phân loại ý định là một trong những công việc quan trọng nhất khi phát triển hệ thống chatbot và ảnh hưởng lớn tới chất lượng chatbot khi triển khai sau này. Công việc này cũng đòi hỏi nhiều thời gian và công sức đối với nhà phát triển chatbot.

Bên cạnh đó thách thức đối với nhà phát triển chatbot là đôi khi người dùng có thể hỏi những câu không thể lường trước được. Vì vậy cần nhà phát triển chatbot cần phải liên tục kiểm tra và đánh giá hiệu quả của chatbot. Phát triển theo hướng hội thoại (Conversation Driven Development - CDD) là quá trình "lắng nghe" người dùng và sử dụng những thông tin đó để cải thiện chatbot. Đây được xem là cách tiếp cận thực tiễn tốt nhất để phát triển chatbot.

Bằng cách thực hiện CDD ở mọi giai đoạn phát triển chatbot, nhà phát triển có thể định hướng chatbot của mình hướng tới ngôn ngữ và hành vi của người dùng.

Phát triển theo hướng hội thoại bao gồm:

- **Chia sẻ (share):** đưa chatbot vào thực tiễn ngay khi có thể
- **Đánh giá (review):** đánh giá các cuộc hội thoại thường xuyên
- **Chú giải (annotate):** chú giải các câu của người dùng và sử dụng chúng như một phần của dữ liệu huấn luyện
- **Kiểm thử (test):** kiểm thử thường xuyên để đảm bảo chatbot luôn hoạt động như mong đợi
- **Theo dõi (track):** theo dõi khi nào chatbot xảy ra lỗi và đo lường hiệu suất của nó theo thời gian
- **Khắc phục (fix):** xử lý các cuộc hội thoại không thành công

2.1.2.1.d. Xác định thực thể (Entity extraction)

Thực thể (entity) là một mẫu thông tin được lấy ra từ tin nhắn của người dùng. Một số thực thể có giá trị có thể trích xuất như:

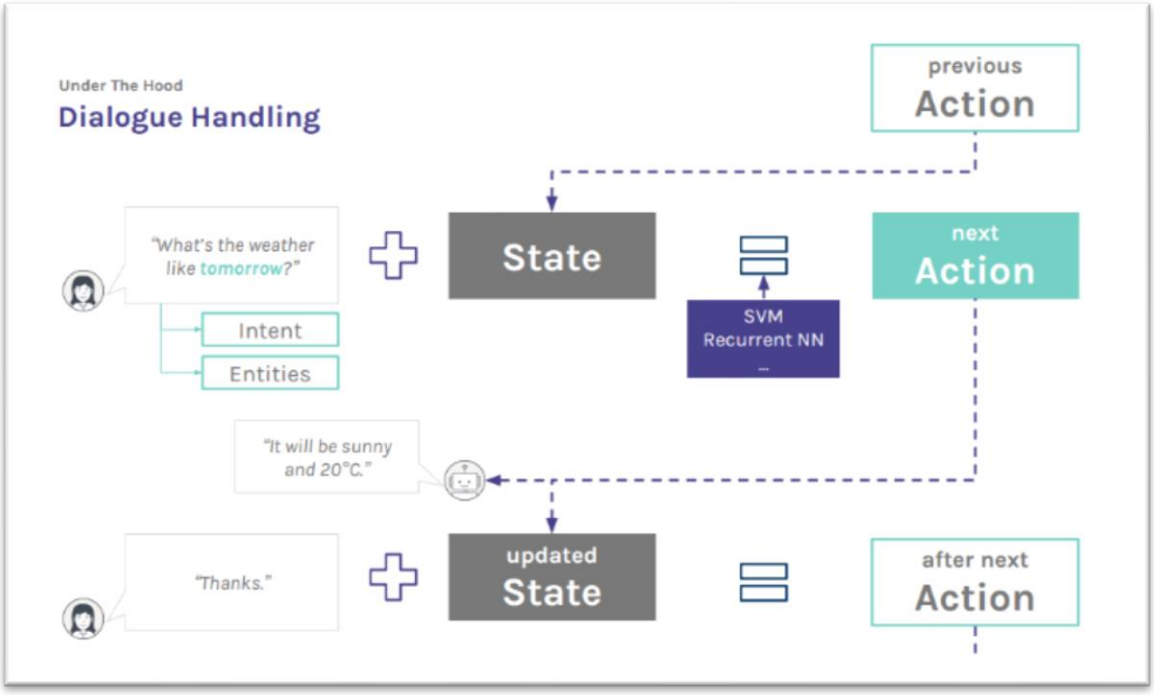
- Vị trí
- Số điện thoại
- Địa chỉ liên lạc
- Khóa học
- Họ tên

Việc chọn lựa thực thể nào cần trích xuất dựa trên chatbot cần dữ liệu nào để làm tiền đề cho các bước xử lý tiếp theo nhằm đưa ra câu trả lời và hành động phù hợp với ý định của người dùng. Việc trích xuất thực thể sẽ xác định được vị trí bắt đầu và vị trí kết thúc của thực thể. Cách tiếp cận phổ biến cho bài toán trích xuất thực thể là mô hình hóa bài toán thành bài toán gán nhãn chuỗi (sequence labeling). Đầu vào của bài toán gán nhãn chuỗi là một dãy các từ và đầu ra là một dãy các nhãn. Một số thuật toán huấn luyện mô hình gán nhãn chuỗi có thể kể đến như: mô hình CRF (Conditional Random Fields), mô hình Markov ẩn (HMM – Hidden Markov Model).

2.1.2.2. Thành phần quản lý hội thoại (Dialogue management)

Trong các phiên hội thoại dài (long conversation) giữa người dùng và chatbot, chatbot sẽ cần ghi nhớ những thông tin về ngữ cảnh (context) và quản lý các trạng thái hội thoại (dialog state). Vấn đề quản lý hội thoại (dialogue management) trong một cuộc hội thoại dài là rất quan trọng vì nó sẽ đảm bảo việc trao đổi giữa người và máy thông suốt.

Chức năng của thành phần quản lý hội thoại là nhận đầu vào từ thành phần NLU, quản lý các trạng thái hội thoại (dialog state), ngữ cảnh hội thoại (dialogue context) và truyền đầu ra cho thành phần sinh ngôn ngữ (Natural Language Generation, viết tắt là NLG).



Hình 2.3: Mô hình quản lý trạng thái hội thoại

Xem Hình 2.3 ở trên có thể thấy thành phần quản lý hội thoại dựa trên ý định (intent) và thực thể (entity) nhận được từ thành phần NLU thêm vào đó là hành động trước đó để quyết định hành động tiếp theo nào nên được thực hiện.

Hiện nay, các sản phẩm chatbot thường dùng mô hình trạng thái hữu hạn (Finite State Automata – FSA), mô hình Frame-based (Slot Filling) hoặc kết hợp cả hai mô hình này.

2.1.2.2.a. Mô hình trạng thái hữu hạn FSA

Mô hình FSA là mô hình quản lý hội thoại đơn giản nhất. Tại mỗi trạng thái chatbot sẽ đưa ra câu hỏi và chỉ nhận câu trả lời cho câu hỏi đó trong khi bỏ qua các thông tin không liên quan. Ví dụ một hệ thống mua bán hàng online, khi người dùng muốn đặt một món hàng nào đó. Hệ thống chatbot sẽ hỏi các câu hỏi như tên, số điện thoại, địa chỉ (xem Hình 2.4).

Hình 2.4: Mô hình quản lý hội thoại FSA

Như đã thấy ở Hình 2.4 các trạng thái FSA tương ứng với các hành động mà câu hỏi sẽ thực hiện. Các hành động này phụ thuộc vào phản hồi của người dùng ở mỗi câu hỏi.

Ưu điểm của mô hình FSA là chatbot sẽ định dạng trước câu trả lời mong muốn từ phía người dùng. Tuy nhiên vấn đề của mô hình này là khi người dùng đưa ra nhiều thông tin khác nhau trong một câu hội thoại. Giả sử khi chatbot hỏi "*Xin quí khách cho biết tên ạ*" mà người dùng trả lời đầy đủ thông tin của các câu hỏi sau đó như "*Tên tôi là Nguyễn Văn A số điện thoại là 0xxxxxxxx. Tôi ở địa chỉ 123, đường ABC*" thì chatbot chỉ chấp nhận câu trả lời "*Tên tôi là Nguyễn Văn A*" và bỏ qua thông tin số điện thoại và địa chỉ mà người dùng đã cung cấp. Ngay sau đó, chatbot lại hỏi thông tin về số điện thoại của người dùng, việc bị hỏi lại các thông tin vừa cung cấp sẽ khiến người dùng cảm thấy khó chịu.

2.1.2.2.b. Mô hình Frame-based

Mô hình Frame-based (hay có tên gọi khác là Form-based) có thể giải quyết vấn đề mà mô hình FSA gặp phải. Mô hình Frame-based dựa trên các frame được định nghĩa sẵn để định hướng cuộc hội thoại. Mỗi frame sẽ bao

gồm các thông tin (slot) cần điền và các câu hỏi yêu cầu người dùng cung cấp thông tin của các slot đó.

Thay vì cứng nhắc hỏi từng thông tin một cách tuần tự như mô hình FSA. Mô hình Frame-based cho phép người dùng trả lời nhiều câu hỏi cùng một lúc.

Thành phần quản lý của mô hình Frame-based sẽ đưa câu hỏi cho người dùng, hệ thống sẽ điền vào các slot dựa vào các thông tin mà người dùng cung cấp cho đến khi đủ thông tin cần thiết. Khi người dùng cung cấp nhiều thông tin cùng một lúc, hệ thống sẽ điền vào các slot tương ứng và ghi nhớ để không hỏi lại những câu hỏi đã có câu trả lời.

Trong các miền ứng dụng phức tạp, một cuộc hội thoại có thể có nhiều frame khác nhau. Vấn đề đối với nhà phát triển chatbot là biết khi nào cần chuyển đổi giữa các frame. Cách tiếp cận phổ biến để quản lý việc chuyển đổi giữa các frame là định nghĩa các luật. Các luật này dựa trên các thành tố như câu hội thoại hay câu hỏi gần nhất mà người dùng đưa ra.

2.1.2.3. Thành phần sinh ngôn ngữ (NLG)

Natural language genernation (NLG) liên quan đến việc tự động sinh văn bản ngôn ngữ tự nhiên bằng máy tính. NLG được sử dụng trong một loạt các nhiệm vụ như: Dịch máy, chuyển giọng nói thành văn bản, tự động sửa văn bản hoặc tự động hoàn thành văn bản. Trong khuôn khổ đề tài, NLG được sử dụng để sinh câu trả lời cho người dùng dựa trên việc ánh xạ các hành động đã được xác định tại thành phần quản lý hội thoại thành ngôn ngữ tự nhiên. Có 4 phương pháp ánh xạ thường được sử dụng bao gồm: Template-Base, Plan-based, Class-based, RNN-base.

Hình 2.5: Ánh xạ từ thành phần quản lý hội thoại và sinh ra văn bản

Trong khuôn khổ các câu trả lời đã được xác định trước. Đề tài sẽ sử dụng Template-Base. Phương pháp này là dùng những câu trả lời đã được định nghĩa sẵn để trả lời cho người dùng. (xem ví dụ ở Hình 2.5 và Bảng 2.1)

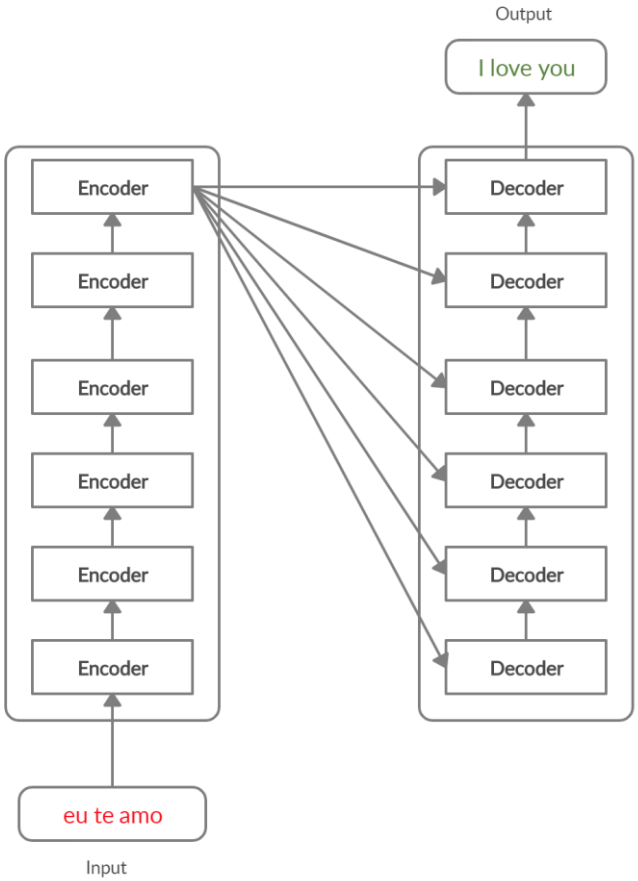
Bảng 2.1: Ánh xạ từ thành phần quản lý hội thoại và sinh ra văn bản

Sematic Frame	Natural Language
Gioi_thieu_khoa_hoc()	Bạn đang quan tâm đến khóa học nào ạ?
Gioi_thieu_khoa_hoc(khoa_hoc=$KH)	Khóa học $KH bạn quan tâm là chương trình ABC

2.1.3. Kiến trúc Transformer

Trước khi đi vào kiến trúc DIET mà Rasa đang sử dụng cần phải tìm hiểu nền tảng của kiến trúc này đó là kiến trúc Transformer.

Năm 2017, Google công bố bài báo "Attention Is All You Need" [8] đã tạo ra một bước ngoặt lớn trong lĩnh vực NLP. Transformer là một mô hình học sâu được thiết kế để phục vụ giải quyết nhiều bài toán trong xử lý ngôn ngữ tự nhiên, ví dụ như bài toán dịch tự động, bài toán sinh ngôn ngữ, phân loại, nhận dạng thực thể, nhận dạng giọng nói, chuyển văn bản thành giọng nói. Tuy nhiên, khác với RNNs, Transformer không xử lý các phần tử trong một chuỗi một cách tuần tự. Nếu dữ liệu đầu vào là một câu, Transformer không cần phải xử lý phần đầu câu trước rồi đi dần tới phần cuối câu. Thay vào đó tất cả các từ trong câu sẽ đi vào mô hình cùng một lúc. Do khả năng tính toán song song này, Transformer có thể tận dụng khả năng tính toán song song của GPU và giảm thời gian xử lý đáng kể.

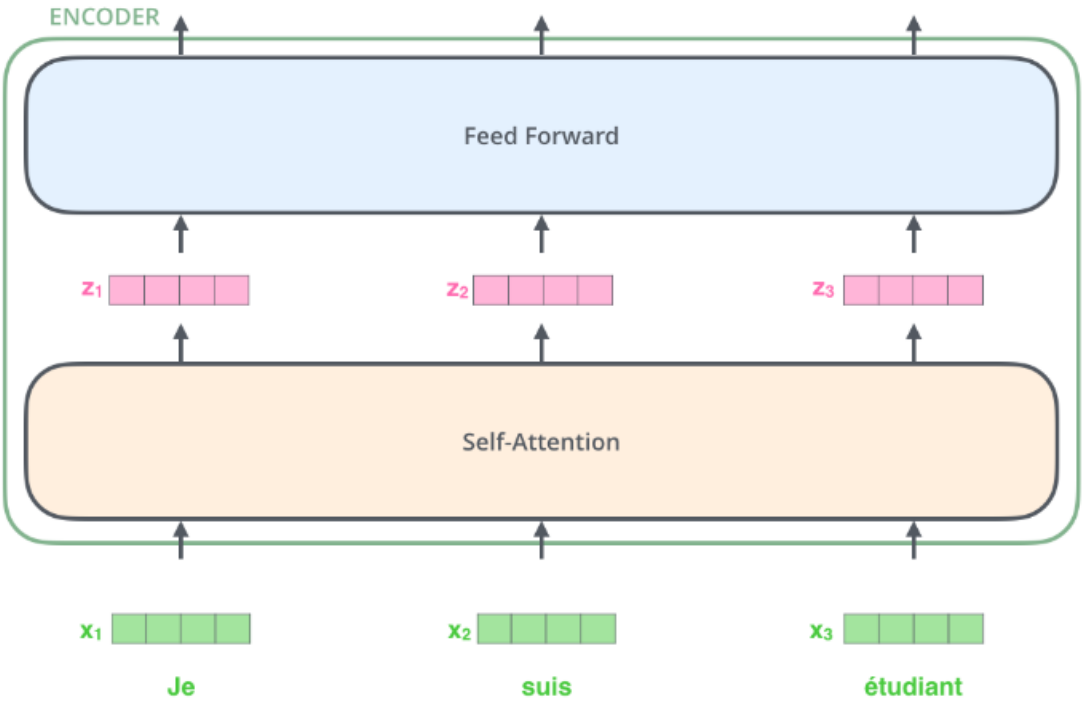


Hình 2.6: Kiến trúc tổng quát của Transformer

Khác với các kiến trúc như RNN, LSTM phải sử dụng hồi qui để nhớ các thông tin của các từ trước đó (vì các từ đi vào mô hình một cách tuần tự), Transformer sử dụng cơ chế self-attention. Trong kiến trúc Transformer chứa 6

encoder và 6 decoder (xem Hình 2.6). Ở đây đề tài chỉ tập trung khai thác vào thành phần encoder. Thành phần sẽ tạo ra các vector đặc trưng đại diện cho câu.

Thành phần encoder gồm 2 thành phần như được mô tả ở Hình 2.7 dưới đây:



Hình 2.7: Thành phần encoder

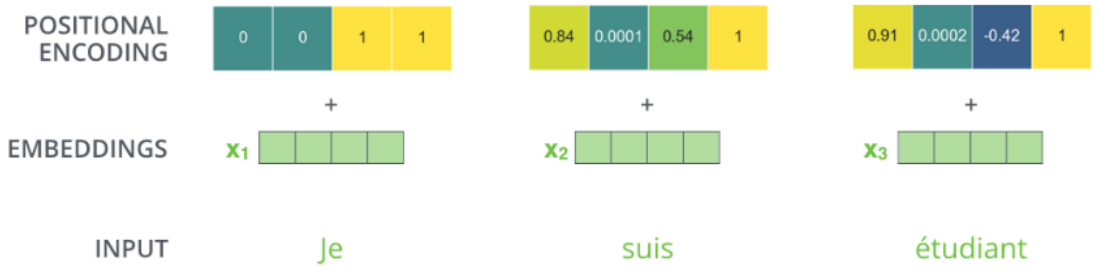
Đầu vào của encoder là một từ đã đi qua lớp nhúng (Word Embedding), lớp nhúng sẽ chuyển chuỗi đầu vào thành vector có số chiều d = 512.

Vì tất cả các từ trong câu đều đi vào mô hình trong cùng một lúc, nên vấn đề đặt ra là làm sao biết được từ nào đứng trước, từ nào đứng sau. Ví dụ câu "*Đi chơi không*" và câu "*Không đi chơi*" trên thực tế sẽ mang ý nghĩa khác nhau mặc dù số lượng từ và từ trong câu đều hoàn toàn giống nhau. Để giải quyết vấn đề thứ tự từ trong câu kiến trúc Transformer sử dụng phương pháp Position Encoding (PE) (xem Hình 2.8). Position Encoding là một vector có số chiều bằng với chiều của đầu ra của lớp Word Embedding (d = 512). Công thức tính của Position Encoding như sau:

$$\begin{cases} PE(pos, 2i) = \sin(pos/10000^{2i/d_{model}}) \\ PE(pos, 2i+1) = \cos(pos/10000^{2i/d_{model}}) \end{cases}$$

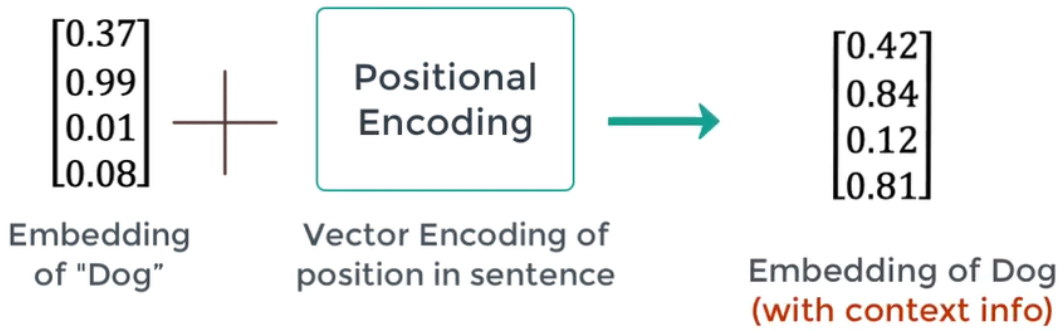
Với *pos* là vị trí của từ đó trong câu còn *i* là index của vector PE.

Ví dụ: Giá trị PE của x_2 tức từ thứ 2 trong câu (pos = 1). Giá trị 0.84 đầu tiên được tính bằng công thức: $PE_{1,0} = \sin(1/10000^{0/d_{model}}$ ở đây 2i = 0 vì i = 0 tức là phần tử đầu tiên của vector PE.



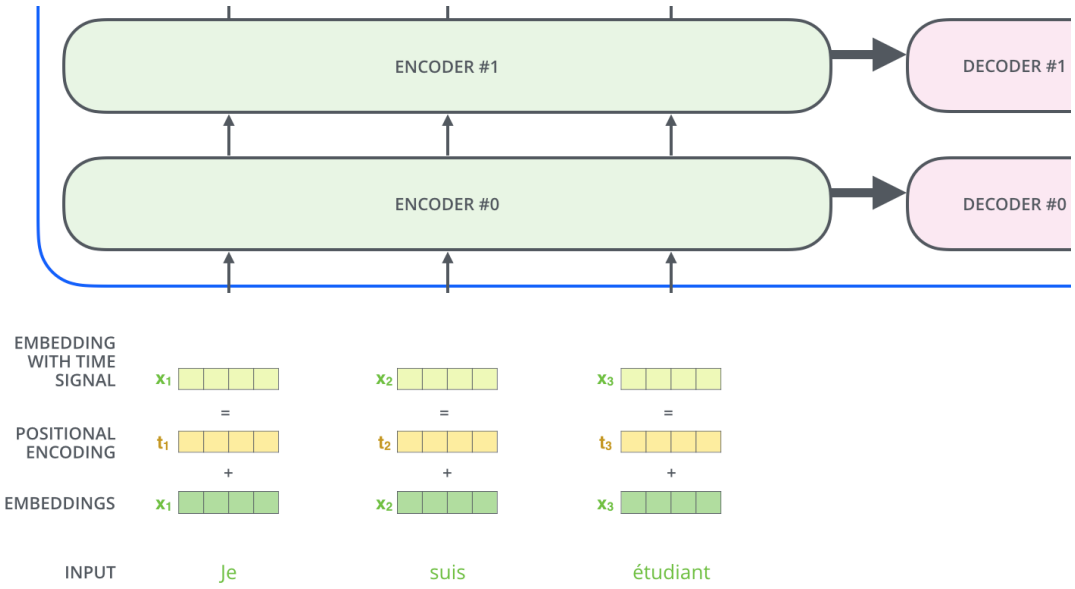
Hình 2.8: Kĩ thuật PE

Toàn bộ quá trình trên có thể trực quan hóa như Hình 2.9 dưới đây:



Hình 2.9: Trực quan hóa cách thức hoạt động của kĩ thuật PE

Từ đây có thể thấy sau khi cộng thêm vector PE thì các vector embedding đã có thêm thông tin về vị trí của từ so với các từ khác trong câu. Các vector đó sẽ được đưa vào thành phần encoder của kiến trúc Transformer (xem Hình 2.10).

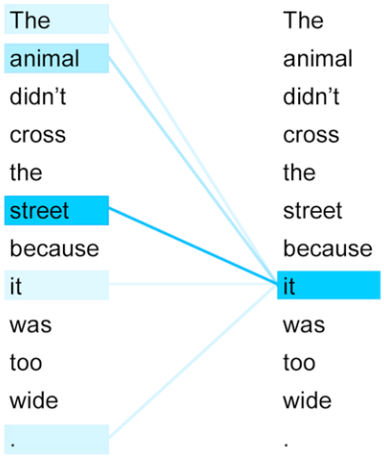


Hình 2.10: Thành phần encoder

Tiếp tục mổ xẻ thành phần encoder, một encoder gồm 2 thành phần bao gồm thành phần self-attention và một mạng Neural truyền thẳng (Feed Forward Neural Network - FFNN).

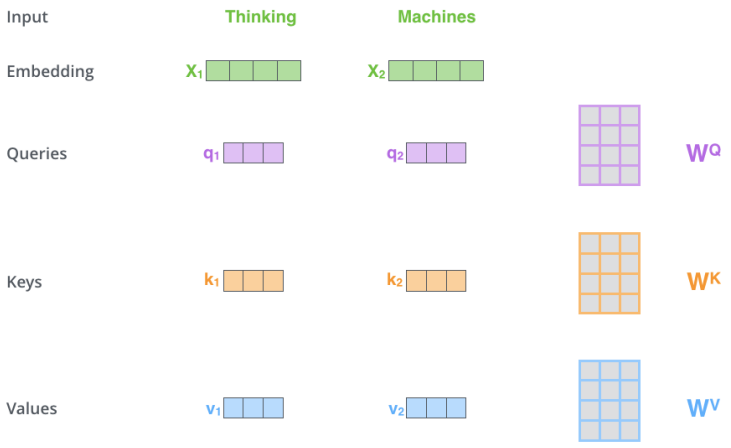
Self-attention giúp cho encoder nhìn vào các từ khác trong lúc mã hóa một từ cụ thể. Từ đó sẽ tạo được một vector mang thêm thông tin về quan hệ giữa từ đang được mã hóa và các từ khác trong câu.

Ví dụ trong câu "*The animal didn't cross the street because it was too tired*." Chữ "*it*" ở đây biểu diễn cho từ là "*animal*". Tuy nhiên nếu đối với một đứa trẻ mới học đọc thì có thể nó sẽ không biết "*it*" biểu diễn cho từ "*animal*" hay "street" hay bất cứ từ nào ở gần nó khác. Do đó cơ chế self-attention sẽ đưa ra mức độ chú ý của từ đó với từ khác như được mô tả ở Hình 2.11 dưới đây.



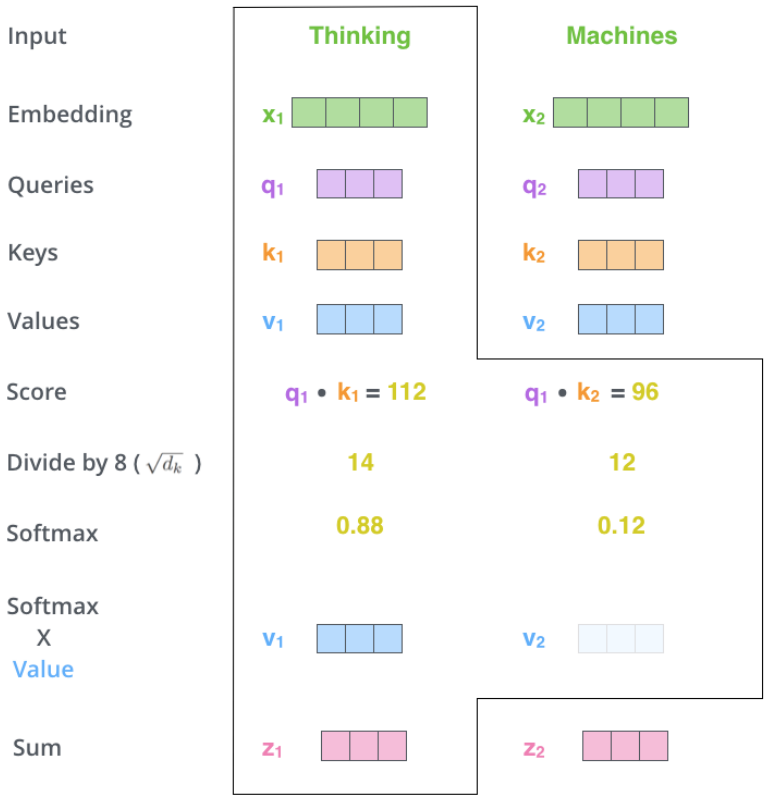
Hình 2.11: Cơ chế self-attention

Quá trình tính self-attention tạo ra 3 vector: Queries, Keys, Values. Bộ 3 vector này được tạo ra từ phép nhân ma trận giữa vector word embedding đầu vào với 3 ma trận trọng số W^Q, W^K, W^V (đây là 3 ma trận trọng số cần được tối ưu). (xem Hình 2.12)



Hình 2.12: Các bộ vector được xử lý bởi self-attention

Bước kế tiếp là tính điểm. Với mỗi từ, cần tính điểm của các từ khác trong câu đối với từ đó. Giá trị này giúp quyết định từ nào cần chú ý và chú ý bao nhiêu khi mã hóa một từ. Điểm được tính bằng tích vô hướng giữa vector Query của từ đang xét với lần lượt các vector Key của các từ trong câu. Các điểm đã tính được được chia cho 8 (căn bậc 2 của 64 – số chiều của vector Key) [8]. Điều này giúp cho độ dốc trở nên ổn hình hơn. Tiếp đến, các giá trị này lại được truyền qua hàm softmax để đảm bảo các giá trị đều dương và có tổng không vượt quá 1. Kế đến là nhân vector Value với mỗi giá trị điểm đã tính ở trên rồi cộng lại với nhau. Ý tưởng của việc này là bảo toàn giá trị vector của các từ cần được chú ý và loại bỏ vector của các từ không liên quan (bằng cách nhân nó với một số rất nhỏ - nếu một từ có độ chú ý thấp so với từ đang xét thì giá trị đầu ra của hàm softmax sẽ rất nhỏ). (xem Hình 2.13)

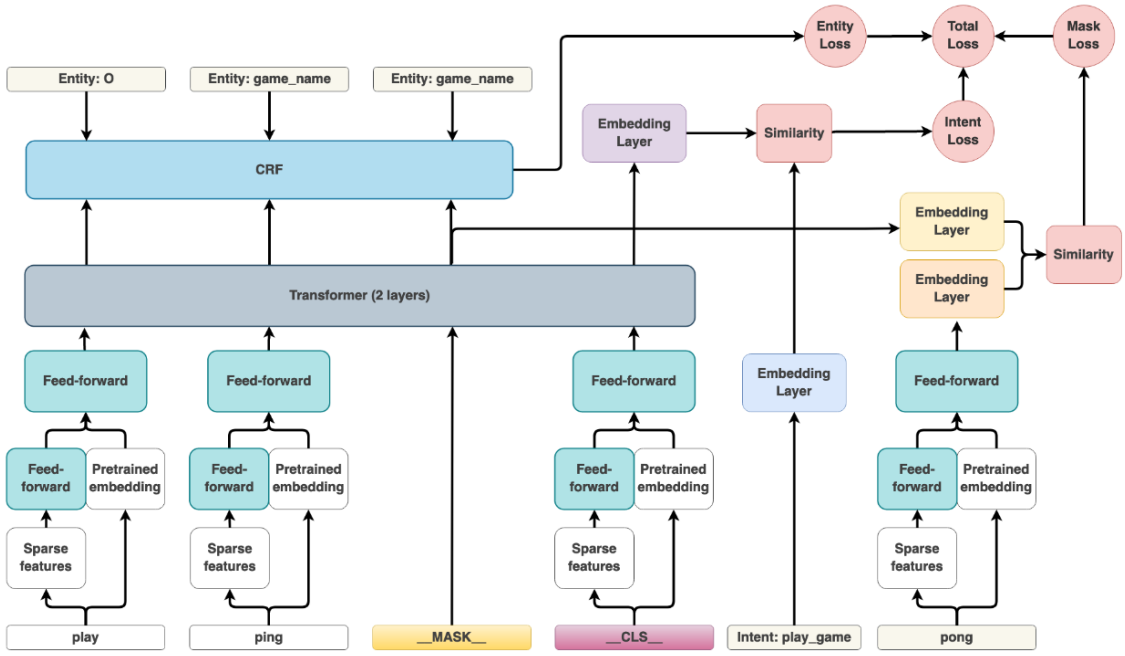


Hình 2.13: Quá trình tính điểm của một từ so các từ khác

Tuy nhiên trên thực tế, kiến trúc Transformer sử dụng đến 8 bộ Q, K, V. Kĩ thuật này được gọi là "Multihead Attention". Ý tưởng đằng sau kĩ thuật này là một từ có thể liên quan đến một từ khác theo nhiều khía cạnh khác nhau. Mỗi "head" sẽ cho ra một ma trận attention riêng, các ma trận sẽ được nối lại với nhau và nhân với một ma trận trọng số W^O (đây cũng là ma trận cần được tối ưu) để cho ra một ma trận attention duy nhất.

Đầu ra của lớp self-attention cuối cùng sẽ được đưa vào các lớp Feed Forward Neural Network và sinh ra ma trận đại diện cho toàn bộ câu.

2.1.4. Kiến trúc DIET



Hình 2.14: Kiến trúc DIET

DIET (Dual Intent and Entity Transformer) [9] là một kiến trúc đa tác vụ có thể thực hiện phân loại ý định và nhận dạng thực thể. Nó được cấu tạo từ nhiều thành phần (component) khác nhau cho phép nhà phát triển có thể linh hoạt trong việc thay đổi các thành phần để phù hợp với từng yêu cầu bài toán cụ thể. Ví dụ, nhà phát triển có thể sử dụng mô hình nhúng từ khác như PhoBERT [10], Fasttext [11] hoặc có thể loại bỏ hoàn toàn thành phần này khỏi kiến trúc. Kiến trúc này như một bộ các mảnh ghép Lego có thể dễ dàng "Plug and Play". Hình 2.14 bao gồm tất cả thành phần có trong kiến trúc DIET.

Hiện tại các mô hình ngôn ngữ được đào tạo trước với tập dữ liệu lớn (Large-scale pre-trained) đã cho thấy kết quả đầy hứa hẹn dựa trên các tiêu chuẩn đánh giá hiệu suất như Glue và SuperGLUE. Vì những mô hình này được đào tạo trên tập dữ liệu với qui mô lớn nên chúng có thể tổng quát hóa các nhiệm vụ trong NLP. Nhưng vấn đề đặt ra là những mô hình này rất nặng đòi hỏi máy tính với khả năng tính toán mạnh mẽ và tốn nhiều thời gian để huấn luyện. Đó là một thách thức đối với nhà phát triển chatbot khi mô hình chatbot cần huấn luyện lại liên tục để phù hợp với miền dữ liệu. Do đó mặc dù mang lại kết quả tốt nhưng chúng không phù hợp để áp dụng vào các ứng dụng AI hội thoại.

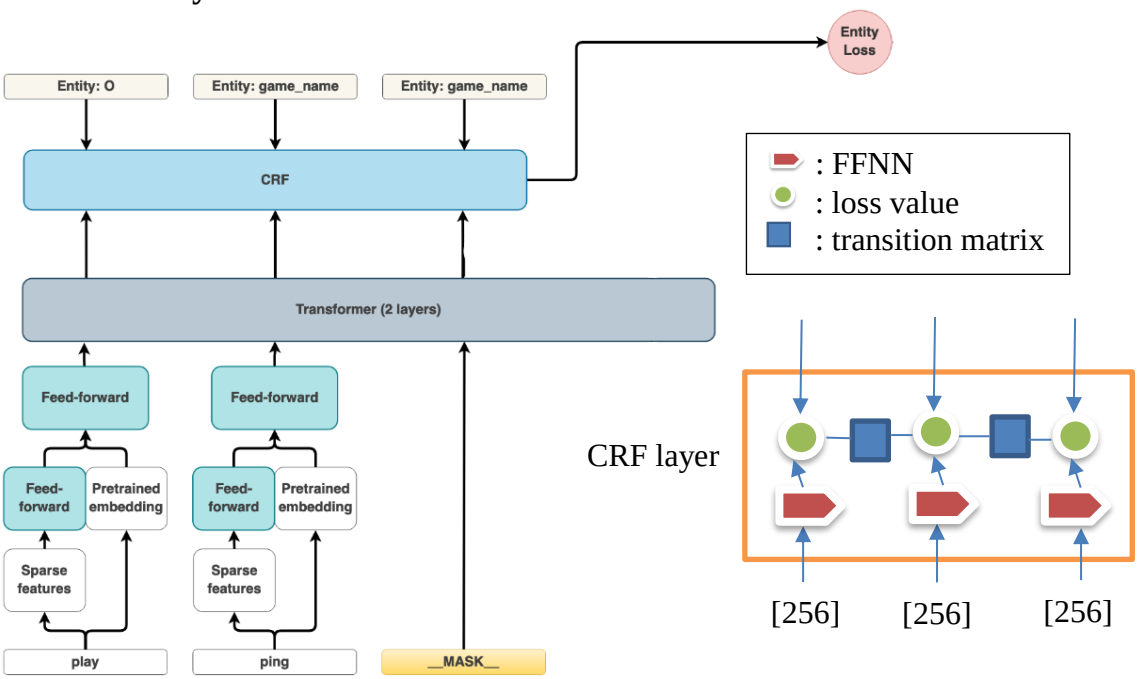
Vì những nhược điểm đó, vào năm 2019 Rasa đã cho ra mắt kiến trúc DIET và đã mang lại sự đột phá trong việc xây dựng chatbot vì những lí do sau:

- Kiến trúc này theo kiểu module giúp cho nhà phát triển có thể linh hoạt hơn trong các thử nghiệm và đánh giá (có thể dễ dàng thay đổi bất cứ module nào)
- Có độ chính xác và hiệu suất tương đương với các mô hình được đào tạo trước với qui mô lớn (Large-scale pre-trained)
- Vượt trội hơn so với mô hình SOTA hiện tại và tốc độ huấn luyện nhanh hơn gấp 6 lần

Vì là một kiến trúc thực hiện đa tác vụ nên kiến trúc DIET xử lý 3 bài toán tương ứng với 3 hàm mất mát cần tối ưu:

- Entity loss
- Intent loss
- Mask loss

2.1.4.1. Entity loss



Hình 2.15: Thành phần nhận dạng thực thể

Nhiệm vụ đầu tiên của Kiến trúc DIET là nhận dạng thực thể. Xem Hình 2.15 ở trên có thể thấy sau khi tách từ thì mỗi từ (token) trong câu sẽ đi qua một đường ống (pipeline) riêng biệt. Tại mỗi đường ống từ (token) sẽ đi qua 2 component (xem Hình 2.15):

- *Pretrained embeddings:* component này sử dụng các mô hình đã được đào tạo trước. Có thể là BERT, Spacy, Glove, …. Đầu ra của component này là một dense vector đại diện cho từ đầu vào.
- *Sparse features + Feed Forward Neural Network (FFNN):* mỗi từ (token) sẽ được trích xuất đặc trưng bằng phương pháp Lexical Syntactic. Sau đó được đưa vào một Mạng thần kinh truyền thẳng (Feed forward neural network – FFNN).

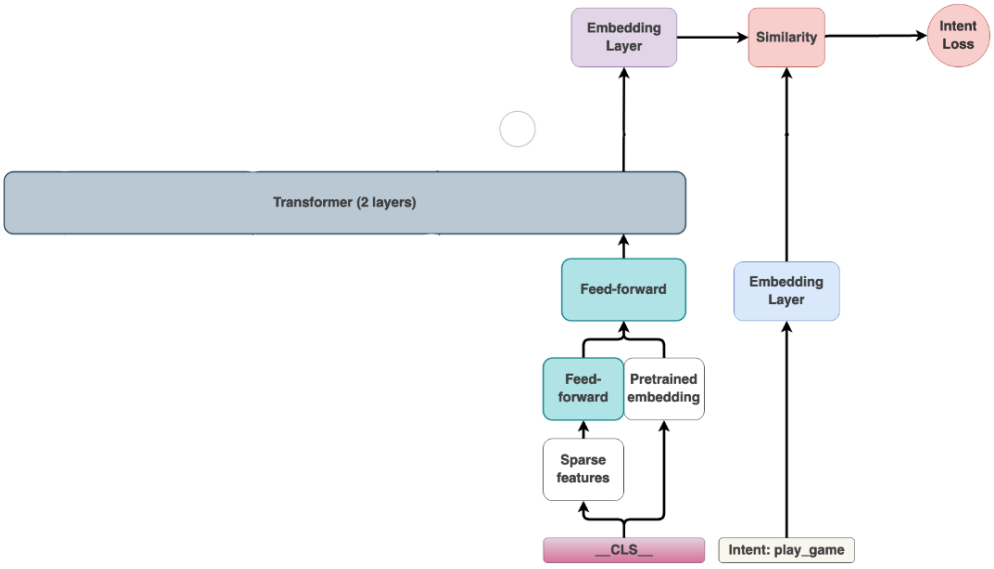
Đầu ra của 2 component này (*Pretrained embeddings* và *Sparse features + Feed Forward Neural Network*) sẽ được nối lại với nhau và lại đưa vào một mạng FFNN khác. Đầu ra của mạng FFNN này là một vector 256 chiều.

Các vector đó (tức là các vector 256 chiều đại diện cho từng token) được đưa vào một mạng Transformer 2 tầng. Sau đó đầu ra của tầng Transformer sẽ được đưa vào một tầng Conditional random field (CRF). Bên trong tầng CRF là một FFNN lấy đầu vào từ tầng Transformer và quyết định thực thể cần trích xuất là gì (xem Hình 2.15). Giữa các tầng FFNN này là một ma trận chuyển tiếp (transition matrix). Ma trận chuyển tiếp giải quyết vấn đề nếu một token là một thực thể thì các token lân cận cũng có thể là một thực thể cần trích xuất. Đối với mỗi token đều sẽ được gán nhãn cụ thể (có phải là một thực thể trong tập huấn luyện hay không) và được sử dụng trong quá trình huấn luyện mạng FFNN và ma trận chuyển tiếp.

Hàm loss được tính dựa trên 3 giá trị tức là 3 hình tròn màu xanh lá (đại diện cho nhãn thực tế của từ) như được mô tả ở Hình 2.15.

Một điểm đặc biệt trong kiến trúc DIET là tất cả các mạng FFNN đều là các kết nối không đầy đủ. Các mạng FFNN có một tỉ lệ dropout nằm trong khoảng 80%. Điều này khiến cho kiến trúc trở nên nhẹ nhàng, dễ dàng tính toán, giảm thời gian huấn luyện và tránh được vấn về Overfitting. Một điểm nữa là tất cả các FFNN này đều chia sẻ cùng một bộ trọng số (weight). Cụ thể tất cả các FFNN có đầu vào từ sparse feature (hình chữ nhật có góc bo tròn nhỏ) chia sẻ cùng bộ trọng số W_1, tất các các FFNN có đầu vào là sự kết hợp giữa Sparse features + *FFNN* và Pretrained Embedding (hình chữ nhật có góc bo tròn lớn) sẽ chia sẻ cùng bộ trọng số W_2. (xem Hình 2.15)

2.1.4.2. Intent loss



Hình 2.16: Thành phần phân lớp ý định

Trong kiến trúc DIET có một token đặc biệt được gọi là __CLS__. Ý tưởng của token đặt biệt này là nó sẽ tóm tắt toàn bộ câu và tạo một ma trận đại diện cho toàn bộ câu đầu vào. Token đặc biệt này cũng đi vào đường ống (pipeline) như các token thông thường khác. Tuy nhiên đầu ra tại pretrained embedding và sparse features có một số điểm khác biệt như sau:

- Đầu ra của *pretrained embedding* là một phép nhúng câu (sentence embedding)
- Đầu ra của *Sparse features* là tổng các *sparse features* của các token riêng lẻ trong câu. Câu sẽ được trích xuất đặc trưng bằng phương pháp n-gram và cũng được đưa vào một Mạng thần kinh truyền thẳng.

Ví dụ: với tập dữ liệu cho ý định "so_luong_tuyen_sinh" như sau:

```
data = [
'Mỗi năm CUSC nhận vào bao nhiêu sinh viên ạ?',
'Nhu cầu tuyển sinh mỗi năm của trung tâm là bao nhiêu
ạ?',
'Mỗi năm trung tâm nhận vào bao nhiêu sinh viên ạ?',
'chỉ tiêu tuyển sinh năm nay bao nhiêu vậy ad?',
]
```

Dựa vào các văn bản trên và với n = 1 và n = 2 ở cấp độ "từ" (có thể sử dụng cấp độ "kí tự") ta có danh sách các từ được sử dụng được gọi là *từ điển* như sau. Để ý rằng khi n = 1 là một trường hợp đặc biệt của n-gram được gọi là mô hình Túi đựng từ (Bag Of Words).

```
Với N = 1:
['ad', 'bao', 'chỉ', 'cusc', 'cầu', 'của', 'là', 'mỗi',
'nay', 'nhiêu', 'nhu', 'nhận', 'năm', 'sinh', 'tiêu',
'trung', 'tuyển', 'tâm', 'viên', 'vào', 'vậy']
Với N = 2:
['bao nhiêu', 'chỉ tiêu', 'cusc nhận', 'cầu tuyển', 'của
trung', 'là bao', 'mỗi năm', 'nay bao', 'nhiêu sinh',
'nhiêu vậy', 'nhu cầu', 'nhận vào', 'năm cusc', 'năm
của', 'năm nay', 'năm trung', 'sinh mỗi', 'sinh năm',
'sinh viên', 'tiêu tuyển', 'trung tâm', 'tuyển sinh',
'tâm là', 'tâm nhận', 'vào bao', 'vậy ad']
```

Với mỗi văn bản sẽ tạo được một vector đặc trưng bằng số chiều của từ điển, mỗi phần tử đại diện cho số từ tương ứng xuất hiện trong văn bản đó. Với 4 văn bản trên ta sẽ có 4 vector đặc trưng như sau:

```
Với N = 1:
[[0 1 0 1 0 0 0 1 0 1 0 1 1 1 0 0 0 0 1 1 0]
 [0 1 0 0 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 0 0]
 [0 1 0 0 0 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0]
 [1 1 1 0 0 0 0 0 1 1 0 0 1 1 1 0 1 0 0 0 1]]
Với N = 2:
[[1 0 1 0 0 0 1 0 1 0 0 1 1 0 0 0 0 0 1 0 0 0 0 0 1 0]
 [1 0 0 1 1 1 1 0 0 0 1 0 0 1 0 0 1 0 0 0 1 1 1 0 0 0]
 [1 0 0 0 0 0 1 0 1 0 0 1 0 0 0 1 0 0 1 0 1 0 0 1 1 0]
 [1 1 0 0 0 0 0 1 0 1 0 0 0 0 1 0 0 1 0 1 0 1 0 0 0 1]]
```

Bởi vì token đặc biệt này đại diện cho toàn bộ câu đầu vào nên sẽ được dùng để phân loại ý định. Đầu ra của tầng Transformer sẽ đi qua một tầng Embedding (Embedding layer). Song song đó nhãn ý định cũng đi qua một tầng Embedding. Phương pháp này lấy ý tưởng dựa trên kĩ thuật Starspace [12] được Facebook phát triển có khả năng so sánh các thực thể khác loại (văn bản và nhãn của chính nó) bằng cách nhúng chúng vào cùng một không gian vector sao cho sự so sánh giữa chúng là có ý nghĩa. Vì vậy đầu ra của hai tầng Embedding trên đều là một vector 20 chiều.

$$h_{CLS} = E(a_{CLS}),\ h_{intent} = E(y_{intent})\ với\ h\ \in\ \mathbb{R}^{20}$$

Sự tương đồng giữa 2 vector sẽ được tính toán dựa trên Độ tương tự cosine:

$$Cosine\ similarity = \frac{\vec{a}.\vec{b}}{||\vec{a}||\,||\vec{b}||}$$

Với $\vec{a}.\vec{b} = \sum_1^n a_i.b_i = a_1.b_1 + a_2.b_2 + \cdots + a_n.b_n$

Độ tương tự cosine có tính định hướng, tức là 2 vector cùng hướng (góc giữa 2 vector bằng 0) sẽ có độ tương tự cosine bằng 1. Mục tiêu của thuật toán là thay đổi trọng số tại hai tầng Embedding sao cho vector đại diện cho một văn

bản phải gần vector nhãn ý định của nó nhất có thể tương đương với sự so sánh giữa 2 thực thể khác nhau có ý nghĩa. [12, 13]

Ví dụ dưới đây mô tả ý tưởng của kĩ thuật Starspace. Ta có tập dữ liệu bao gồm các tài liệu được gán nhãn sau đây (xem Bảng 2.2 và Bảng 2.3):

Bảng 2.2: Tập tài liệu mô tả kĩ thuật StarSpace

	Tài liệu
D_1	This food was great!
D_2	I love veggie pizza
D_3	Cheap Italian Restaurant
D_4	Find me a vegetarian place
D_5	Give me a pizza

Và tập hợp các nhãn sau đây:

Bảng 2.3: Tập nhãn mô tả kĩ thuật StarSpace

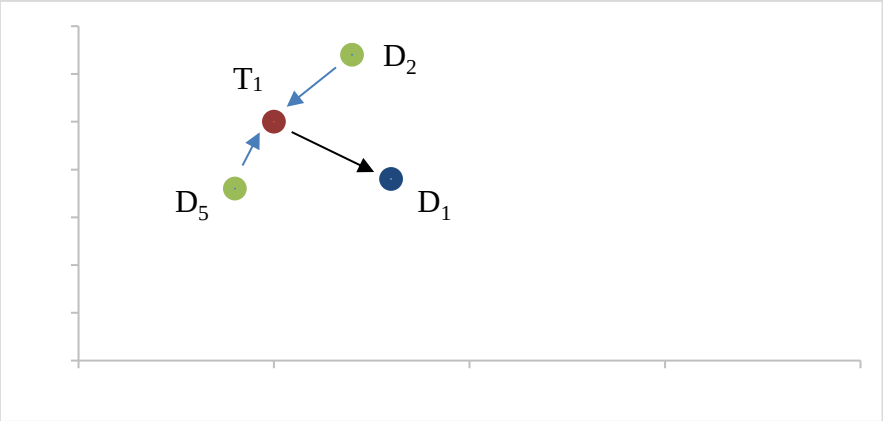
	Nhãn
T_1	#pizza
T_2	#positive
T_3	#italian
T_4	#vegetarian
T_5	#meat

Ta có thể có bảng quan hệ giữa tập tài liệu và các nhãn như sau:

Bảng 2.4: Quan hệ giữa tập tài liệu và tập nhãn

	T_1	T_2	T_3	T_4	T_5
D_1		X			
D_2	X	X		X	
D_3			X		
D_4				X	
D_5	X				X

Xem Bảng 2.4 có thể thấy cột T_1 có hai tài liệu được gán nhãn là D_2 và D_5. Ở một góc độ nào đó cả 2 đều mang nghĩa tương đương (đều mang ý nghĩa liên quan đến pizza) với nhãn T_1. Vì vậy nếu nhúng tài liệu và nhãn vào cùng một không gian vector thì chúng sẽ gần nhau.



Hình 2.17: Biểu đồ mô tả cách thức hoạt động của kĩ thuật StarSpace

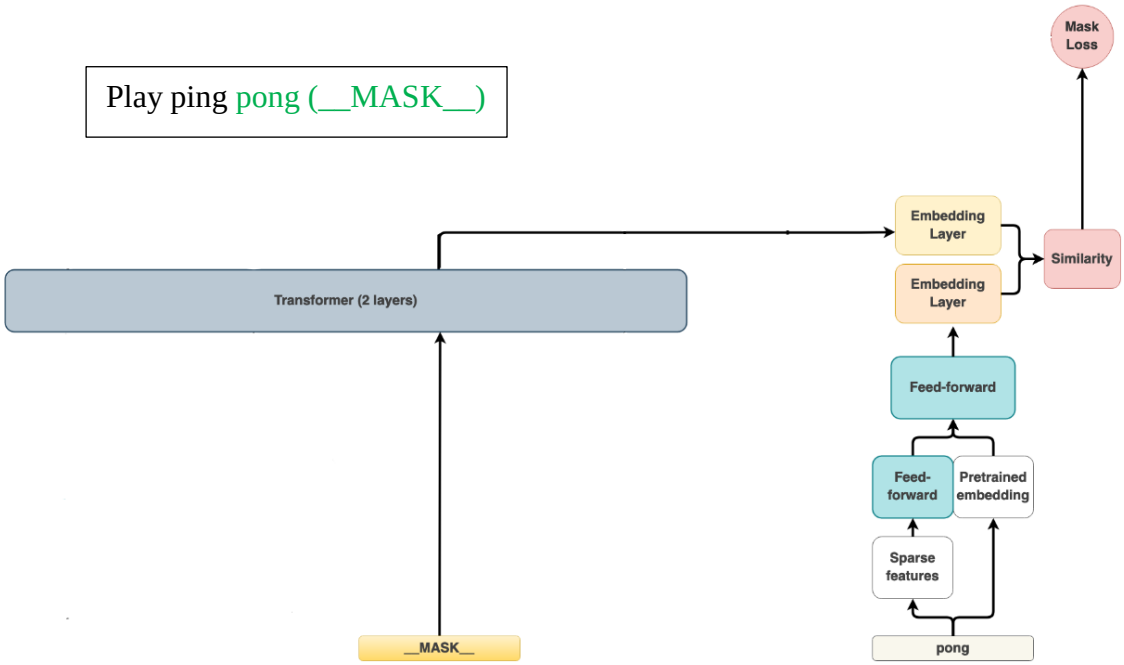
Mục tiêu của kĩ thuật StarSpace là làm sao cho hai tài liệu được gán T_1 là D_2 và D_5 hướng dần về T_1. D_1 không được gán nhãn T_1 sẽ hướng dần ra xa T_1. (xem Hình 2.17)

Như đã đề cập ở trên các nhãn ý định đều được qua lớp nhúng nên kiến trúc DIET có thể đào tạo một mô hình để so sánh sự tương đồng giữa một nhãn với các nhãn còn lại. Kĩ thuật này đã giúp cho kiến trúc DIET có thể xử lý các câu hỏi đa ý định. [12, 14]

2.1.4.3. Mask Loss

Kiến trúc DIET còn có thể được huấn luyện như một mô hình ngôn ngữ. Mô hình ngôn ngữ là một mô hình có thể dự đoán từ tiếp theo của câu dựa trên các từ đã được biết trước đó. Trong suốt quá trình huấn luyện, kiến trúc này sẽ che đi một số từ ngẫu nhiên. Mục tiêu của thuật toán là dự đoán đâu là từ gốc đã bị che đi.

Mục đích của việc sử dụng mask token và huấn luyện thêm một mô hình ngôn ngữ trong khi đã sử dụng pretrained model là giúp cho mô hình phù hợp với miền dữ liệu mà chatbot đang triển khai hơn. Đặc biệt trong giao tiếp qua tin nhắn, khi có thể xuất hiện từ sai chính tả hoặc tiếng lóng. Vì vậy việc huấn luyện thêm một mô hình ngôn ngữ sẽ cho phép mô hình nắm bắt được các trường hợp đặc biệt trong miền dữ liệu này.

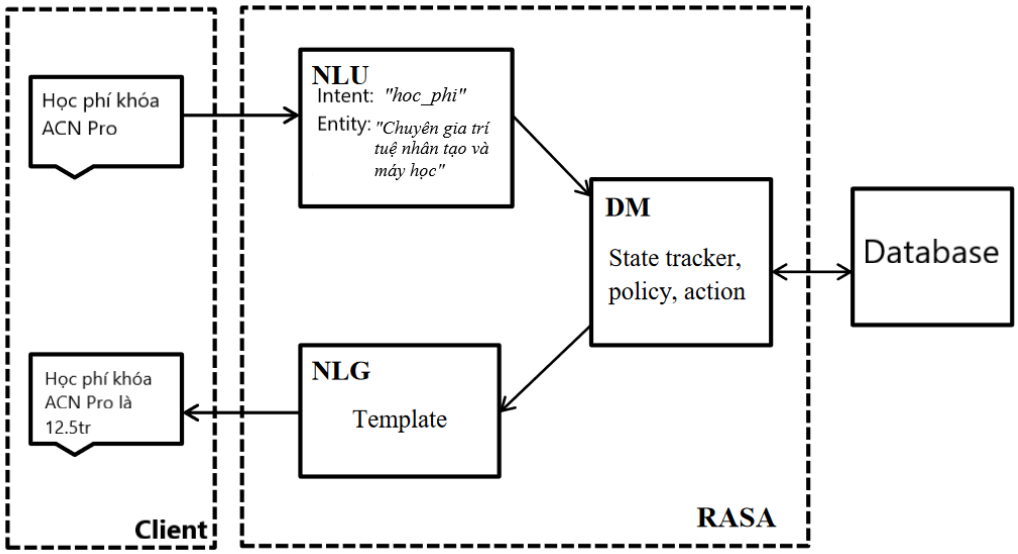


Hình 2.18: Thành phần huấn luyện mô hình ngôn ngữ

Xem Hình 2.18 có thể thấy mask token cũng sẽ đi qua transformer layer. Song song đó, từ được che đi, ở đây là từ "pong" cũng sẽ đi qua một đường ống (pipeline) tương tự như các token khác. Cả 2 sẽ đi qua một embedding layer. Độ tương đồng sẽ được tính giữa từ được mô hình dự đoán và từ gốc. [9]

2.2. Xây dựng hệ thống

2.2.1. Cấu trúc của hệ thống chatbot



Hình 2.19: Cấu trúc của toàn bộ hệ thống

Hình 2.19 mô tả quá trình xử lý của hệ thống chatbot với đầu vào là câu hỏi của người dùng và đầu ra là câu trả lời của bot. Các thành phần trong hệ thống chatbot sẽ đảm nhiệm các vai trò sau:

NLU: Có nhiệm vụ vector hóa ngôn ngữ, phân loại ý định, trích xuất thông tin thực thể từ tin nhắn của người dùng. Ví dụ khi người dùng hỏi "*Học phí khóa ACN Pro là bao nhiêu vậy ạ*" thì hệ thống sẽ vector hóa câu hỏi đó của người dùng. Sau đó đưa vào mô hình đã được huấn luyện trước đó và đưa ra ý định là "*hoc_phi*". Tiếp đến từ câu hỏi đó hệ thống sẽ trích xuất được thực thể *khoa_hoc* với thông tin là "*ACN Pro*". Tuy nhiên thực tế các khóa học có thể có nhiều tên gọi khác nhau. Ở đây khóa học "*ACN Pro*" còn có thể được gọi là khóa học "*Lập trình AI*" hay "*Trí tuệ nhân tạo*". Để đảm bảo hệ thống trích xuất tốt thông tin khóa học thông qua nhiều cách diễn đạt khác nhau từ người dùng. Hệ thống sẽ ánh xạ các tên gọi khác nhau đó thành một tên gọi chung. Khóa học "ACN Pro" sẽ được ánh xạ sang tên gọi chung là "*Chuyên gia trí tuệ nhân tạo và máy học*".

DM: Hệ thống sẽ dựa vào trạng thái và ngữ cảnh hội thoại để xác định hành động (action) nào sẽ được sử dụng để xử lý cho câu đầu vào trên. Ở đây có 2 loại action trong hệ thống:

- *Responses*: sẽ trả về cho người dùng câu trả lời dựa trên bộ câu trả lời đã được định sẵn
- *Custom action*: sẽ trả về cho người dùng câu trả lời thông qua việc gọi API hoặc truy xuất Database. Trong trường hợp như ví dụ trên sau khi hệ thống trích xuất được tên khóa học là "*ACN Pro*" thì hệ thống sẽ truy

xuất cơ sở dữ liệu học phí và trả về cho người dùng học phí của khóa học tương ứng

NLG: Dựa trên action đã được xác định từ thành phần DM hệ thống sẽ tham chiếu action đó với câu trả lời bằng ngôn ngữ tự nhiên đã được định sẵn.

Hiện nay có rất nhiều dự án mã nguồn mở cho phép lập trình viên có thể dễ dàng xây dựng chatbot như: Botpress, Botkit, BotMan, …. Nhưng nổi trội hơn hết là dự án mã nguồn mở Rasa với hơn 3.500 thành viên và số lượng tải về là 500.000. Các tính năng và lỗ hỏng liên tục được cập nhật và sửa đổi. Bên cạnh đó Rasa còn cho phép tùy chỉnh cơ chế học máy một cách linh hoạt giúp cho việc đánh giá và so sánh dễ dàng hơn bao giờ hết.

Về phần giao diện hiển thị , tương tác giữa người dùng và chatbot đề tài sử dụng ReactJS framework để xây dựng ứng dụng chat trên nền tảng web.

Về phần module website Quản lý học phí, đề tài sử dụng ngôn ngữ lập trình PHP và mô hình MVC để xây dựng.

2.2.2. Ứng dụng Rasa xây dựng chatbot

Hiện nay Rasa cung cấp 2 phương pháp xây dựng dữ liệu huấn luyện cho chatbot:

- **Pretrained Embeddings** (*Intent_classifier_sklearn*): Việc phân loại ý định người dùng sẽ dựa trên các tập dữ liệu được lọc trước, sau đó được sử dụng để thể hiện từng từ trong thông điệp người dùng dưới dạng từ nhúng (word embedding) hay biểu diễn ngôn ngữ dưới dạng vector (word2vec). Các tập dữ liệu này có thể được cung cấp từ Spacy hoặc MITIE …
- **Supervised Embeddings** (*Intent_classifier_tensorflow_embedding*): Nhúng được giám sát. Với phương pháp này thì người dùng sẽ phải tự xây dựng dữ liệu từ đầu do không có dữ liệu đào tạo sẵn có. Nhưng với các bài toán trong một lĩnh vực nhỏ thì nó sẽ đảm bảo tính chính xác hơn nhiều và tránh dư thừa dữ liệu so với phương pháp ở trên.

Tuy nhiên với sự ra đời của kiến trúc DIET nhà phát triển chatbot đã có thể kết hợp cả 2 phương pháp trên.

Cấu hình hiện tại mà chatbot đang sử dụng xử lý 3 nhiệm vụ bao gồm: phân loại ý định, trích xuất thực thể, xử lý câu hỏi đa ý định như sau:

```
language: vi
pipeline:
- name: WhitespaceTokenizer
intent_tokenization_flag: True
```

```
intent_split_symbol: "+"
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
  constrain_similarities: true
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: FallbackClassifier
  threshold: 0.6
  ambiguity_threshold: 0.1
```

2.2.3. Xây dựng dữ liệu chatbot

Nguồn dữ liệu xây dựng chatbot được cung cấp bởi Trung tâm Công nghệ phần mềm Đại học Cần Thơ và dữ liệu từ người dùng thực tế khi tương tác với chatbot.

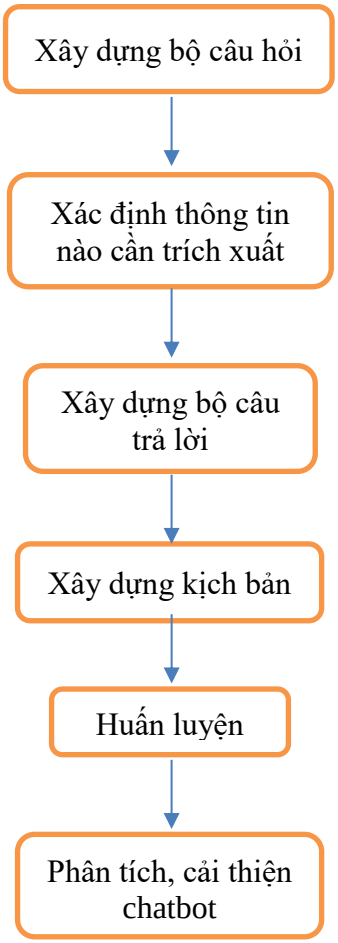
Vì là chatbot tư vấn khóa học, một miền dữ liệu đóng nên có thể phần nào đó xác định được khung kịch bản khi người dùng tương tác với chatbot. Ví dụ như khi người dùng hỏi về học phí thường sau đó sẽ hỏi tiếp về lịch học. Do đó việc xây dựng các khung kịch bản cho chatbot với miền dữ liệu đóng là một việc khả thi và cần thiết.

Ý định của người dùng có thể được diễn đạt theo nhiều cách khác nhau. Ví dụ với mục đích hỏi học phí người dùng có thể hỏi "*Học phí khóa ACN Pro là bao nhiêu*", "*Học phí khóa này bao nhiêu vậy ạ*", …. Vì vậy cần phải xây dựng một bộ câu hỏi đa dạng về mặt câu từ để chatbot có khả năng phân loại ý định tốt nhất có thể.

Bên cạnh đó như đã thấy ở ví dụ ở trên nếu người dùng hỏi "*Học phí khóa học này bao nhiêu vậy ạ*" thì làm sao chatbot biết được "*khóa học này*" là khóa học nào. Vì vậy thông qua việc trích xuất thực thể hệ thống có thể xác định được khóa học nào mà người dùng đang quan tâm thông qua các tin nhắn đã nhắn

trước đó và lưu trữ thông tin đó nhằm mục đích phục vụ cho cuộc hội thoại diễn ra tiếp sau đó. Nếu người dùng chưa đề cập đến cụ thể khóa học nào hệ thống sẽ đưa ra gợi ý các khóa học và từ đó hệ thống có thể trích xuất được thông tin về khóa học mà người dùng quan tâm.

Tóm lại việc xây dựng chatbot có thể được mô tả thông qua sơ đồ ở Hình 2.20 như sau:



Hình 2.20: Qui trình xây dựng chatbot

2.2.3.1. Xây dựng bộ câu hỏi

Việc xây dựng bộ câu hỏi tức là xác định câu hỏi nào thuộc ý định nào. Các câu hỏi được cung cấp bởi Trung tâm Công nghệ phần mềm Đại học Cần Thơ và thu thập từ người dùng được phân loại vào các ý định cụ thể. Hiện nay chatbot đang hỗ trợ khoảng 28 ý định. (xem Bảng 2.5)

Bảng 2.5: Bộ ý định hiện có của chatbot

Ý định	Diễn giải ý định	Số lượng câu
greet	Chào hỏi	5
bye	Tạm biệt	6
so_luong_tuyen_sinh	Số lượng tuyển sinh	7

dang_ky_xet_tuyen	Đăng kí xét tuyển	10
dieu_kien_tuyen_sinh_chuan_quoc_te	Điều kiện tuyển sinh chuẩn quốc tế	4
chuong_trinh_hoc	Chương trình học	11
doi_tuong_tuyen_sinh	Đối tượng tuyển sinh	8
kha_nang_toan_hoc	Khả năng toán học	6
kha_nang_ve	Khả năng vẽ	10
dang_ki_cao_dang_dai_hoc_cntt	Đăng kí học cao đẳng hoặc đại học CNTT	7
hinh_thuc_hoc	Hình thức học	5
dai_hoc_can_tho	Cơ sở vật chất tại Đại học Cần Thơ	4
may_tinh	Có cần máy tính riêng không	8
Phong_hoc	Chất lượng phòng học tại trung tâm	7
phong_trao	Phong trào tại trung tâm	7
bang_cap_quoc_te	Bằng cấp quốc tế	7
acn_pro	Bằng ACN Pro	3
gioi_thieu_khoa_hoc	Giới thiệu khóa học hiện có tại trung tâm	85
lien_thong_khoa_hoc	Liên thông đại học	17
co_hoi_viec_lam	Cơ hội việc làm	25
hoc_phi	Học phí	25
thoi_gian_khoa_hoc	Thời khóa biểu, lịch khai giảng	18
he_vua_hoc_vua_lam	Hệ vừa học vừa làm là gì	5
nha_tuyen_dung_doi_tac	Liên kết với nhà tuyển dụng	5
thoi_gian_khoa_hoc+hoc_phi	Đa ý định: thời gian khóa học và học phí	10
gioi_thieu_khoa_hoc+hoc_phi	Đa ý định: giới thiệu khóa học và học phí	2
gioi_thieu_khoa_hoc +hoc_phi+thoi_gian_khoa_hoc	Đa ý định: thới thiệu khóa học, học phí và thời gian khóa học	4
out_of_scope	Câu hỏi ngoài luồng	4

Ví dụ đối với ý định hỏi về các phong trào tại trung tâm có thể được hỏi theo nhiều cách khác nhau như:

```
- intent: phong_trao
examples: |
  - Ngoài chương trình học chính khóa, phía Trung tâm có hoạt
động gì cho sinh viên không?
  - Trung tâm có tổ chức phong trào thường xuyên không
  - Trung tâm có tổ chức phong trào cho sinh viên không ạ
```

```
  - Có nhiều hoạt động ngoại khóa cho sinh viên tham gia tìm
hiểu không ạ?
  - Ngoài các giờ học chính khóa có sinh hoạt ngoại khóa gì
không?
  - trung tâm có tổ chức hoạt động ngoại khóa không ạ
```

Bên cạnh đó, trong ngôn ngữ giao tiếp thường ngày một câu nói có thể chứa nhiều hơn một ý định. Hệ thống cần xác định từng ý định là gì và có câu trả lời phù hợp cho từng ý định đó. Ví dụ khi người dùng hỏi "*khóa học Lập trình viên quốc tế học phí bao nhiêu và thời gian học là khi nào?*" thì hệ thống cần xác định ý định đầu tiên là học phí, ý định thứ hai là thời gian học và dựa vào đó để có câu trả lời phù hợp nhất. Tuy nhiên, việc thêm quá nhiều câu đa ý định có thể làm tăng tính phức tạp cho hệ thống. Ví dụ sẽ có ít khi người dùng hỏi "*học phí khóa học Lập trình viên quốc tế là bao nhiêu và thông tin dịch bệnh Covid-19 sao rồi*" nếu thêm đa ý định này vào hệ thống sẽ trở nên dư thừa và khiến hệ thống trở nên phức tạp hơn – có thể xảy ra bùng nổ tổ hợp và các ý định bị chồng chéo lẫn nhau. Vì vậy tùy theo tương tác giữa chatbot và người dùng trong thực tế để xử lý các câu đa ý định một cách linh hoạt và đảm bảo cuộc hội thoại luôn đi theo đúng hướng và có ý nghĩa.

```
- intent: thoi_gian_khoa_hoc+hoc_phi
examples:
  - Thời gian bắt đầu của khóa học này là khi nào vậy ạ, học
phí thế nào ạ
  - Thời khóa biểu của khóa học là khi nào ạ? học phí thế nào
  - Học phí ngành này là bao nhiêu và lịch học như thế nào
vậy
  - khóa học Phát triển [ứng dụng Web với LARAVEL &
ANGULARJS](khoa_hoc) học trong bao lâu vậy ạk
  - Học phí ngành này là bao nhiêu và lịch học như thế nào
vậy
```

2.2.3.2. Xác định thông tin cần trích xuất

Thực thể là thông tin được trích xuất từ tin nhắn của người dùng và được lưu trữ vào slot (một bộ nhớ của chatbot). Dựa vào thông tin trích xuất được hệ thống sẽ quyết định câu trả lời phù hợp cho người dùng và cũng nhờ việc lưu trữ nên hệ thống tránh được việc hỏi lại thông tin từ phía người dùng. Đề tài trích xuất thông tin tên khóa học và dựa vào đó sẽ đưa ra các câu trả lời phù hợp nhất.

```
entities:
  - khoa_hoc
```

Bên cạnh đó, một số khóa học có thể có cách gọi khác nhau như khóa học "*Chuyên gia trí tuệ nhân tạo và máy học*" có thể được gọi là khóa học "*ACN Pro*". Vì vậy, cần xây dựng bộ các từ đồng nghĩa để giúp hệ thống dễ dàng trích xuất thông tin khóa học hơn. Hiện tại, hệ thống có 8 bộ từ đồng nghĩa tương ứng với 8 khóa học khác nhau. Mỗi bộ từ là những cách gọi khác nhau cho một khóa học cụ thể.

Ví dụ đối với khóa học "*Chuyên gia trí tuệ nhân tạo và máy học*" và "*An ninh thông tin*" có thể có các tên gọi khác nhau như sau:

```
- synonym: chuyên gia trí tuệ nhân tạo và máy học
example: |
  - Lập trình AI
  - Machine Learning
  - trí tuệ nhân tạo
  - acn pro
- synonym: an ninh thông tin
example: |
  - hacker mũ trắng
  - bảo mật thông tin
```

2.2.3.3. Xây dựng câu trả lời

Sau khi xác định được ý định cũng như trích xuất được thông tin từ người dùng. Ứng với mỗi ý định cũng phải xây dựng các mẫu câu (template) trả lời tương ứng. Ví dụ đối với ý định hỏi khả năng vẽ có cần thiết không sẽ trả lời với nội dung dưới đây.

```
utter_kha_nang_toan_hoc:
  - text: Tư duy toán học ở khá trở lên cũng là lợi thế khi
tham gia học, nhưng không bắt buộc. Điều kiện xét tuyển của
ngành này là tốt nghiệp THPT (Ưu tiên cho hồ sơ nộp sớm). Các
bạn có kiến thức về lập trình căn bản (tư duy lập trình), VD
như lập trình C là 1 lợi thế, nhưng không bắt buộc. Nếu chưa
biết qua lập trình căn bản, các bạn cần nhiệt huyết và nỗ lực
nhiều hơn, có thể tự học, đọc tài liệu thêm. Trung tâm có tạo
điều kiện cho các bạn học dự thính môn Lập trình căn bản nếu
nếu có lớp phù hợp.
```

Bên cạnh đó có thể xây dựng bộ câu trả lời cho chatbot thông qua action. Action là những hành động mà chatbot xử lý để đưa ra câu trả lời phù hợp cho người dùng. Câu trả lời có thể dựa trên các ý định, thực thể và dữ liệu lấy từ các nguồn bên ngoài. Hiện tại có ba loại action trong Rasa Core:

- **Default actions:** các hành động mặc định được tích hợp sẵn vào trình quản lý hội thoại (dialogue manager) ví dụ như lắng nghe người dùng, khởi động phiên hội thoại, …

- **Utter actions:** các tập câu trả lời cố định do nhà phát triển chatbot xây dựng như ví dụ ở trên.
- **Custom actions:** một số câu trả lời cần tính linh hoạt và cập nhật liên tục. Ví dụ như người dùng hỏi "*Kết quả xổ số hôm nay*" thì hệ thống cần lấy dữ liệu kết quả xổ số mới nhất từ bên ngoài hệ thống để trả lời cho người dùng.

Để tăng tính linh hoạt và khả năng mở rộng của hệ thống, thông tin của các khóa học được lưu vào một bảng gồm 23 hàng tương ứng với 23 khóa học khác nhau và 5 cột tương ứng với các thông tin khác nhau sẽ được linh hoạt sử dụng để xác định câu trả lời phù hợp nhất sẽ trả về cho người dùng.

Khi người dùng đặt câu hỏi, hệ thống sẽ xác định được ý định của người dùng và trích xuất được thực thể tên khóa học. Từ đó hệ thống sẽ xác định được câu trả lời phù hợp với ý định và thực thể mà người dùng quan tâm thông qua bảng thông tin đã đề cập ở trên. Ví dụ khi người dùng hỏi "*Khóa học Trí tuệ nhân tạo là gì vậy ạ*" thì ý định ở đây sẽ là "***gioi_thieu_khoa_hoc***" dựa vào ý định đó hệ thống sẽ gọi đến custom action tương ứng tức là custom action "***ActionGioiThieuKhoaHoc***" action này sẽ dựa vào thông tin tên khóa học đã trích xuất được trước đó là "*Trí tuệ nhân tạo*" để xác định utter giới thiệu khóa học tương ứng là "***utter_tri_tue_nhan_tao***" được lưu trữ trong bảng thông tin. Bảng 2.6 dưới đây mô tả tóm tắt bảng lưu trữ thông tin khóa học mà hệ thống đang sử dụng.

Bảng 2.6: Tóm tắt bảng lưu trữ thông tin khóa học

Khóa học	Utter giới thiệu	Utter liên thông	Utter việc làm	Loại
Lập trình viên quốc tế	utter_lap_trinh_vien_quoc_te	utter_lien_thong_lap_trinh_vien_quoc_te	utter_viec_lam_lap_trinh_vien	1
Mỹ thuật đa phương tiện	utter_my_thuat_da_phuong_tien	utter_lien_thong_lap_trinh_vien_quoc_te	utter_viec_lam_my_thuat_da_phuong_tien	1
Chuyên gia trí tuệ nhân tạo và máy học	utter_tri_tue_nhan_tao		utter_viec_lam_tri_tue_nhan_tao	1
An ninh thông tin	utter_an_toan_an_ninh_thong_tin		utter_viec_lam_lap_trinh_vien	0
Quản trị mạng doanh nghiệp	utter_quan_tri_mang_doanh_nghiep		utter_viec_lam_lap_trinh_vien	0
Cao đẳng công nghệ thông tin	utter_cao_dang_cntt		utter_viec_lam_lap_trinh_vien	1

Cao đẳng thiết kế đồ họa	utter_thiet_ke_do_hoa		utter_viec_lam_my_thuat_da_phuong_tien	1

Đối với một số câu hỏi hệ thống cần truy xuất cơ sở dữ liệu để lấy kết quả trả về cho người dùng. Ví dụ như các câu hỏi liên quan đến học phí, thời gian khóa học. Ở đây, cơ sở dữ liệu về thời gian khóa học có sẵn và đã được Trung tâm công nghệ phần mềm cung cấp. Nhưng dữ liệu về các học phí các khóa học thì chưa có nên cần phải xây dựng.

Các khóa học tại Trung tâm công nghệ phần mềm gồm 3 loại bao gồm: khóa học ngắn hạn, khóa học dài hạn và một số khóa học đặc biệt. Tùy theo loại khóa học mà hệ thống sẽ có câu trả lời khác nhau.

Các câu hỏi liên quan đến thời gian khóa học được xử lý qua các bước sau đây:

1. Xác định ý định và thực thể từ tin nhắn của người dùng
2. Lấy thời gian cách đây 6 tháng (có thể thay đổi) so với hiện tại của hệ thống
3. Truy xuất cơ sở dữ liệu thời gian khóa học với thực thể khóa học đã xác định ở Bước 1 cùng với điều kiện khóa học đó có khai giảng trong 6 tháng gần đây
4. Nếu dữ liệu có tồn tại thì trả kết quả về cho người dùng với thông tin ngày khai giảng, ngày bắt đầu đăng kí, thời gian kết thúc đăng kí, thời khóa biểu của khóa học đó
5. Nếu dữ liệu không tồn tại hệ thống sẽ gợi ý một số khóa học có thể người dùng sẽ quan tâm

Các câu hỏi liên quan đến học phí được xử lý qua các bước sau:

1. Xác định thực thể và ý định từ người dùng
2. Từ thực thể khóa học đã xác định được ở Bước 1 xác định loại khóa học được lưu trữ trong Bảng 2.6
3. Với thông tin loại khóa học xác định được ở Bước 2. Tùy theo loại khóa học mà có cách xử lý và câu trả lời khác nhau. Có 3 loại khóa học: khóa học ngắn hạn, khóa học dài hạn, khóa học đặc biệt
    a. Đối với khóa học ngắn hạn và dài hạn hệ thống sẽ truy xuất cơ sở dữ liệu và tùy theo loại khóa học sẽ có cách trả lời khác nhau được trả về cho người dùng
    b. Đối với những khóa học đặc biệt hệ thống sẽ truy xuất câu trả lời được định nghĩa sẵn trong Bảng 2.6 để trả về cho người dùng
4. Nếu không tìm được thông tin học phí của khóa học hệ thống sẽ gợi ý một số khóa học có thể người dùng sẽ quan tâm

Bên cạnh 2 custom action được kể ở trên hệ thống còn có 4 custom action khác bao gồm:

1. Action Giới thiệu khóa học: dựa trên thông tin khóa học trích xuất được sẽ trả về câu giới thiệu khóa học tương ứng
2. Action Liên thông khóa học: dựa trên thông tin khóa học trích xuất được hệ thống sẽ trả lời các câu hỏi liên quan đến vấn đề liên thông đại học của từng khóa học cụ thể
3. Action Cơ hội việc làm: dựa trên thông tin khóa học trích xuất được sẽ hệ thống sẽ trả lời các câu hỏi liên quan đến vấn đề cơ hội việc là của từng khóa học cụ thể
4. Action Đăng kí đại học, cao đẳng Công nghệ thông tin: dựa vào chương trình cao đẳng hay đại học mà người dùng quan tâm hệ thống sẽ trả về thông tin đăng kí cụ thể cho từng chương trình

2.2.3.4. Xây dựng kịch bản (story)

Sau khi đã xây dựng được bộ câu hỏi và câu trả lời. Bước tiếp theo là tiến hành xây dựng kịch bản cho chatbot bằng cách sắp xếp phù hợp giữa ý định và hành động tương ứng. Rasa cung cấp 2 phương thức để xây dựng kịch bản bao gồm:

- ***Rules***: phù hợp với các đoạn hội thoại nhỏ như trả lời câu hỏi thường gặp. Tuy nhiên nhược điểm của phương thức này là không có khả năng tổng quát hóa vì nó luôn đi theo một đường dẫn đã được xác định sẵn. Ví dụ dưới đây là một luật hướng dẫn bot cách trả lời câu hỏi liên quan đến đăng kí xét tuyển.

```
version: "2.0"
rules:
- rule: đăng kí xét tuyển
steps:
- intent: dang_ky_xet_tuyen
- action: utter_dang_ky_xet_tuyen
```

Đối với các câu hỏi đa ý định luật sẽ có định dạng như sau:

```
- rule: multi intent thời gian học và học phí
steps:
- intent: thoi_gian_khoa_hoc+hoc_phi
- action: custom_action_lich_hoc
- action: custom_action_hoc_phi
```

Ví dụ trên là cách mà bot sẽ trả lời khi người dùng hỏi câu hỏi có đa ý định về thời gian học và học phí.

- ***Stories***: được sử dụng đối với các cuộc đối thoại dài. Có khả năng tổng quát hóa thành các đường dẫn hội thoại không lường trước. Dưới đây là

một đoạn hội thoại được thu thập từ người dùng thực tế. Người dùng chào hỏi, yêu cầu thông tin về các chương trình học, học phí, thời gian khóa học và một số vấn đề liên quan đến khóa học mà người dùng quan tâm.

```
- story: Story from Conversation
steps:
- intent: greet
- action: utter_greet
- intent: chuong_trinh_hoc
- action: utter_chuong_trinh_hoc
- intent: hinh_thuc_hoc
- action: utter_hinh_thuc_hoc
- intent: gioi_thieu_khoa_hoc
entities:
- khoa_hoc: mỹ thuật đa phương tiện
- slot_was_set:
- khoa_hoc: mỹ thuật đa phương tiện
- action: custom_action_gioi_thieu_khoa_hoc
- intent: kha_nang_ve
- action: utter_kha_nang_ve
- intent: thoi_gian_khoa_hoc+hoc_phi
- action: custom_action_lich_hoc
- action: custom_action_hoc_phi
- intent: bye
- action: utter_bye
```

Việc xây dựng đoạn hội thoại như trên một phần được tôi thêm thủ công, phần còn lại là thông qua việc học tương tác (Interactive learning) với bot, đây là một phương pháp hiệu quả để khám phá được khả năng hiện tại của chatbot và sửa chữa gần như ngay lập tức bất kì lỗi nào mà nó gặp phải. Thời điểm hiện tại chatbot đã được đưa vào thử nghiệm với một số người dùng nhất định. Thông tin về các đoạn hội thoại được thu thập, đánh giá, sửa lỗi giúp cho chatbot ngày càng giao tiếp tự nhiên hơn như con người.

2.2.4. Xây dựng ứng dụng chat

2.2.4.1. Xây dựng server chatbot Rasa

Để xây dựng ứng dụng chat thời gian thực đề tài sử dụng giao thức websocket để tạo kết nối và giao tiếp 2 chiều giữa client và server thông qua TCP(Transmission Control Protocol). WebSocket cung cấp giao thức giao tiếp hai chiều mạnh mẽ, nó có độ trễ thấp và dễ xử lý lỗi, hiện đã được hỗ trợ trên nhiều trình duyệt (Firefox, Google Chrome, Safari, …). Giao thức chuẩn thông thường của Websocket là ws://, giao thức secure là wss://. Vì vậy websocket thường được sử dụng cho những trường hợp yêu cầu realtime như chat, hiển thị biểu đồ hay thông tin chứng khoán.

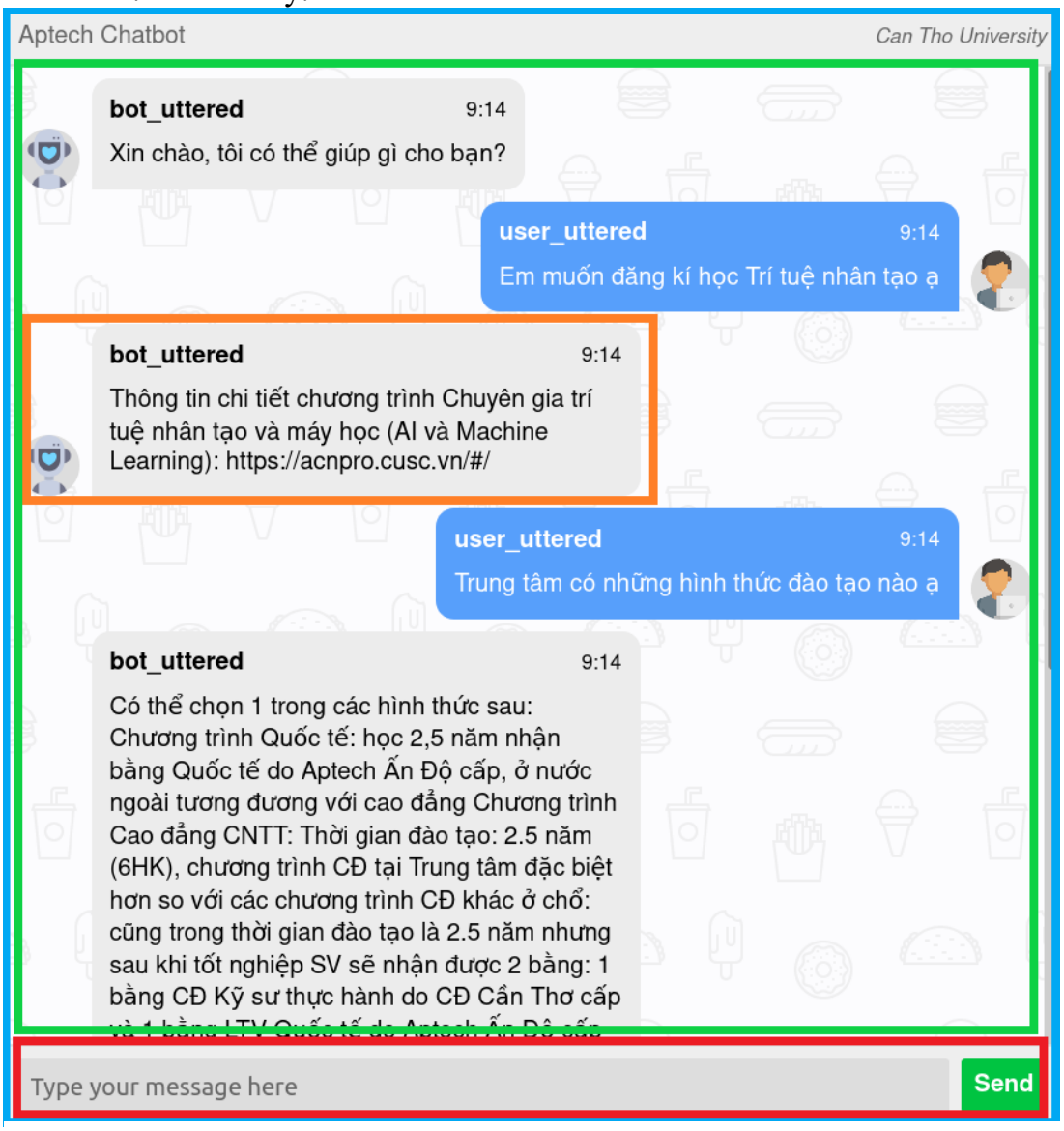
Rasa hỗ trợ sử dụng giao thức websocket bằng cách cấu hình file *credentials.yml* như sau:

```
socketio:
  user_message_evt: user_uttered
  bot_message_evt: bot_uttered
  session_persistence: true/false
```

Đầu tiên ở phía người dùng sẽ gửi yêu cầu kết nối đến server. Nếu đồng ý kết nối server sẽ trả về phía người dùng session_id được sử dụng để định danh người dùng trong suốt quá trình trao đổi dữ liệu. Sự kiện người dùng lắng nghe và gửi tin nhắn đến server được gán với "*user_uttered*", người dùng gửi tin nhắn đến server bao gồm nội dung tin nhắn và session_id đã được cung cấp trước đó, hệ thống sẽ xử lý tin nhắn của người dùng bao gồm: mã hóa, phân lớp ý định và trả về câu trả lời phù hợp nhất. Sự kiện bot lắng nghe và trả lời tin nhắn cho người dùng được gán với "*bot_uttered*".

2.2.4.2. Xây dựng giao diện chat

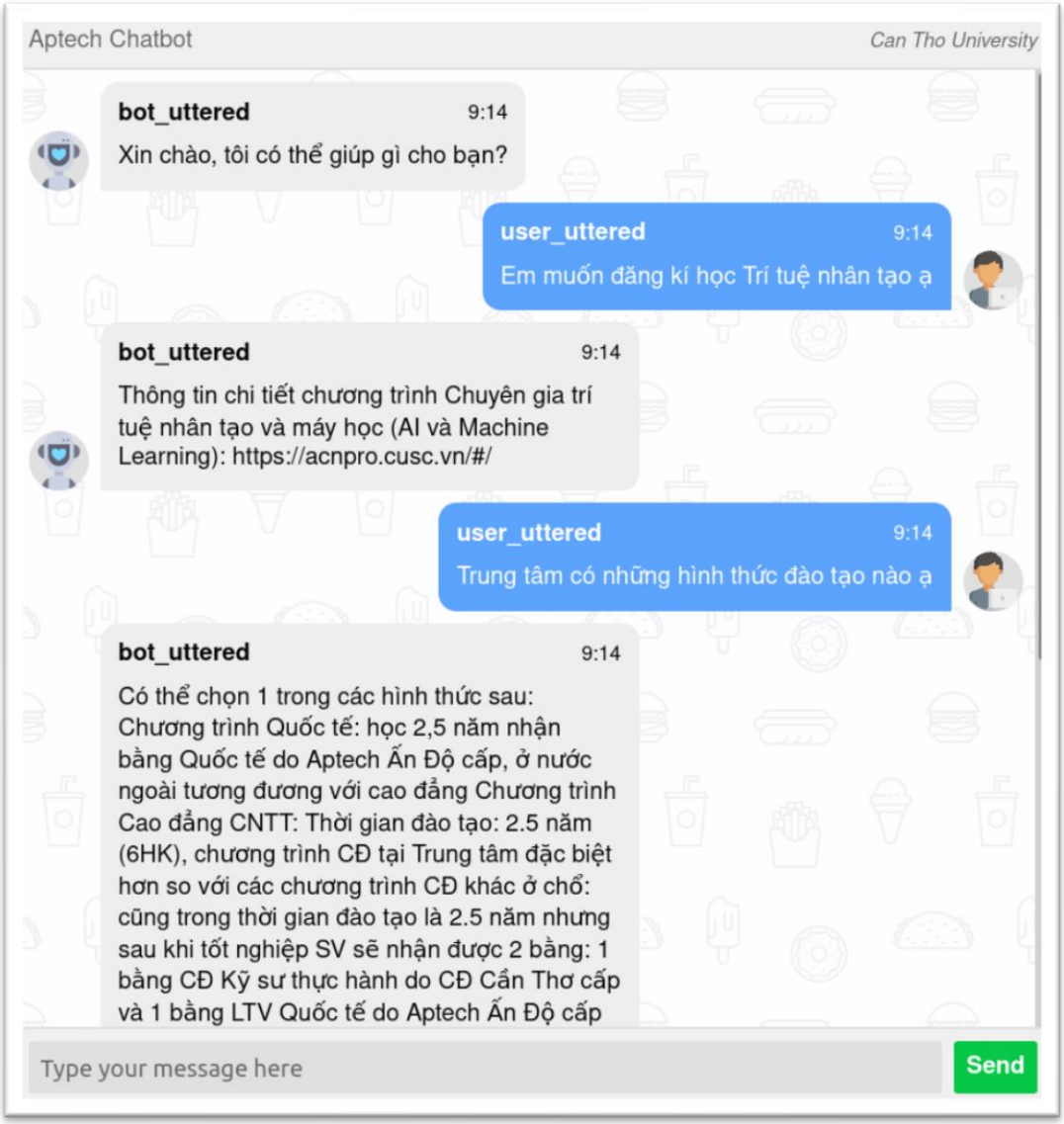
Đề tài sử dụng ReactJS framework để xây dựng giao diện chat giữa người dùng và chatbot. Giao diện chat được phân chia thành các component được mô tả ở Hình 2.21 dưới đây:



Hình 2.21: Các component của ứng dụng chat

- Khung màu xanh da trời tương ứng với component **App** nhận nhiệm vụ tạo, duy trì kết nối với server, trao đổi dữ liệu JSON về nội dụng chat giữa client và server. Bên cạnh đó component này còn nhận nhiệm vụ phân phối dữ liệu đến các component con.
- Khung màu xanh lá cây tương ứng với component **MessageList** nhận nhiệm vụ xử lý dữ liệu gửi đến từ component **App** và tiếp tục gửi dữ liệu đến component con **MessageItem**.

- Khung màu cam dương tương ứng với component **MessageItem** nơi nhận dữ liệu, xác định tin nhắn đó là của người dùng hay chatbot và hiển thị nội dung chat cho người dùng.
- Khung màu đỏ tương ứng với component **Input** nơi nhập nội dung câu hỏi từ người dùng và gửi về cho server.



Hình 2.22: Giao diện hệ thống chat

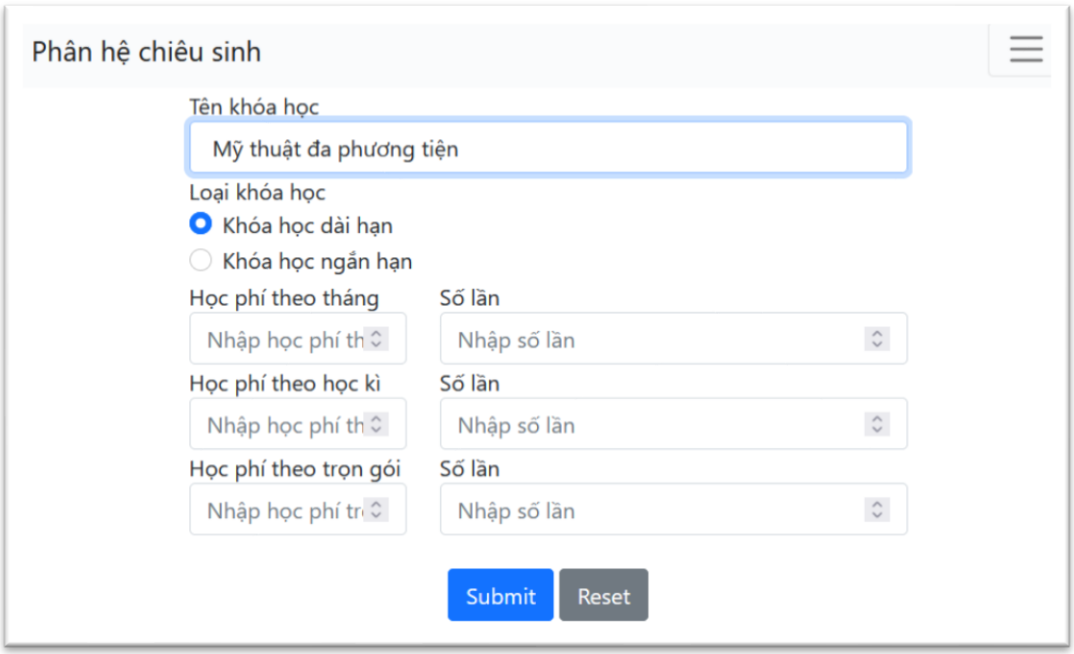
Hình 2.22 trên mô tả toàn bộ giao diện của hệ thống chat giữa người dùng và chatbot.

2.2.5. Module website học phí

Để hỗ trợ cho chatbot thuận tiện trong việc trả lời các câu hỏi liên quan đến học phí. Đề tài cũng xây dựng một module website Quản lý học phí là một phần của dự án "Website hỗ trợ xét tốt nghiệp". Module website Quản lý học phí

được xây dựng trên mô hình MVC cho phép quản trị viên có thể dễ dàng quản lý học phí của các khóa học hiện đang có tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ.

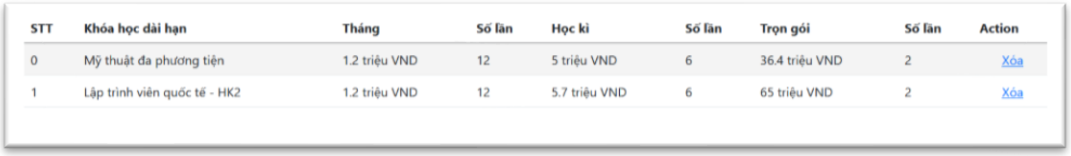
- Chức năng Nhập học phí: chức năng này cho phép quản trị viên lựa chọn tên khóa học, loại khóa học tương ứng (tùy loại khóa học sẽ có form nhập liệu khác nhau) và điền đầy đủ thông tin về học phí của khóa học. Sau đó nhấn nút "Submit" để lưu vào CSDL. (xem hình 2.23).



Hình 2.23: Giao diện nhập học phí

Cơ sở dữ liệu này sẽ được hệ thống chatbot truy xuất để trả lời các câu hỏi liên quan đến học phí các khóa học có tại trung tâm.

- Chức năng hiển thị học phí theo từng loại khóa học: hiện tại học phí tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ gồm 2 loại bao gồm học phí dài học và học phí ngắn hạn. Khóa học ngắn hạn chỉ cho phép đóng học phí 1 lần trong suốt quá trình học. Học phí dài hạn có thể chia ra nhiều lần đóng: đóng theo học kì, đóng theo tháng hoặc đóng trọn gói. Việc hiển thị học phí theo từng loại khóa học sẽ giúp cho quản trị viên có thể dễ dàng quản lý học phí hơn. Bên cạnh đó để xóa học phí của một khóa học quản trị viên chỉ cần nhấn "Xóa" để xóa học phí của khóa học đó khỏi CSDL. (xem Hình 2.24 và Hình 2.25)



STT	Khóa học dài hạn	Tháng	Số lần	Học kì	Số lần	Trọn gói	Số lần	Action
0	Mỹ thuật đa phương tiện	1.2 triệu VND	12	5 triệu VND	6	36.4 triệu VND	2	Xóa
1	Lập trình viên quốc tế - HK2	1.2 triệu VND	12	5.7 triệu VND	6	65 triệu VND	2	Xóa

Hình 2.24: Giao diện quản lý khóa học dài hạn

STT	Khóa học ngắn hạn	Số tiền	Action
0	PTƯD trên iOS	2.3 triệu VND	Xóa
1	Lập trình Java căn bản	2.4 triệu VND	Xóa
2	Thiết kế đồ hoạ cho quảng cáo	3.2 triệu VND	Xóa
3	Chuyên gia QTM cao cấp	4.5 triệu VND	Xóa
4	Phát triển ứng dụng Web với Laravel và AngularJS	2.7 triệu VND	Xóa
5	Thiết kế web chuyên nghiệp	3.6 triệu VND	Xóa
6	Lập trình Arduino	4.5 triệu VND	Xóa

Hình 2.25: Giao diện quản lý khóa học ngắn hạn

CHƯƠNG 3: KIỂM THỬ VÀ ĐÁNH GIÁ

3.1. Tiêu chí đánh giá

3.1.1. Confusion matrix

Phương pháp đánh giá kết quả đối với bài toán phân lớp, confusion matrix thể hiện có bao nhiêu điểm dữ liệu "thực sự" thuộc vào một lớp và được "dự đoán" rơi vào một lớp. Một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:

- ***True Positive (TP):*** những dự đoán mô hình dự đoán là nhãn "*positive*" và trong thực tế nhãn thật sự cũng là "*positive*"
- ***True Negative (TN):*** những dự đoán mô hình dự đoán là nhãn "*negative*" và trong thực tế nhãn thực sự cũng là "*negative*"
- ***False Positive (FP):*** những dự đoán mô hình dự đoán là nhãn "*positive*" nhưng trong thực tế nhãn này lại là "*negative*"
- ***False Negative (FN):*** những dự đoán mà mô hình dự đoán là nhãn "*negative*" nhưng trong thực tế nhãn này lại là "*positive*"

		Nhãn trong thực tế	
		Positive	Negative
Nhãn mô hình dự đoán	Positive	**TP_i (True Positive)**	**FP_i (False Positive)**
	Negative	**FN_i (False Negative)**	**TN_i (True Negative)**

Ví dụ: Mô hình phân loại bình luận dự đoán 1005 bình luận về một mặt hàng cụ thể xem bình luận mang tính chất tiêu cực hay tích cực. Dưới đây là những gì mô hình đã dự đoán:

- 90 bình luận tích cực và tất cả đều được dự đoán đúng bởi mô hình
- 915 bình luận tiêu cực nhưng trong thực tế có 910 trong số đó lại là bình luận tích cực

		Nhãn trong thực tế	
		Tích cực	Tiêu cực
Nhãn mô hình dự đoán	Tích cực	**90 (True Positive)**	**0 (False Positive)**
	Tiêu cực	**910 (False Negative)**	**5 (True Negative)**

3.1.2. Tiêu chí đánh giá Precision

Tiêu chí này trả lời cho câu hỏi "*Những dự đoán của mô hình chính xác bao nhiêu?*". Vì vậy chỉ số precision càng cao thì mô hình dự đoán càng chính xác. Dựa trên confusion matrix trên ta có công thức:

$$Precision = \frac{TP\ (True\ Positive)}{TP\ (True\ Positive) + FP(False\ Positive)}$$

Đối với ví dụ ở trên, chỉ số Precision trả lời cho câu hỏi "*Trong tất cả trường hợp được mô hình dự đoán là bình luận tích cực thì có bao nhiêu dự đoán là chính xác?*"

$$\frac{90}{90 + 0} = 100\%$$

Mô hình dự đoán 90 bình luận là tích cực và trên thực tế 90 bình luận này đều mang tính chất tích cực. Dựa vào kết quả trên có thể đi đến kết luận 100% số bình luận mô hình dự đoán tích cực là chính xác.

Tuy nhiên, Precision = 100% vẫn chưa đảm bảo mô hình là tốt, vì chỉ số này chưa trả lời được câu hỏi liệu mô hình đã tìm được tất cả điểm positive hay chưa. Trong trường hợp trên có thể thấy mô hình chỉ có thể tìm được 90 bình luận tích cực nhưng trong thực tế có đến 910 bình luận tích cực mà mô hình không thể dự đoán chính xác.

3.1.2. Tiêu chí đánh giá Recall

Tiêu chí này trả lời cho câu hỏi "*Mô hình có dự đoán thiếu kết quả nào không?*"

$$Recall = \frac{TP\ (True\ Positive)}{TP\ (True\ Positive) + FN(False\ Negative)}$$

Đối với ví dụ trên, chỉ số Recall trả lời cho câu hỏi "*Tất bình luận tích cực trong thực tế, mô hình dự đoán đúng bao nhiêu trong số đó?*"

$$\frac{90}{90 + 910} = 9\%$$

Mô hình chỉ dự đoán 90 bình luận là tích cực nhưng trong thực tế có đến 1000 bình luận tích cực. Từ kết quả trên suy ra mô hình chỉ có thể dự đoán 9% số lượng bình luận tích cực trong thực tế.

3.1.3. Tiêu chí đánh giá F1-Score

Để đánh giá độ tin cậy chung của mô hình, người ta kết hợp 2 chỉ số Precision và Recall thành một chỉ số đánh giá duy nhất F-Score.

$$F1\ Score = 2.\frac{Precision.Recall}{Precision + Recall}$$

Từ công thức trên có thể thấy một mô hình có chỉ số F1-Score cao khi cả 2 chỉ số Precision và Recall đều cao.

3.2. Đánh giá

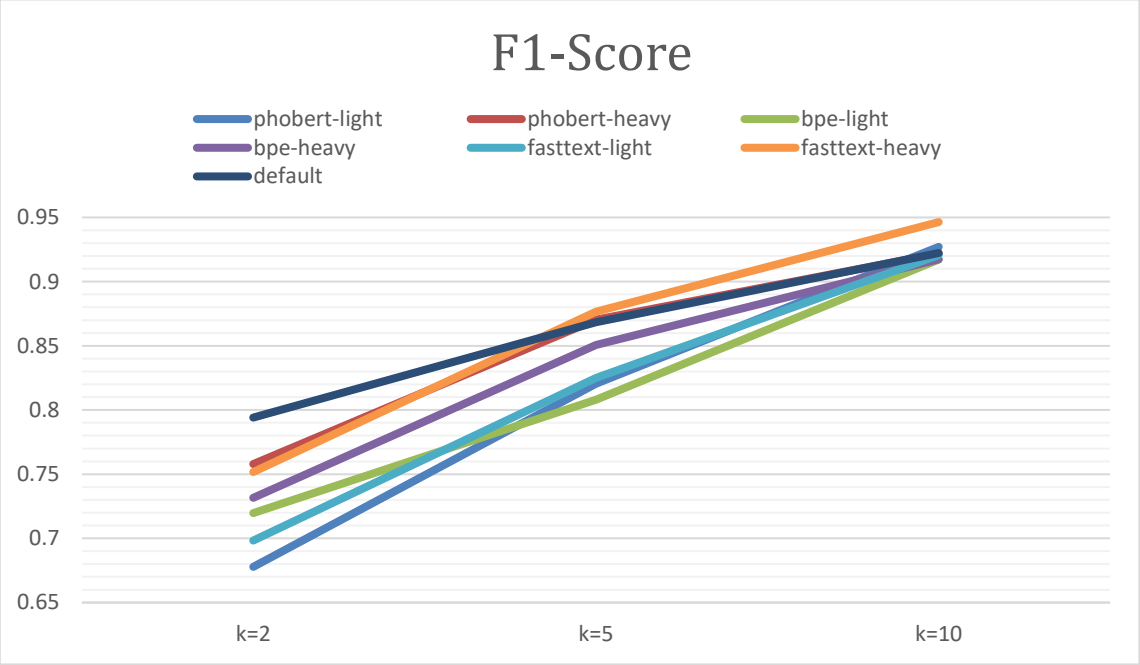
Cấu hình phần cứng hệ thống sử dụng để đánh giá chatbot (xem Bảng 3.1)

Bảng 3.1: Cấu hình hệ thống đánh giá chatbot

Thành phần	Thông số kĩ thuật
Vi xử lí CPU	Intel(R) Core(TM) i7-8850H CPU

Bộ nhớ RAM	16GB
Hệ điều hành	Ubuntu 20.04 LTS
Dung lượng ổ cứng	128GB SSD
Card đồ họa	NVIDIA Quadro P1000 4GB

Đề tài thử nghiệm 7 cấu hình chatbot khác nhau tương ứng với các kiến trúc DIET khác nhau [15]. Sử dụng nghi thức đánh giá k-fold với k lần lượt là 2, 5, 10. Mỗi mô hình được huấn luyện 100 epochs. Ta thu được biểu đồ như Hình 3.1 dưới đây:



Hình 3.1: Biểu đồ so sánh độ chính xác khi phân loại ý định của 7 cấu hình

Biểu đồ trên cho thấy có 2 cấu hình cho độ chính xác ổn định trong suốt quá trình đánh giá là cấu hình "*default*" và "*fasttext-heavy*". Cả 2 có cấu hình cụ thể như sau:

- **Default**

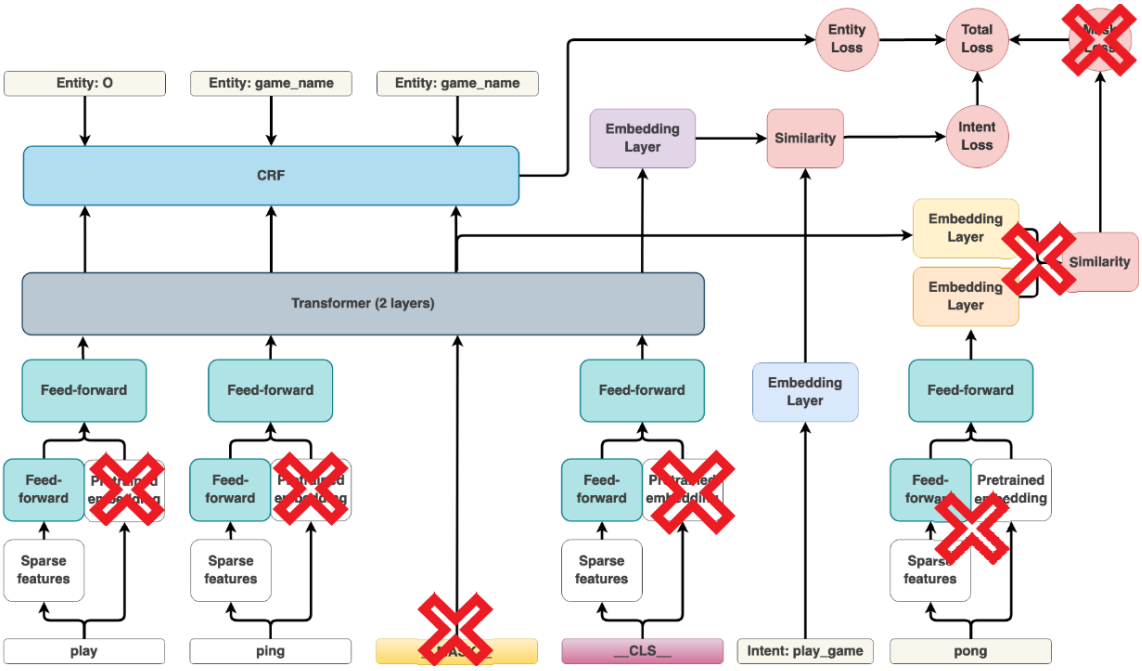
```
language: vi
pipeline:
- name: WhitespaceTokenizer
  intent_tokenization_flag: True
  intent_split_symbol: "+"
- name: RegexFeaturizer
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: DIETClassifier
  epochs: 100
  constrain_similarities: true
```

```
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: FallbackClassifier
  threshold: 0.3
  ambiguity_threshold: 0.1
```

Kiến trúc DIET của cấu hình "*default*" có những đặc điểm sau:

- Không sử dụng pre-trained model
- Không sử dụng mask token

Hình 3.2 dưới đây mô tả kiến trúc DIET của cấu hình trên:



Hình 3.2: Kiến trúc DIET cấu hình "default"

- **Fasttext-heavy**

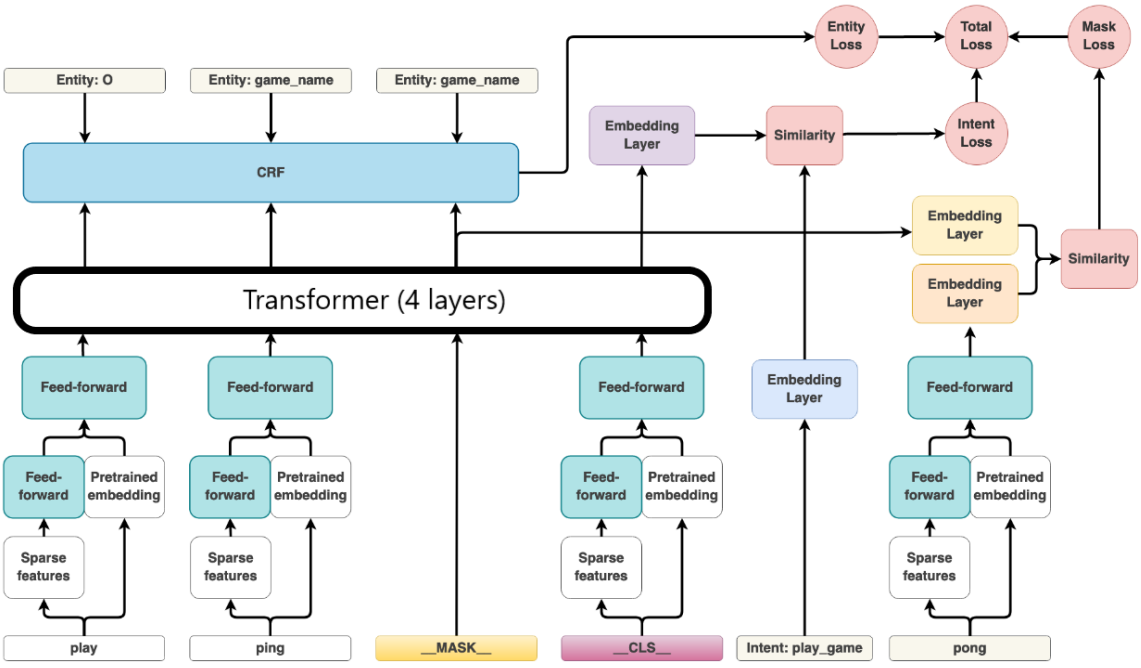
```
language: vi
pipeline:
- name: WhitespaceTokenizer
  intent_tokenization_flag: True
  intent_split_symbol: "+"
- name: LexicalSyntacticFeaturizer
- name: CountVectorsFeaturizer
  analyzer: char_wb
  min_ngram: 1
  max_ngram: 4
- name: fasttext_featurizer.FastTextFeaturizer
  cache_dir: pretrained_model
  file: cc.vi.300.bin
- name: DIETClassifier
  epochs: 100
  number_of_transformer_layers: 4
  transformer_size: 256
```

```
  use_masked_language_model: true
  drop_rate: 0.25
  constrain_similarities: true
  embedding_dimension: 30
- name: EntitySynonymMapper
- name: ResponseSelector
  epochs: 100
  constrain_similarities: true
- name: FallbackClassifier
  threshold: 0.3
  ambiguity_threshold: 0.1
```

Cấu hình "*Fasttext-heavy*" có những đặc điểm như sau:

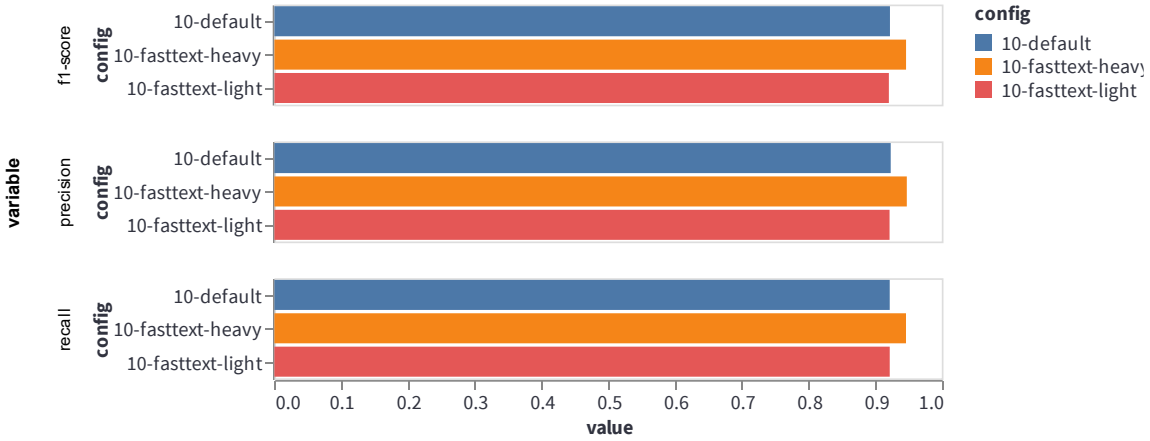
- Tích hợp pre-trained model Fasttext dành cho tiếng Việt vào kiến trúc DIET
- Sử dụng mask token
- Tăng tầng transformer từ 2 lên 4
- Tăng tỉ lệ dropout từ 0.2 lên 0.25

Kiến trúc của cấu hình trên được mô tả như Hình 3.3 dưới đây:



Hình 3.3: Kiến trúc DIET cấu hình "fasttext-heavy"

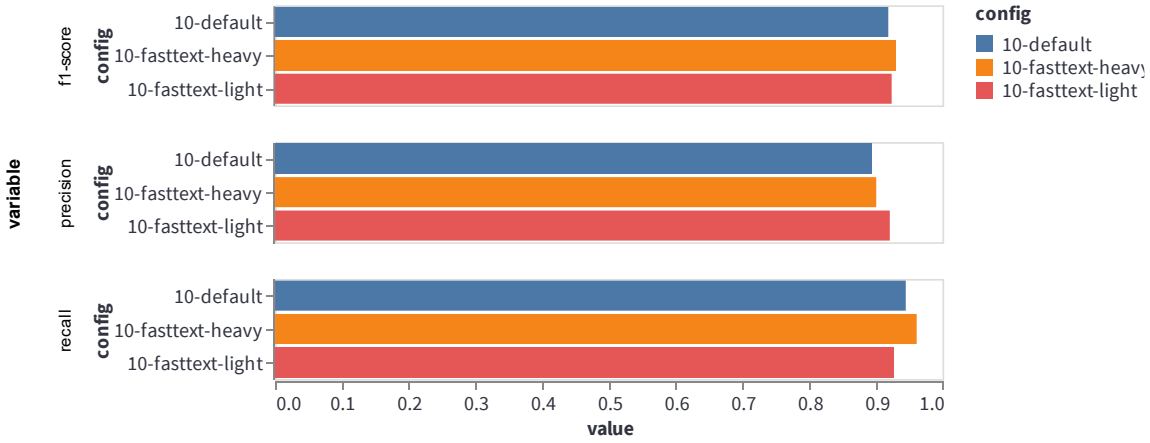
Biểu đồ dưới đây cho thấy kết quả đánh giá chi tiết của 2 cấu hình trên với k=10:



Hình 3.4: Biểu đồ so sánh phân loại ý định của 3 cấu hình

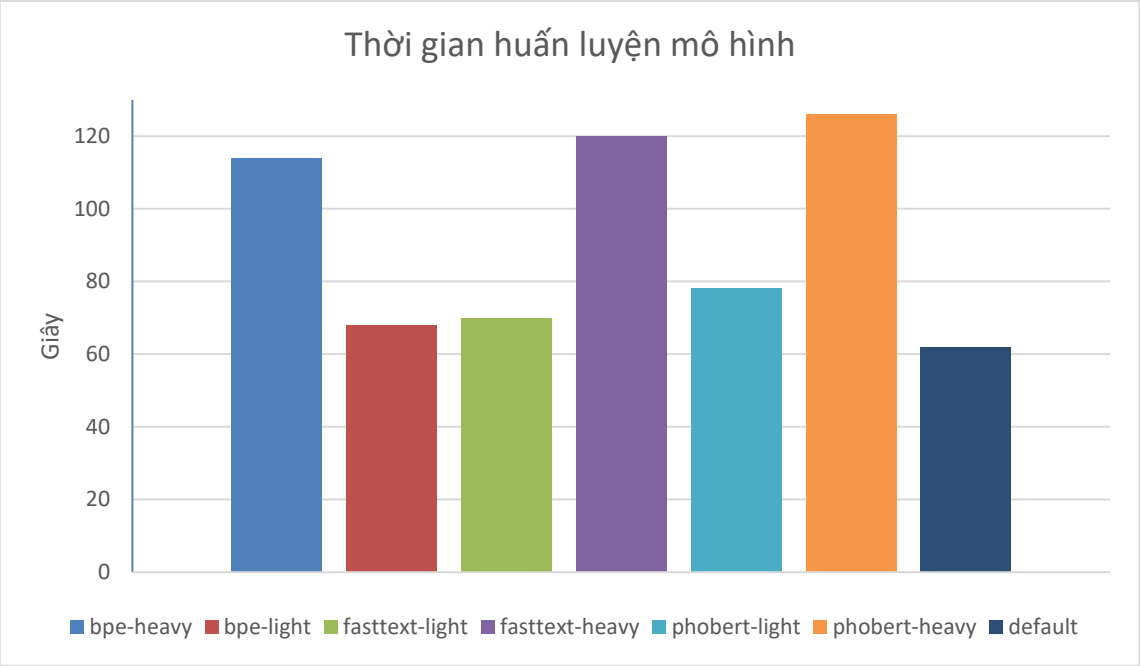
Có thể thấy độ chính xác khi phân loại ý định của cấu hình "*fasttext-heavy*" cao hơn so với cấu hình "*default*" khoảng 2.41%.

Bên cạnh đó kết quả trích xuất thực thể của cấu hình "*fasttext-heavy*" cũng cao hơn cấu hình "*default*" khoảng 0.5%. (xem Hình 3.5)



Hình 3.5: Biểu đồ so sánh nhận dạng thực thể của 3 cấu hình

Tất cả mô hình (mỗi mô hình huấn luyện 100 epochs) đều được huấn luyện trên GPU nên tốc độ huấn luyện cũng khá nhanh. Dưới đây là biểu đồ so sánh thời gian huấn luyện đối với từng cấu hình.



Hình 3.6: Biểu đồ so sánh thời gian huấn luyện mô hình

Như đã thấy ở biểu đồ trên (xem Hình 3.6) thời huấn luyện mô hình với cấu hình "*default*" nhanh hơn gấp 2 lần so với cấu hình "*fasttext-heavy*" và là cấu hình có thời gian huấn luyện nhanh nhất. Khi chỉ mất 1 phút để huấn luyện xong mô hình.

Từ đó có thể thấy, việc nên lựa chọn cấu hình "*default*" hay "*fasttext-heavy*" tùy thuộc vào từng trường hợp. Nếu chatbot được triển khai trên một hệ thống phần cứng ở mức tối thiểu. Nhà phát triển có thể đánh đổi độ chính xác không đáng kể để có thời gian huấn luyện mô hình nhanh hơn. Việc đánh đổi giữa độ chính xác và thời gian huấn luyện sẽ tùy vào từng trường hợp mà nhà phát triển ứng dụng có thể linh hoạt lựa chọn.

Tuy nhiên việc đánh giá chatbot như trên cũng chỉ mang tính khách quan. Vì vậy chatbot đã đưa vào thực nghiệm với một số người dùng thử nghiệm. Thử nghiệm tương tác với bot 20 câu hỏi ngẫu nhiên có liên quan đến bộ câu hỏi đã được đào tạo. Sau 5 lần kiểm thử thu được kết quả theo bảng đánh giá sau:

Bảng 3.2: Kết quả thực nghiệm

Lần thực nghiệm	**Số câu đúng**	**Độ chính xác**
1	10/20	50%
2	14/20	70%
3	14/20	70%
4	18/20	90%
5	19/20	95%

Trong 3 lần thực nghiệm đầu tiên có thể thấy hệ thống cho độ chính xác không được cao do hệ thống chưa có nhiều dữ liệu về các câu hỏi khác nhau cho

cùng một ý định. Bên cạnh đó chatbot cũng chưa trả lời được các câu hỏi đa ý định. Sau mỗi lần thực nghiệm tôi sẽ xem xét những câu mà chatbot chưa xác định ý định chính xác và gán nhãn thủ công các ý định đó. Bên cạnh đó tôi cũng nhận dạng những câu hỏi đa ý định mà người dùng hay hỏi và thêm vào hệ thống. Thử nghiệm cuối cùng khi hệ thống đã có bộ dữ liệu gồm 376 câu hỏi cho 28 ý định, mỗi ý định trung bình có khoảng 13 câu hỏi (bao gồm dữ liệu thu thập từ người dùng và dữ liệu có sẵn) hệ thống cho độ chính xác khá cao trên 95%, các câu hỏi đều liên quan đến bộ câu hỏi đã được đào tạo. Đối với những câu hỏi bot dự đoán với độ tin cậy thấp hơn 70% bot sẽ đưa ra gợi ý để người dùng diễn đạt lại ý định.

Dưới đây là 20 câu hỏi đã tương tác với chatbot ở thử nghiệm cuối (xem Bảng 3.3)

Bảng 3.3: Bộ câu hỏi sử dụng ở thử nghiệm cuối

STT	Câu hỏi	Câu trả lời
1	Em muốn biết chỉ tiêu tuyển sinh là bao nhiêu?	CUSC có nhu cầu tuyển dụng và đào tạo 300 nhân lực CNTT cho mỗi chuyên ngành để cung cấp cho doanh nghiệp phần mềm và nội dung số, ưu tiên hồ sơ nộp sớm cho đến khi đủ 300, vì cam kết đảm bảo việc làm nên chỉ có thể hỗ trợ số lượng này thôi, nên em cứ nộp hồ sơ đăng ký xét tuyển trước từ bây giờ để giữ chỗ cho chắc.
2	Em đăng kí xét tuyển vào trung tâm bằng cách nào ạ?	Để đăng ký xét tuyển, em điền đầy đủ thông tin vào phiếu này, ký tên, dán ảnh và đem về xác nhận ở trường THPT hoặc xác nhận tại địa phương và gửi về trường theo địa chỉ Trung tâm Công nghệ Phần mềm trường Đại học Cần Thơ, số 01 - đường Lý Tự Trọng, Q. Ninh Kiều, TP. Cần Thơ (hoặc nộp trực tiếp nếu nhà gần). Em có thể chụp hình phiếu đăng ký xét tuyển sau khi đã làm xong và gửi qua đây nhé. https://aptech.cusc.vn/Portals/1/Phieu-dang-ky-xet-tuyen-2021-Aptech.pdf https://arena.cusc.vn/Portals/1/Phieu-dang-ky-xet-tuyen-2021-Arena.pdf https://aptech.cusc.vn/Portals/1/Phieu-dang-ky-xet-tuyen-2021.pdf
3	trung tâm hiện tại có chương trình đào tạo gì vậy ạ	Trường Đại học Cần Thơ có tuyển sinh ngành Công nghệ Thông tin chuẩn Quốc tế với các chuyên ngành Lập trình viên, Mỹ thuật đa phương tiện, Trí tuệ nhân tạo và máy học. Một số điểm khác biệt: Thời gian học ngắn, Đảm bảo việc làm, Bằng cấp Quốc tế được công nhận toàn cầu, có thể học song song nhận thêm bằng ĐH/CĐ, Chương trình 2,5 năm chỉ tập trung chuyên ngành, tiếng Anh, kỹ năng, Thực hành nhiều, tích lũy kinh nghiệm thực tiễn trong quá trình học thông qua Đồ án cuối mỗi học kỳ, kiến tập, thực tập. Tăng cường ngoại ngữ qua môi trường học Quốc tế, hỗ trợ học phí chứng chỉ ngoại ngữ, Chỉ cần tốt nghiệp THPT, không thi tuyển, không

		xét điểm thi, học bạ, Chương trình cập nhật liên tục. Em tham khảo thêm tại đây nhé: https://aptech.cusc.vn/tuyen-sinh-APTECH.html; https://aptech.cusc.vn/tuyen-sinh-ARENA.html; https://acnpro.cusc.vn/#/.
4	Đối tượng tuyển sinh của trung tâm là những đối tượng nào ạ?	Đối tượng: Tốt nghiệp THPT Hình thức tuyển sinh: Xét tuyển Chỉ tiêu: CUSC có nhu cầu tuyển dụng và đào tạo 300 nhân lực CNTT cho mỗi chuyên ngành để cung cấp cho doanh nghiệp phần mềm và nội dung số (Ưu tiên cho hồ sơ nộp sớm) Ngay từ bây giờ em có thể nộp hồ sơ để giữ chổ trước, khi có điểm thi tốt nghiệp hoặc bằng tốt nghiệp tạm thời sẽ bổ sung sau, nộp hồ sơ trực tiếp hoặc nộp qua đường bưu điện tới địa chỉ: Trung tâm Công nghệ Phần mềm Đại học Cần Thơ, số 01, Lý Tự Trọng, Ninh Kiều, Cần Thơ) Hồ sơ gồm có: Phiếu đăng ký xét tuyển - Xét tuyển học bạ: Dành cho chương trình học song song tùy theo điều kiện của trường liên kết (Trung tâm liên kết đào tạo Đại học Cần Thơ, Cao đẳng Cần Thơ, Cao đẳng VHNT Cần Thơ)
5	Em muốn đăng kí học đại học công nghệ thông tin	Đối với chương trình song bằng 4.5 năm, ngoài phiếu đăng ký xét tuyển, PHẦN ĐẠI HỌC CẦN THƠ NGÀNH CNTT HỆ VLVH sẽ có xét thêm học bạ hoặc điểm thi tốt nghiệp THPT. Hồ sơ này chị sẽ hướng dẫn sau, khi em đăng ký nhập học. Điều kiện: Tổng 3 môn trong tổ hợp xét tuyển đủ 14 điểm. Có 4 tổ hợp: A00, A01, B00, D01.
6	Học ngành này thì em có cần giỏi toán giỏi vật lý không ạ?	Tư duy toán học ở khá trở lên cũng là lợi thế khi tham gia học, nhưng không bắt buộc. Điều kiện xét tuyển của ngành này là tốt nghiệp THPT (Ưu tiên cho hồ sơ nộp sớm). Các bạn có kiến thức về lập trình căn bản (tư duy lập trình), VD như lập trình C là 1 lợi thế, nhưng không bắt buộc. Nếu chưa biết qua lập trình căn bản, các bạn cần nhiệt huyết và nỗ lực nhiều hơn, có thể tự học, đọc tài liệu thêm. Trung tâm có tạo điều kiện cho các bạn học dự thính môn Lập trình căn bản nếu nếu có lớp phù hợp.
7	Em không biết vẽ, vậy có học MTĐPT được không?	Vẽ - chỉ là lợi thế chứ không phải là nhân tố quyết định bạn có học được hay không. MTĐPT không yêu cầu phải có năng khiếu đặc biệt. Tư duy thẩm mỹ và tính sáng tạo là yếu tố hàng đầu giúp bạn thành công.
8	Phòng gym đại học cần thơ em được dùng không ạ	Trung tâm Công nghệ Phần mềm trực thuộc trường trường Đại học Cần Thơ nên khi em là sinh viên của Trung tâm thì có thể sử dụng các dịch vụ của trường Đại học Cần Thơ như: ký túc xá, Trung tâm học liệu (thư viện hiện đại hàng đầu Đông Nam Á), phòng tập Gym…, ngoài ra em có thể tham gia các phong trào, hoạt động của trường ĐHCT tổ chức như hội thao, hội diễn văn nghệ, cắm trại,…
9	Em có cần laptop không ạ	Nếu có điều kiện các em trang bị máy tính riêng để thực hành thêm tại nhà sẽ thuận tiện hơn vì CUSC áp dụng

		phương pháp học blended learning (kết hợp giữa lý thuyết, đào tạo online, và các buổi thực hành). Tuy nhiên trường hợp các em chưa mua kịp máy tính vẫn có thể học được vì tại Trung tâm có riêng phòng lý thuyết, phòng thực hành được trang bị máy tính cấu hình mạnh kết nối Internet cáp quang tốc độ cao phục vụ học tập và thực hành ngoài giờ, toàn bộ các địa điểm trong trường được phủ sóng wifi giúp sinh viên liên tục cập nhật thông tin mọi lúc, mọi nơi. Bên cạnh đó đối với chuyên ngành Mỹ thuật Đa phương tiện Trung tâm còn trang bị phòng Arena Studio để các em có thể chụp ảnh, quay video clip,… các thiết bị như máy chụp ảnh, máy quay phim, bảng vẽ Wacom để phục vụ, hỗ trợ sinh viên trong suốt quá trình học.
10	Ngoài chương trình học chính khóa, phía Trung tâm có hoạt động gì cho sinh viên không?	Em có thể tham gia các phong trào, hoạt động của trường ĐHCT tổ chức như hội thao, hội diễn văn nghệ, cắm trại,… Bên cạnh đó Trung tâm vẫn có các hoạt động riêng như các cuộc thi học thuật, văn nghệ, dã ngoại, tham quan kiến tập…
11	Bằng ACN Pro có giá trị quốc tế không	ACN Pro là chương trình do Aptech - Ấn Độ kết hợp trường Đại học Cần Thơ triển khai tại Việt Nam. Bằng cấp do Aptech - Ấn Độ cấp, có giá trị Quốc tế, được công nhận rộng rãi trên 52 quốc gia.
12	Em vừa tốt nghiệp THPT, em muốn tìm hiểu và dự định xét tuyển nguyện vọng vào ngành Công nghệ thông tin. Admin có thể cho một số tiêu chuẩn của mình không ạ?	Thời gian đào tạo: 4,5 năm với chuyên ngành ĐH CNTT, chương trình ĐH tại Trung tâm đặc biệt hơn so với các chương trình ĐH khác ở chỗ: cũng trong thời gian đào tạo là 4,5 năm nhưng sau khi tốt nghiệp SV sẽ nhận được 2 bằng: 1 bằng ĐH CNTT hệ VLVH do ĐHCT cấp và 1 bằng LTV Quốc tế do Aptech Ấn Độ cấp. Mời em tham khảo thêm thông tin chi tiết tại đây: https://aptech.cusc.vn/tuyen-sinh-APTECH.html
13	Sau khi học xong khóa lập trình viên quốc tế em có thể liên thông lên Đại học không	Sau khi tốt nghiệp chương trình Lập trình viên Quốc tế, sinh viên có thể liên thông ở các trường ĐH Quốc tế như: RMIT (Úc) học tại Việt Nam hoặc Úc; Swinburne, Southern Cross (Úc); Greenwich (Anh) học tại TPHCM/HN; UCLan (Anh), Oxford, Middlesex (Anh) học tại London - Anh hoặc Singapore, Lincoln (Malaysia)… hoặc các trường ĐH Việt Nam như ĐH Sư phạm Kỹ thuật Vĩnh Long… Tiết kiệm 2/3 thời gian và học phí.

14	Học ngành này xong em có cơ hội đi làm ở đâu	Theo Vietnamworks (8/2019), CNTT nằm trong top 5 các ngành nghề được trả lương cao nhất hiện nay: Mức lương trung bình của lập trình viên có kinh nghiệm là 30 triệu đồng/tháng, Còn lương khởi điểm lập trình viên dưới 2 năm kinh nghiệm là 7.7 triệu đồng/tháng, Các bạn có thể nhận được mức lương khủng hơn nữa nếu định hướng chuyên sâu thêm về trí tuệ nhân tạo (AI), Machine Learning, Khoa học dữ liệu: từ 35 đến 42,5 triệu đồng/tháng (CUSC có đào tạo), Mức lương đề xuất cho designer ở TB thấp là 8,6 triệu, còn chuyên viên giàu kinh nghiệm, mức lương cao nhất là 25 triệu/tháng, khi phát triển lên quản lý, mức lương còn tăng gấp nhiều lần. http://hrinsider.vietnamworks.com/kt_chuyen_nganh/top-nhung-nganh-duoc-tra-luong-cao-nhat-o-viet-nam-dac-biet-co-mot-nghe-luong-tu-130-180-trieu-thang-va-chua-bao-gio-het-hot http://kenh14.vn/top-8-nganh-nghe-danh-cho-dan-thiet-ke-do-hoa-sau-khi-ra-truong-2019100419355864.chn
15	Em được biết học phí khóa 1 của chương trình trên website. Vậy cho em hỏi học phí này có phát sinh gì thêm không và có chính sách hỗ trợ học phí dành cho sinh viên không?	Học phí khóa dài hạn Lập trình viên quốc tế - HK1 đã bao gồm giáo trình, đồng phục, balo, cơ sở vật chất, hỗ trợ học phí học Anh văn lấy chứng chỉ, phí hỗ trợ việc làm (không phát sinh chi phí thêm trong quá trình học): 3.47 triệu đồng/ tháng (tổng cộng 20 lần đóng), nếu đóng theo học kỳ giảm còn 16.8 triệu đồng/ HK (4 lần đóng), đóng trọn năm giảm còn 32.9 triệu đồng (2 lần đóng).
16	Thời gian bắt đầu của khóa học này là khi nào vậy ạ	Chương trình Lập trình viên Quốc tế khai giảng vào ngày 05/07/2021. Bắt đầu đăng kí từ ngày 07/06/2021, Kết thúc đăng kí vào ngày 05/07/2021. Thời gian học cụ thể như sau: Thứ 2 đến thứ 6
17	Cho em hỏi bằng đại học mà hệ vừa làm vừa học là như thế nào vậy?	Hệ Vừa làm vừa học là hình thức học với thời gian linh hoạt được sắp xếp để các em có thể học song song với chương trình Lập trình viên Quốc tế nhận 2 bằng. Bằng Đại học chính quy, Vừa làm vừa học hay liên thông hiện đều có giá trị ngang nhau. Chị gửi em xem thông tin về bằng Đại học trên trang Báo Điện tử Giáo dục Việt Nam Cơ quan chủ quản: Hiệp hội các Trường đại học, cao đẳng Việt Nam https://giaoduc.net.vn/giao-duc-24h/bang-dai-hoc-chinh-quy-tai-chuc-hay-lien-thong-deu-co-gia-tri-ngang-nhau-post199628.gd Trên các trang báo Thanh niên, Dân trí, Tuổi trẻ cũng có đăng em nhe

18	Thời gian bắt đầu của khóa học này là khi nào vậy ạ, học phí thế nào ạ	• Chương trình Lập trình viên Quốc tế khai giảng vào ngày 05/07/2021. Bắt đầu đăng kí từ ngày 07/06/2021, Kết thúc đăng kí vào ngày 05/07/2021. Thời gian học cụ thể như sau: Thứ 2 đến thứ 6 • Học phí khóa dài hạn Lập trình viên quốc tế - HK1 đã bao gồm giáo trình, đồng phục, balo, cơ sở vật chất, hỗ trợ học phí học Anh văn lấy chứng chỉ, phí hỗ trợ việc làm (không phát sinh chi phí thêm trong quá trình học): 3.47 triệu đồng/ tháng (tổng cộng 20 lần đóng), nếu đóng theo học kỳ giảm còn 16.8 triệu đồng/ HK (4 lần đóng), đóng trọn năm giảm còn 32.9 triệu đồng (2 lần đóng).
19	Em hoi ve khoa hoc quan tri mang doanh nghiep: Hoc phi bao nhieu Thoi gian dao tao Dao tao	• Chương trình "Chuyên viên quản trị mạng doanh nghiệp" được CUSC thiết kế nhằm đáp ứng nhu cầu thực tế về quản trị hệ thống mạng cho doanh nghiệp trên nền hệ điều hành Windows 10 & Windows Server 2016. Giúp học viên quản trị hiệu quả hệ thống mạng doanh nghiệp, xây dựng hạ tầng mạng có tính bảo mật cao, giúp tiết kiệm thời gian, giảm bớt các chi phí, và mang đến một nền tảng CNTT vững chắc cho doanh nghiệp. Để biết thêm thông tin chi tiết bạn có thể truy cập tại: https://aptech.cusc.vn/Default.aspx?tabid=309 • Học phí khóa ngắn hạn Chuyên viên Quản trị mạng Doanh nghiệp là 13.4 triệu đồng • Chương trình Quản trị mạng Doanh nghiệp khai giảng vào ngày 07/06/2021. Bắt đầu đăng kí từ ngày 06/04/2021, Kết thúc đăng kí vào ngày 07/06/2021. Thời gian học cụ thể như sau: 17:30-19:30 T2,4,6
20	Đặt giùm 2 cái pizza đi ạ	Xin lỗi, yêu cầu này nằm ngoài phạm vi tôi có thể trả lời. Hiện tại CUSC đào tạo các khóa học ngắn hạn như: An toàn, an ninh thông tin; Quản trị mạng doanh nghiệp; Thiết kế và cài đặt mạng; Bảo mật ứng dụng web; Điều tra tội phạm mạng; Phát triển ứng dụng Web với PHP và MySQL; Phát triển ứng dụng Web với ASP.NET MVC; Phát triển ứng dụng Web với Laravel và AngularJS; Phát triển ứng dụng Web với JAVA EE, Lập trình C# căn bản; Lập trình Java căn bản; Phát triển ứng dụng Android; An toàn và bảo mật thông tin cá nhân; Phát triển ứng dụng iOS; Thiết kế và quản trị website với Wordpress; Kiểm thử phần mềm; Lập trình Arduino cơ bản. Bên cạnh đó CUSC cũng đào tạo các khóa học dài hạn như: Lập trình viên quốc tế, Chuyên gia trí tuệ nhân tạo, Mỹ thuật đa phương tiện. Hoặc bạn có thể để lại tên và số điện thoại chuyên viên sẽ gọi và tư vấn cụ thể hơn cho bạn

Có thể thấy ở Bảng 3.3, 2 câu hỏi số 18 và 19, chatbot đã có thể trả lời các câu hỏi đa ý định từ người dùng. Từ kết quả thực nghiệm và quá trình xây dựng chatbot tôi rút ra những kết luận sau đây:

- Dữ liệu đào tạo chatbot và các kịch bản là yếu tố quan trọng nhất quyết định chatbot có thông minh hay không. Vì vậy, cần phải đưa chatbot vào thực nghiệm càng sớm càng tốt để đánh giá những gì chatbot có thể làm và những gì chưa làm được nhằm khắc phục cũng như thu thập thêm dữ liệu từ người dùng. Càng được tương tác với người dùng thực tế, gặp phải nhiều cách diễn đạt ý định khác nhau sẽ giúp chatbot thông minh hơn.
- Việc quyết định nên tách hay gộp ý định cũng rất quan trọng. Đối với các ý định nhập nhằng hoặc gần nhau có thể khiến chatbot lẫn lộn ý định tương đương với việc độ chính xác của chatbot giảm.
- Đối với những câu hỏi đa ý định nên thêm vào khi đó là những câu mà người dùng thường hỏi. Nếu thêm quá nhiều câu hỏi đa ý định vô nghĩa sẽ dẫn đến bùng nổ tổ hợp và khiến chatbot trở nên phức tạp hơn.
- Việc xác định bộ từ đồng nghĩa cũng rất quan trọng vì một số khóa học có thể có nhiều tên gọi khác nhau. Bộ từ đồng nghĩa đa dạng sẽ giúp cho chatbot nhận dạng thực thể tốt hơn.
- Đối với những câu hỏi bot dự đoán với độ tin cậy thấp hơn 70% hoặc ngoài phạm vi có thể trả lời bot sẽ đưa ra câu trả lời yêu cầu người dùng diễn đạt lại ý định hoặc gợi ý một số khóa học để đưa người dùng về phạm vi mà chatbot có thể trả lời được.

PHẦN KẾT LUẬN

1. Kết quả đạt được

- Xây dựng được hệ thống chatbot hỗ trợ tư vấn 23 khóa học hiện có tại Trung tâm Công nghệ phần mềm Đại học Cần Thơ
- Xử lý các câu hỏi đa ý định từ người dùng
- Xây dựng Module website Nhập học phí hỗ trợ cho chatbot
- Xây dựng giao diện chat trực tuyến giữa người dùng và chatbot

2. Hướng phát triển

- Phát triển thêm bộ câu hỏi và huấn luyện lại chatbot
- Phát triển thêm bộ từ đồng nghĩa của các khóa học
- Triển khai chatbot vào thực tế
- Liên tục đánh giá và kiểm thử chatbot để xử lý các ý định ngoài luồng hội thoại và các câu hỏi đa ý định

TÀI LIỆU THAM KHẢO

[1] S. Perez-Soler, S. Juarez-Puerta, E. Guerra, J. de Lara, "Choosing a chatbot development tool," *IEEE Software*, 2021.

[2] Đ.T. Nghị, H. Tùng, "Chatbot cho sinh viên Công nghệ thông tin," trong *Kỷ yếu Hội nghị KHCN Quốc gia lần thứ XII về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Huế, 2019*.

[3] N.T. Nghe, T.Q. Định, "Hệ thống hỗ trợ tư vấn tuyển sinh đại học," *Tạp chí Khoa học Trường Đại học Cần Thơ*, số Công nghệ TT 2015, tr. 152-159, 2015.

[4] Đ.V. Mạnh, T.Đ. Nghĩa, N.V. Anh, H.T.H. Vân, "Chatbot bán hàng tự động iBotSale," trong *Hội thảo quốc gia lần thứ XXIII: Một số vấn đề chọn lọc của Công nghệ thông tin và truyền thông, Quảng Ninh, 2020.*

[5] V.L.M. Nguyên, L.C. Tâm, N.V. Hưng, N.Đ.T. Nguyên, L.T.H. Hiến, L.T. N. Khiết, P.T. Trinh, "CODEBOT-Một Hệ thống chatbot trả lời các câu hỏi liên quan đến lập trình C++ và Python," *Tạp chí Khoa học*, 2021.

[6] N.T. Tiến, "*Nghiên cứu và xây dựng chatbot hỗ trợ người dùng trong ngân hàng*," Luận văn thạc sĩ, Trường Đại học Công nghệ, Hà Nội, 2019.

[7] T. Bocklisch, J. Faulkner, N. Pawlowski, A. Nichol, "Rasa: Open source language understanding and dialogue management," in *NIPS Workshop on Conversational AI, 2017.*

[8] A. Varswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems, 2017.*

[9] T. Bunk, D. Varshneya, V. Vlasov, A. Nichol, "Diet: Lightweight language understanding for dialogue systems," 2020.

[10] D.Q. Nguyen, A.T. Nguyen, "PhoBERT: Pre-trained language models for Vietnamese," *in EMNLP, 2020.*

[11] T. Mikolov, A. Joulin, E. Grave, P. Bojanowski, "Enriching Word Vectors with Subword Information," in *Transactions of the Association for Computational Linguistics, 2016.*

[12] L. Wu, A. Fisch, S. Chopra, K. Adams, A. Bordes, J. Weston, "StarSpace: Embed All The Things!," in *The Thirty-Second AAAI Conference on Artificial Intelligence, 2017.*

[13] Rasa Docs [Trực tuyến]. Địa chỉ: https://rasa.com/docs/ [Truy cập 11/09/2021].

[14] N.C. Hùng, Đ.H. Minh, "Về một phương pháp xác định mục tiêu văn bản trong tiếng Việt," *Thông tin công nghệ,* 2020.

[15] T. Nguyen, M. Shcherbakov, "Enhancing Rasa NLU model for Vietnamese chatbot," in *International Journal of Open Information Technologies, 2021*.