**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**
---


**CHU LÊ LONG**

# NGHIÊN CỨU, XÂY DỰNG CHATBOT HỎI ĐÁP THÔNG TIN KHÁCH SẠN SỬ DỤNG RASA FRAMEWORK

**LUẬN VĂN THẠC SĨ KỸ THUẬT**
***(Theo định hướng ứng dụng)***

HÀ NỘI - 2020

**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**
---

**CHU LÊ LONG**

# NGHIÊN CỨU, XÂY DỰNG CHATBOT HỎI ĐÁP THÔNG TIN KHÁCH SẠN SỬ DỤNG RASA FRAMEWORK

CHUYÊN NGÀNH: **KHOA HỌC MÁY TÍNH**
MÃ SỐ: **8.48.01.01**

## LUẬN VĂN THẠC SĨ KỸ THUẬT

***(Theo định hướng ứng dụng)***

NGƯỜI HƯỚNG DẪN KHOA HỌC

**PGS.TS. NGUYỄN MẠNH HÙNG**

HÀ NỘI – 2020

# LỜI CAM ĐOAN

Tôi là Chu Lê Long, học viên lớp Cao học khóa 2019 đợt 1, chuyên ngành Khoa học máy tính của trường Học viện Công nghệ Bưu chính Viễn thông.

Tôi xin cam đoan luận văn này là do tôi tự nghiên cứu, tìm hiểu, xây dựng. Nội dung của luận văn có tham khảo, sử dụng các thông tin và tài liệu từ các nguồn sách, tạp chí, bài báo được liệt kê trong danh mục các tài liệu tham khảo và được trích dẫn hợp pháp.

**Tác giả**

*(Ký và ghi rõ họ tên)*

**CHU LÊ LONG**

# LỜI CÁM ƠN

Em xin gửi lời cảm ơn tới các thầy cô giáo, cán bộ của Học viện Công nghệ Bưu chính Viễn thông nói chung đã giảng dạy, truyền đạt kiến thức cho em trong quá trình học tập và nghiên cứu chương trình Thạc sĩ.

Em xin gửi lời cảm ơn sâu sắc tới **PGS.TS. Nguyễn Mạnh Hùng** đã tận tình hướng dẫn, giúp đỡ và động viên em để hoàn thành luận văn "NGHIÊN CỨU, XÂY DỰNG CHATBOT HỎI ĐÁP THÔNG TIN KHÁCH SẠN SỬ DỤNG RASA FRAMEWORK".

Do kiến thức và kinh nghiệm thực tiễn còn hạn chế nên luận văn không tránh khỏi những thiếu sót nhất định. Em xin trân trọng tiếp thu các ý kiến của các thầy, cô để luận văn được hoàn thiện hơn.

Trân trọng cám ơn.

**Tác giả**

*(Ký và ghi rõ họ tên)*

**CHU LÊ LONG**

# MỤC LỤC

# DANH MỤC KÝ HIỆU VÀ CÁC CHỮ VIẾT TẮT

| Từ viết tắt | Từ chuẩn | Diễn giải |
|---|---|---|
| AI | Artificial Intelligence | Trí tuệ nhân tạo |
| API | Application Programming Interface | Giao diện lập trình ứng dụng |
| CRF | Conditional Random Fields | Mô hình CRF |
| DL | Deep learning | Học sâu |
| DM | Dialog Management | Quản lý hội thoại |
| FAQ | Frequently Asked Questions | Các câu hỏi thường gặp |
| ML | Machine Learning | Học máy |
| NLG | Natural language generation | Sinh ngôn ngữ tự nhiên |
| NLP | Natural Languague Processing | Xử lý ngôn ngữ tự nhiên |
| NLU | Natural language understanding | Hiểu ngôn ngữ tự nhiên |
| QA | Question Answering | Các cặp câu hỏi đáp |
| SDK | Software Development Kit | Bộ công cụ hỗ trợ phát triển |

# DANH MỤC HÌNH VẼ VÀ ĐỒ THỊ

# DANH MỤC BẢNG BIỂU

# MỞ ĐẦU

Ngày nay, cùng với sự phát triển của khoa học kỹ thuật, Chatbot đang được ứng dụng phổ biến và mạnh mẽ trong nhiều lĩnh vực, tạo nên một cơn sốt công nghệ khi có nhiều hãng công nghệ nổi tiếng thế giới tham gia như Google, Facebook, Microsoft, IBM...Theo Grand View Research, thị trường Chatbot dự kiến sẽ đạt khoảng 1,25 tỷ đô la trên toàn cầu vào năm 2025. Hơn nữa, các chuyên gia dự đoán rằng thị trường này sẽ tăng trưởng với tốc độ tăng trưởng gộp hàng năm hơn 24%. Đặc biệt là xu hướng chuyển dịch phát triển AI chatbot có khả năng hội thoại, xử lý những tương tác phức tạp hơn với khách hàng. Ở Việt Nam, chatbot đã bắt đầu được áp dụng ở trong một số lĩnh vực như chăm sóc khách hàng, mua sắm trực tuyến, trả lời thông tin ngân hàng, y tế… Đối với lĩnh vực du lịch, khách sạn chatbot chưa được sử dụng nhiều dù rằng đây là lĩnh vực rất phù hợp cho các ứng dụng chatbot. Chatbot có thể thay thế con người trong việc trả lời các câu hỏi có tính lặp đi lặp lại, hỗ trợ khách hàng 24/7, tiếp thị quảng cáo cho doanh nghiệp…

Với mong muốn hiểu sâu hơn về chatbot và các kỹ thuật giúp chatbot trả lời câu hỏi xử lý theo ngôn ngữ tự nhiên (NLP), em quyết định chọn đề tài "Nghiên cứu, xây dựng Chatbot hỏi đáp thông tin khách sạn sử dụng Rasa Framework" làm đề tài luận văn thạc sĩ. Qua đề tài em muốn nâng cao sự hiểu biết về AI Chatbot, NLP (xử lý ngôn ngữ tự nhiên) và nghiên cứu khả năng áp dụng thực tiễn tại Việt Nam.

Nội dung luận văn được chia ra làm 3 phần như sau:

Chương 1: Giới thiệu tổng quan về hệ thống chatbot, kiến trúc high-level và các thành phần cơ bản của AI chatbot, một số nền tảng và ứng dụng của chatbot.

Chương 2: Nghiên cứu một số kĩ thuật được sử dụng trong chatbot, tìm hiểu về Rasa Framework.

Chương 3: Trình bày về quá trình xây dựng chatbot trả lời thông tin khách sạn, thực nghiệm và đánh giá các kết quả.

# CHƯƠNG 1: GIỚI THIỆU TỔNG QUAN VỀ CHATBOT

## 1.1.Khái niệm

Theo từ điển Cambridge, chatbot là một một chương trình máy tính được thiết kế để trò chuyện với con người, đặc biệt là qua internet [15].

Chatbot thường trao đổi với người dùng qua hình thức tin nhắn hoặc âm thanh. Do được thiết kế để mô phỏng cách trò chuyện với con người, các hệ thống chatbot thường phải điều chỉnh và thử nghiệm liên tục.

Chatbot thường được sử dụng trong các hệ thống hội thoại cho các mục đích khác nhau bao gồm dịch vụ khách hàng, định tuyến yêu cầu hoặc để thu thập thông tin. Mặc dù một số ứng dụng chatbot sử dụng các phương pháp phân loại từ (word-classification), xử lý ngôn ngữ tự nhiên (NLP) và trí tuệ nhân tạo (AI), một số ứng dụng khác chỉ cần quét các từ khóa chung và tạo phản hồi bằng các cụm từ phổ biến thu được từ thư viện hoặc cơ sở dữ liệu liên quan.

Ngày nay, hầu hết các chatbot được truy cập trực tuyến thông qua cửa sổ popup của trang web hoặc thông qua các trợ lý ảo như Google Assistant, Amazon Alexa hoặc các ứng dụng nhắn tin như Facebook Messenger hoặc WeChat...

## 1.2.Lịch sử ra đời

Dưới đây là tóm tắt lịch sử ngắn gọn về lịch sử hình thành của chatbot [6], [16]. [17].

**1950**

Phép thử Turing là một phương pháp để xác định xem một cỗ máy có thể chứng minh trí thông minh của nó giống với não người hay không. Nếu một cỗ máy có thể tham gia vào một cuộc hội thoại với con người mà không bị phát hiện là một cỗ máy, thì nó đã thể hiện được trí tuệ của con người. Các phép thử Turing được thực hiện để xác định xem một chương trình máy tính có thể phân biệt được máy tính với con người trong một cuộc trò chuyện chỉ có văn bản thuần túy hay không. Bằng cách gõ câu hỏi cho cả hai đối tượng thử nghiệm, người thẩm vấn sẽ cố gắng xác định đối tượng nào là máy tính và đối tượng nào là con người. Máy tính sẽ vượt qua phép thử

Turing nếu người thẩm vấn không thể nói sự khác biệt giữa chủ thể con người và máy tính.

**1966**

Chatbot đầu tiên ra đời năm 1960, tên là Eliza, và là một chương trình máy tính của Joseph Weizenbaum (Viện Công nghệ Massachusetts, Mỹ). Chương trình được thiết kế theo cách bắt chước cuộc trò chuyện của con người. Chatbot Eliza hoạt động bằng cách chuyển các từ mà người dùng đã nhập vào máy tính và sau đó ghép nối chúng vào danh sách các câu trả lời có kịch bản. Nó sử dụng một kịch bản mô phỏng một nhà tâm lý trị liệu. Kịch bản được chứng minh là một tác động đáng kể đến việc xử lý ngôn ngữ tự nhiên và trí thông minh nhân tạo và là một trong những chương trình đầu tiên có thể vượt qua bài kiểm tra Turing.

**1972**

Parry được xây dựng bởi bác sĩ tâm thần người Mỹ Kenneth Colby vào năm 1972. Chương trình bắt chước một bệnh nhân tâm thần phân liệt. Nó là một chương trình ngôn ngữ tự nhiên tương tự như suy nghĩ của một cá nhân. Parry hoạt động thông qua một hệ thống phức tạp các giả định, phân bổ và "phản ứng cảm xúc" được kích hoạt bằng cách thay đổi trọng số được gán cho các đầu vào bằng lời nói. Trong cùng năm đó, Parry và Eliza đã "gặp" và "nói chuyện" với nhau tại Hội nghị Quốc tế về Truyền thông Máy tính ở Washington DC. Sau đó, Parry cũng đã vượt qua một phiên bản của Turing Test.

**1981**

Được phát triển vào những năm 1980 và phát hành trực tuyến vào năm 1997, chatbot Jabberwacky được thiết kế để "mô phỏng trò chuyện của con người tự nhiên theo cách thú vị và hài hước". Mục đích ban đầu của dự án Chatbot Jabberwacky là tạo ra một trí tuệ nhân tạo có khả năng vượt qua các phép thử Turing. Nó được thiết kế để bắt chước tương tác của con người và thực hiện các cuộc hội thoại với người dùng. Mục đích cuối cùng của chương trình là chuyển từ một hệ thống dựa trên văn bản sang toàn bộ hoạt động bằng giọng nói. Tác giả của nó tin rằng nó có thể được kết hợp vào các vật thể xung quanh nhà như robot, các thiết bị thông minh, …Trong

khi tất cả các chatbot trước đó dựa trên cơ sở dữ liệu tĩnh để trả lời và trò chuyện, Jabberwacky thu thập cụm từ được sử dụng bởi những người tham gia trò chuyện với nó. Nó tự thêm những câu trả lời vào cơ sở dữ liệu và tự động phát triển nội dung của riêng mình. Trong năm 2008, Jabberwacky đã phát hành một phiên bản mới và đổi tên thành Cleverbot.

**1992**

Được tạo ra bởi Creative Labs vào đầu những năm 1990, Dr SBAITSO là từ viết tắt của Sound Blaster Artificial Intelligent Text to Speech Operator. Dr. SBAITSO "trò chuyện" với người dùng như thể nó là một nhà tâm lý học. Mặc dù hầu hết các câu trả lời của nó đều là "WHY DO YOU FEEL THAT WAY?" nghĩa là "Bạn cảm thấy như thế nào?". Thay vì bất kỳ loại tương tác phức tạp, khi đối mặt với một cụm từ mà nó không thể hiểu được, nó thường trả lời là "THAT'S NOT MY PROBLEM" (Đó không phải là vấn đề của tôi).

**1995**

ALICE được xây dựng trên cùng một kỹ thuật được sử dụng để tạo nên ELIZA. ALICE ban đầu được sáng tạo bởi Richard Wallace, ra đời vào ngày 23 tháng 11 năm 1995. Chương trình được viết lại bằng ngôn ngữ Java vào năm 1998. ALICEBOT sử dụng một lược đồ XML có tên AIML (Artificial Intelligence Markup Language-Ngôn ngữ đánh dấu trí thông minh nhân tạo) để xác định các quy tắc trò chuyện heuristic. Tuy nhiên, nó lại không thể vượt qua Các phép thử Turing.

**2001**

SmarterChild là một Chatbot có sẵn trên mạng AOL Instant Messenger và Windows Live Messenger (trước đây là MSN Messenger). AOL Instant Messenger là một chương trình tin nhắn tức thời và hiện diện do AOL tạo ra, sử dụng giao thức nhắn tin tức thời OSCAR độc quyền và giao thức TOC để cho phép người dùng đăng ký giao tiếp trong thời gian thực. SmarterChild đóng vai trò giới thiệu cho việc truy cập dữ liệu nhanh và cuộc trò chuyện được cá nhân hóa thú vị hơn. Hơn nữa, khi kết hợp với các nhà mạng, chúng trở thành một kênh tiếp thị hiệu quả và miễn phí. Chúng

giúp người dùng giao tiếp nhanh chóng với hệ thống mạng bằng cách hiển thị các thông tin ngắn gọn với các lựa chọn trên bàn phím điện thoại.

**2006**

IBM Watson được tạo ra với mục tiêu vượt lên và chiến thắng các thí sinh tham dự cuộc thi Jeopardy! Với khả năng chạy hàng trăm thuật toán phân tích ngôn ngữ cùng một lúc, IBM Watson sở hữu một sự thông minh ngôn ngữ đáng ngạc nhiên. IBM thiết lập cho Watson có quyền truy cập vào cơ sở dữ liệu khổng lồ về thông tin. Watson có thể nhanh chóng truy cập 200 triệu trang dữ liệu, làm cho nó trở thành một máy trả lời câu hỏi lý tưởng (hoặc, trong trường hợp của Jeopardy, Watson trở thành máy tạo câu hỏi lý tưởng). Rõ ràng, một hệ thống có thể nhanh chóng lấy thông tin dựa trên đầu vào đàm thoại cũng có thể cung cấp nền tảng cho việc tạo các trợ lý ảo mạnh mẽ. Hiện nay, IBM Watson phục vụ như là "bộ não" cho nhiều chatbots hoạt động trên nhiều ngành công nghiệp và lĩnh vực trên khắp thế giới.

**2010-2016**

Siri, một trợ lý cá nhân thông minh, đã được ra mắt dưới dạng một ứng dụng iPhone và sau đó được tích hợp là một phần của iOS. Năm 2012, Google ra mắt chatbot Google Now. Năm 2016, Google đã giới thiệu trợ lý cá nhân thông minh mới Google Assistant, là một sự tiến hóa của Google Now. Trợ lý này có thể tham gia đối thoại hai chiều với người dùng. Trước đó vào năm 2014, Amazon phát hành Alexa, Microsoft giới thiệu Cortana vào 2015. Với khả năng phân tích và xử lý ngôn ngữ tự nhiên, các trợ lý ảo này kết nối với các dịch vụ web để trả lời các câu hỏi và đáp ứng các yêu cầu của người dùng.

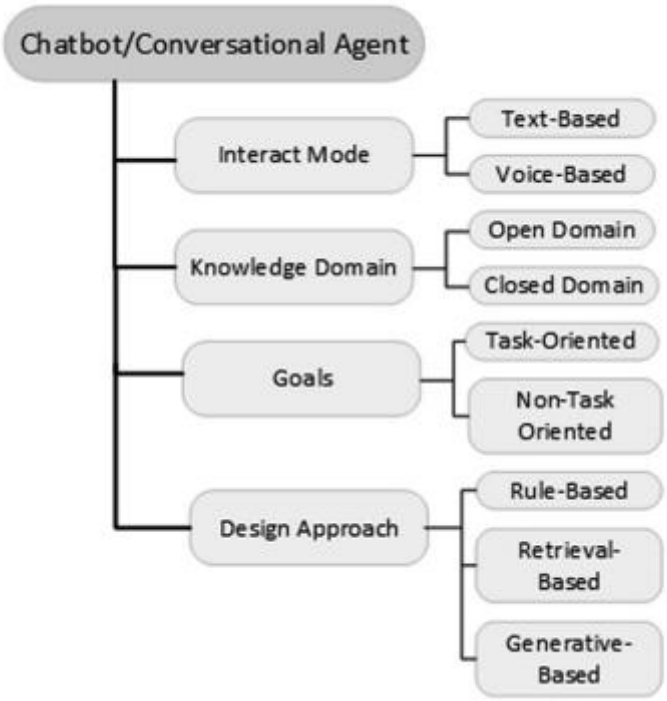
**2016- nay**

Facebook - mạng xã hội lớn nhất thế giới giới thiệu Messenger Platform. Một nền tảng thân thiện hơn và cho phép bất kỳ ai cũng có thể tạo cho mình một chatbot. Ngay sau đó, các ứng dụng chat khác như LINE, WhatsApp, Telegram hay Twitter cũng đưa ra các hỗ trợ hoặc các API cho phép người dùng tạo các Chatbot trên ứng dụng nhắn tin. Nhưng WeChat của Trung Quốc mới chính là tiên phong trong lĩnh vực này khi cho ra mắt Xiaoice - chatbot khá hoàn thiện từ năm 2013. Trong cuộc

đua của các nhà phát triển chatbots, Facebook đang nắm giữ thị phần toàn cầu lớn nhất vì có đến hơn 1 tỷ người sử dụng ứng dụng Messenger hàng tháng. Còn riêng ở thị trường Trung Quốc, WeChat lại là ứng dụng chat số 1 mà không ứng dụng chat nào có thể cạnh tranh nổi.

## 1.3.Phân loại chatbot

Chatbots có thể được phân loại thành nhiều loại khác nhau dựa trên một số tiêu chí. Các phân loại có thể được thực hiện dựa trên các tiêu chí sau [7, tr.947-954].




**Hình 1.1: Phân loại Chatbot [7]**

**- Theo chế độ tương tác (Interact Mode):**

+ Dựa trên văn bản (Text-Based)

+ Dựa trên giọng nói (Voice-Based)

**- Theo miền (Domain):**

+ Miền đóng/miền cụ thể (Closed Domain): Phạm vi của chatbot chỉ giải quyết một số vấn đề trong phạm vi nhất định. Ví dụ: Khách hàng mua ô tô, tư vấn khách hàng mua bảo hiểm nhân thọ, dự báo thời tiết… Loại này phổ

biến, dữ liệu huấn luyện trong phạm vi nhỏ nên dễ huấn luyện, độ chính xác cao.

+ Miền mở (Open Domain): Loại này là mục tiêu của trí tuệ nhân tạo. Một chatbot biết mọi thứ và có thể trả lời mọi vấn đề. Rất nhiều chatbot thông minh được tạo ra. Tuy nhiên trả lời mọi vấn đề và vượt qua được Turing test thì vẫn chưa thể đạt tới.

**- Theo mục tiêu (Goals):**

+ Các chatbot hướng nhiệm vụ (Task-Oriented): được thiết kế cho một nhiệm vụ cụ thể và được thiết lập để có thời gian ngắn các cuộc hội thoại, thường là trong một miền đóng.

+ Các chatbot không hướng nhiệm vụ (Non Task-Oriented): có thể mô phỏng cuộc trò chuyện với một người và thường thực hiện chat cho mục đích giải trí trong các miền mở.

**- Theo Phương pháp thiết kế (Design Approach):**

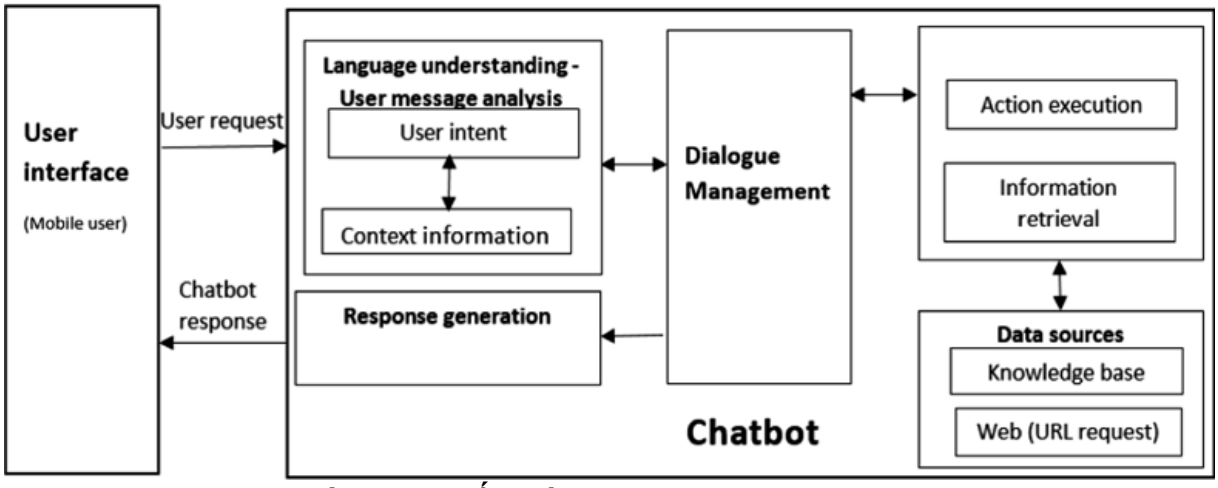
+ Dựa theo luật (Rule- Based): Loại chatbot này khả năng rất hạn chế. Chỉ có khả năng phản hồi chính xác những lệnh cụ thể mà ta đã xác định từ trước hoặc người dùng không được phép tùy ý phản hồi mà phải lựa chọn các phản hồi do lập trình viên tạo ra. Sự thông minh của chatbot phụ thuộc vào mức độ mà ta lập trình ra chatbot.

+ Dựa theo trí tuệ nhân tạo (AI): Loại này có khả năng "hiểu" ngôn ngữ. Nghĩa là chatbot không bị giới hạn bởi tập các luật xác định từ trước, mà có thể hiểu ở phạm vi rộng hơn. Tất nhiên chatbot vẫn phải được "học" từ dữ liệu có sẵn, nhưng nó có khả năng "đoán" được ý nghĩa và ngữ cảnh của những lệnh chưa từng gặp. Một khả năng nữa của chatbot dựa trên AI là khả năng "học thêm". Nghĩa là ta đưa vào càng nhiều câu huấn luyện, xác xuất chatbot phản hồi người dùng chính xác càng cao. Trong phương pháp này có thể chia ra thành dựa trên cơ sở sáng tạo (Generative-Based) như các mô hình theo trình tự, tạo ra trả lời phù hợp trong cuộc trò chuyện hoặc dựa trên truy xuất

(Retrieval-Based) để học lựa chọn các câu trả lời từ cuộc hội thoại hiện tại từ một kho lưu trữ.

## 1.4.Các thành phần cơ bản của hệ thống chatbot

Bước đầu tiên trong việc thiết kế bất kỳ hệ thống nào là chia nó thành các bộ phận cấu thành theo một tiêu chuẩn để có thể tuân theo cách tiếp cận phát triển mô đun. Trong hình dưới giới thiệu một kiến trúc chatbot chung của chatbot [8, tr.373-383].




**Hình 1.2: Kiến trúc chung của chatbot [8]**

Dưới đây trình bày chi tiết về các thành phần của chatbot.

### *1.4.1. NLU (Hiểu ngôn ngữ tự nhiên)*

NLU (Natural Language Understanding - hiểu ngôn ngữ tự nhiên): bao gồm việc xử lý ngôn ngữ tự nhiên (NLP) có nhiệm vụ xác định được ý định câu hỏi (intent classification) và trích chọn thông tin (slots filter).

NLU nhằm mục đích trích xuất ngữ cảnh (context) và ý nghĩa từ đầu vào của người dùng bằng ngôn ngữ tự nhiên, mà có thể không có cấu trúc và phản hồi một cách thích hợp theo ý định của người dùng (user intent). Nó xác định mục đích của người dùng và trích xuất các thực thể (entities) theo miền cụ thể. Cụ thể hơn, một ý định đại diện cho một ánh xạ giữa những gì người dùng nói và hành động (action) nên được thực hiện bởi chatbot. Các hành động tương ứng với các bước mà chatbot sẽ thực hiện khi các ý định cụ thể được kích hoạt bởi các đầu vào của người dùng và có thể có các tham số để xác định thông tin chi tiết về nó. Phát hiện ý định thường

được xây dựng dưới dạng phân loại câu, trong đó các nhãn ý định đơn hoặc nhiều ý định được dự đoán cho mỗi câu.

Thực thể là một công cụ để trích xuất các giá trị tham số từ các đầu vào ngôn ngữ tự nhiên. Ví dụ, hãy xem xét câu "What is the weather in Greece?". Mục đích của người dùng là tìm hiểu dự báo thời tiết. Giá trị thực thể là Greece (Hy Lạp). Do đó, người dùng yêu cầu dự báo thời tiết ở Hy Lạp. Các thực thể có thể do hệ thống xác định hoặc do nhà phát triển xác định.

Ngữ cảnh là các chuỗi lưu trữ ngữ cảnh của đối tượng mà người dùng đang đề cập hoặc nói đến. Ví dụ, một người dùng có thể tham chiếu đến một đối tượng đã được xác định trước đó trong câu sau của họ. Người dùng có thể nhập "Switch on the fan". Ở đây, ngữ cảnh sẽ được lưu là fan (cái quạt) để khi người dùng nói, "Switch it off" làm đầu vào tiếp theo, ý định "tắt" có thể được gọi trên ngữ cảnh "quạt".

Đúc kết lại, khi người dùng gõ một câu "What is the meaning of environment?" trong một chatbot sử dụng ứng dụng nhắn tin như Facebook, Slack, WhatsApp, WeChat hoặc Skype. Sau khi chatbot nhận được yêu cầu của người dùng, thành phần hiểu ngôn ngữ tự nhiên sẽ phân tích nó để suy ra ý định của người dùng và thông tin liên quan (ý định: dịch, thực thể: [từ: environment].

### *1.4.2. DM (Quản lý hội thoại)*

DM (Dialog Management - quản lý hội thoại): Thành phần quản lý đối thoại giữ và cập nhật ngữ cảnh của cuộc hội thoại là ý định hiện tại, các thực thể được xác định hoặc các thực thể bị thiếu cần thiết để thực hiện các yêu cầu của người dùng. Hơn nữa, nó yêu cầu thông tin thiếu, xử lý làm rõ bởi người dùng và đặt câu hỏi tiếp theo. Ví dụ: chatbot có thể phản hồi câu hỏi trên lại bằng câu: "Would you like to tell me as well an example sentence with the word environment?". Quản lý hội thoại cũng có nhiệm vụ xác định được hành động (action) tiếp theo dựa vào trạng thái hành động trước đó hay ngữ cảnh hội thoại. Các ngữ cảnh này phải được đối chiếu trong các kịch bản dựng sẵn (history) đã đào tạo cho bot. Thành phần này cũng đảm nhiệm việc lấy dữ liệu từ hệ thống khác qua các API/Data sources gọi trong action.

### *1.4.3. NLG (Sinh ngôn ngữ tự nhiên)*

NLG (Natural Language Generator - Sinh ngôn ngữ tự nhiên): là thành phần sinh ngôn ngữ dựa vào chính sách (policy) và hành động được xác định trong DM thông qua các tập hội thoại.

Khi phản hồi, NLG chuẩn bị phản hồi giống ngôn ngữ tự nhiên cho người dùng dựa trên ý định và thông tin ngữ cảnh. Các câu trả lời thích hợp được tạo ra bởi một trong các mô hình thiết kế theo luật hoặc theo AI.

## 1.5. Một số nền tảng phát triển chatbot

**- Dialogflow (https://dialogflow.com/)**

- Cung cấp bởi Google
- Trước đây được gọi là Api.ai và rất phổ biến rộng rãi trong cộng đồng chatbot.
- Cung cấp cho người dùng những cách mới để tương tác với sản phẩm của họ bằng cách xây dựng giao diện đàm thoại dựa trên giọng nói và văn bản hấp dẫn bằng AI.
- Kết nối với người dùng trên Google Assistant, Amazon Alexa, Facebook Messenger và các nền tảng và thiết bị phổ biến khác.
- Có khả năng phân tích và hiểu ý định của người dùng để giúp bạn phản hồi theo cách hữu ích nhất.

**- Rasa (https://rasa.com/)**

- Rasa Open Source là một nền tảng để tự động hóa các trợ lý dựa trên văn bản và giọng nói sử dụng học máy.
- Có thể thực hiện các hành động mà bot có thể thực hiện bằng mã Python.
- Thay vì một loạt các câu lệnh if…else khác, logic của bot dựa trên một mô hình học máy được đào tạo trên các ví dụ hội thoại.

**- Wit.ai (https://wit.ai)**

- Được Facebook mua lại trong vòng 21 tháng kể từ khi ra mắt, nhóm wit.ai đóng góp cho công cụ NLP của Facebook trong Facebook.

- Wit.ai giúp các nhà phát triển dễ dàng xây dựng các ứng dụng và các thiết bị mà người dùng có thể nói chuyện hoặc nhắn tin tới.
- Có thể sử dụng wit.ai để xây dựng chatbot, tự động hóa nhà, ...

**- Microsoft Bot Framework (https://dev.botframework.com/)**

- Được cung cấp bởi Microsoft.
- Microsoft Bot Framework có khả năng hiểu ý định của người dùng
- Có thể kết hợp LUIS để hiểu ngôn ngữ tự nhiên, Cortana cho giọng nói và API Bing cho tìm kiếm.

**- Woebot (https://woebot.io/ )**

- Có thể theo dõi tâm trạng của người dùng
- Giúp người dùng cảm thấy tốt hơn
- Cung cấp cái nhìn sâu sắc bằng cách xem mô hình tâm trạng
- Dạy người dùng làm thế nào để tích cực và năng lượng cao

**- Chatfuel (https://chatfuel.com/ )**

- Chatfuel ra đời với mục tiêu làm cho việc xây dựng bot trở nên dễ dàng với bất kỳ ai.
- Không cần biết lập trình cũng có thể tạo ra chatbot
- Đây nền tảng hàng đầu để xây dựng bot trên Facebook Messenger.

## 1.6.Một số ứng dụng của chatbot

Dưới đây là một số ứng dụng điển hình của chatbot trong các lĩnh vực đời sống [19].

**Thương mại điện tử**: Thương mại điện tử bắt đầu dùng chatbot theo nhiều cách khác nhau. Thương hiệu quần áo H&M đã tạo một chatbot trên Kik, đặt câu hỏi cho người dùng về phong cách của họ và cung cấp các tùy chọn ảnh cho người dùng lựa chọn. Với thông tin này, bot tạo ra một hồ sơ thời trang của mỗi người dùng để đưa ra gợi ý trang phục và hướng người dùng mua quần áo. Chatbot LEGO giúp người mua hàng giải quyết một vấn đề khó khăn khi chọn món quà phù hợp. Ralph the Gift Bot sẽ cung cấp các đề xuất quà tặng được cá nhân hóa cho tất cả người dùng trực tiếp trong Messenger. Ralph chọn đề xuất quà tặng dựa trên cách người dùng trả

lời các câu hỏi trong bot. Nó bắt đầu bằng cách hỏi những câu hỏi đơn giản, như địa điểm, tuổi của người bạn mua và ngân sách quà tặng. Khi bot có các chi tiết này, nó cho phép người dùng chọn chủ đề của sản phẩm họ muốn mua (phiêu lưu, du lịch, …). Khi người dùng đã tìm thấy một sản phẩm họ thích, họ sẽ nhận được một liên kết tự động thêm sản phẩm vào giỏ hàng của họ trên trang web Lego, để họ có thể mua sản phẩm đó.

**Nhà hàng và dịch vụ ăn uống:** Một trong những ngành hàng thành công và có kinh nghiệm nhất về việc sử dụng chatbot chính là ngành dịch vụ ăn uống. Phổ biết nhất chính là các chuỗi cửa hàng pizza. Domino's hiện nay có hơn một nửa đơn đặt hàng thông qua các kênh digital và điều này tạo ra lợi nhuận khoảng 5 tỷ đô la trong một năm. Thương hiệu này cũng chính là người tiên phong trong việc áp dụng công nghệ chatbot trong Facebook Messenger. Chatbot của Domino's được kết hợp với tính năng Easy Order, cho phép khách hàng đặt pizza yêu thích chỉ với cú nhấp chuột.

**Tài chính và ngân hàng**: Nhiều công ty hiện đang nắm bắt công nghệ khi họ muốn tái tạo lại cách tương tác với khách hàng. Chatbot là một trong những công nghệ mà hàng loạt các công ty trong ngành chạy đua để sử dụng. Trên thực tế, theo Samsung, có hơn 50% các công ty dịch vụ tài chính, ngân hàng làm việc trong một dự án chatbot. Thông qua chatbot, khách hàng có thể thực hiện các thao tác như kiểm tra số dư trong tài khoản, chuyển tiền và báo với ngân hàng về việc mất thẻ tín dụng, thẻ ghi nợ nhờ vào cải tiến công nghệ xử lí ngôn ngữ.

**Phương tiện truyền thông tin tức**: Theo một nghiên cứu được công bố bởi The Pew Research Center, 67% người trưởng thành ở Mỹ thấy tin tức của họ trên các phương tiện truyền thông và 20% còn lại có xu hướng dùng mạng xã hội để tìm hiểu những gì đang xảy ra xung quanh trên thế giới. Với lý do đó, không có gì ngạc nhiên khi các tổ chức tin tức đã chứng tỏ sự quan tâm trong việc phát triển chatbot để đưa các tin tức cho các social messaging platform. Điển hình là CNN, phát triển chatbot cho Facebook Messenger, Kik và LINE. Thay vì sử dụng chatbot để phân phối các tiêu đề hoặc liên kết đến các bài báo, CNN đang tạo ra những trải nghiệm tương tác

mới. Ví dụ: Trên Facebook Messenger, người dùng có thể hỏi chatbot của CNN về tin tức trên Kik, họ có thể tiếp cận được nhiều thông tin hơn qua định dạng "choose-your-own-adventure".

**Y tế**: Một chatbot về y tế được vận hành bởi công ty khởi nghiệp về sức khỏe kỹ thuật số HealthTap, ban đầu được ra mắt dưới dạng dịch vụ tập trung vào hỏi đáp nhưng giờ đây được gọi là "Global Health Practice" đầu tiên trên thế giới. Thông qua chatbot HealthTap trên Facebook Messenger, các cá nhân có thể đặt câu hỏi về sức khỏe và nhận câu trả lời dựa trên cơ sở dữ liệu Hỏi & Đáp rộng rãi của HealthTap.

**Hàng không**: Người dùng có thể nhận tài liệu chuyến bay của mình qua Messenger, bao gồm xác nhận đặt vé, thông báo đăng ký, thẻ lên máy bay, và cập nhật trạng thái chuyến bay. Ngày càng nhiều hãng hàng không sử dụng chatbot để phản hồi các câu hỏi phổ biến và theo dõi thông tin cơ bản về chuyến bay. Hãng Aeroméxico đã tung ra một chatbot trên Facebook Messenger, và đã giúp hãng phục vụ 1.000 khách hàng mỗi ngày với chi phí thấp hơn so với mức lương của hai nhân viên cũng sẽ xử lý một khối lượng yêu cầu như vậy.

## 1.7.Giới thiệu chatbot trả lời thông tin du lịch, khách sạn

Chatbot ngày càng được áp dụng nhiều trong lĩnh vực du lịch, khách sạn. Trên thực tế, mục đích của chatbot là hỗ trợ chăm sóc và mở rộng quy mô quan hệ khách hàng, tăng trải nghiệm hiện có của khách hàng. Bằng cách này, doanh nghiệp du lịch, khách sạn có thể tiết kiệm được nhiều tiền, và đó là lý do tại sao nhiều doanh nghiệp đang áp dụng công nghệ này. Dưới đây là một số ví dụ điển hình áp dụng chatbot trong lĩnh vực du lịch, khách sạn [18].

**The Cosmopolitan of Las Vegas:**

Vào tháng 1 năm 2017, The Cosmopolitan of Las Vegas đã giới thiệu Rose, một chatbot thông minh, cung cấp dịch vụ khách hàng cho khách thông qua tin nhắn. Khách hàng có thể nhắn tin cho Rose để nhận ngay các đề xuất của nhà hàng và quán bar, có các tiện nghi như gối thêm được giao đến phòng của họ, chơi trò chơi hoặc thậm chí nhận các chuyến tham quan có hướng dẫn xung quanh khu nghỉ mát, nơi các nhân viên phục vụ xử lý các yêu cầu được gửi để Rose đằng sau hậu trường. Tất

cả khách của khách sạn được giới thiệu với Rose bằng cách nhận thẻ có chìa khóa tại bàn đăng ký và được khuyến khích hỏi Rose bất cứ điều gì trong thời gian lưu trú của họ. Hầu hết các khuyến nghị mà Rose cung cấp là dành cho năm quán bar và 20 nhà hàng nằm trong The Cosmopolitan, nơi tất cả khách của khách sạn được đối xử như khách VIP và thường không phải xếp hàng chờ đợi hoặc trả thêm phí vào cửa cho câu lạc bộ đêm Marquee nổi tiếng của mình. Ngoài việc cung cấp các đề xuất về nhà hàng và quán bar, Rose cũng có thể chơi một vài trò chơi

**Marriott Hotels:**

Thương hiệu Marriott International's Aloft Hotels đã giới thiệu chatbot được gọi là ChatBotlr; nó có sẵn thông qua tin nhắn văn bản cho khách và cho phép họ thực hiện các yêu cầu dịch vụ trực tiếp từ điện thoại thông minh của họ - bất cứ nơi nào và bất cứ lúc nào khi họ có nhu cầu.

Khách có thể yêu cầu ChatBotlr mang đồ vệ sinh đến phòng của họ hoặc gọi cho họ vào buổi sáng, để trả lời các câu hỏi về các tiện nghi của khách sạn, hoặc kết nối chúng với danh sách nhạc của Aloft Hotels, và tải thêm. Các thành viên của Marriott Rewards trên Facebook Messenger và Slack có thể nghiên cứu và đặt chỗ du lịch tại hơn 4700 khách sạn, liên kết tài khoản Marriott Awards và SPG của họ, lên kế hoạch cho kỳ nghỉ sắp tới của họ với các bài viết từ tạp chí kỹ thuật số Marriott Traveller và trò chuyện trực tiếp với trung tâm liên kết khách hàng. Bằng cách tận dụng công nghệ, khách sạn có thể tăng cường cá nhân hóa, mở rộng sự lựa chọn và kết nối với khách hàng, nâng cao của trải nghiệm của khách.

**Hyatt Hotels:**

Khách sạn Hyatt, bắt đầu sử dụng Facebook Messenger vào 2015 để trả lời các câu hỏi của khách truy cập, cho phép họ đặt phòng và kiểm tra phòng trống, sử dụng nhân viên quan hệ khách hàng của mình để trợ giúp khách hàng.

Hyatt rất quan tâm và hợp tác với Conversocial, một đối tác tiếp thị của Facebook, để thử nghiệm ứng dụng này như một kênh dịch vụ khách hàng mới. Ví dụ với Hyatt, mọi người có thể đặt phòng bằng ứng dụng hoặc đơn giản là họ có thể tham gia trò chuyện trực tiếp với nhóm Hyatt để đặt câu hỏi dưới dạng miễn phí, như điều gì sẽ có sẵn vào các ngày lễ cụ thể hoặc về bất kỳ ưu đãi nào đang diễn ra bao gồm các tiện nghi của khách sạn.

Trên Messenger cũng cho phép nhận hóa đơn giao dịch, xác nhận, cập nhật giao hàng và các thông điệp quan trọng khác cũng như các hành động cơ bản diễn ra trong một cuộc trò chuyện, được gửi trực tiếp đến khách hàng gọi điện thoại và loại bỏ những rắc rối khi đào qua email hoặc ghi nhớ mật khẩu trang web. Chatbot Messenger đã hỗ trợ Hyatt tăng doanh số, đặt phòng khách sạn và giải quyết thắc mắc của khách hàng.

## 1.8. Kết luận chương

Qua các ví dụ trên, ta có thể nhận thấy bài toán chatbot trả lời thông tin du lịch, khách sạn chỉ cần tập trung vào trả lời các câu hỏi đối thoại liên quan đến một miền cụ thể xoay xung quanh các thông tin về khách sạn, nhằm cố gắng để đạt được các mục tiêu thông tin rất cụ thể. Đây là bài toán trong miền đóng (Closed domain).

Ngoài ra, dù có nhiều nền tảng hỗ trợ xây dựng chatbot nhưng trong luận văn này, tác giả lựa chọn Rasa framework để xây dựng thực nghiệm công cụ chatbot minh họa với các lý do sau:

- Hiện tại hầu hết các nền tảng lớn như Dialogflow (Google), wit.ai, Microsoft Bot Framework… do các hãng lớn cung cấp, có nhiều công cụ xây dựng và tích hợp thông qua API nhưng các thuật toán và dữ liệu người dùng sẽ lưu trên các nền tảng này, khó làm chủ hệ thống trong khi đó việc sử dụng mã nguồn mở Rasa để xây dựng hệ thống Chatbot giúp nắm rõ hơn các công nghệ phía sau chatbot, làm chủ dữ liệu và thông tin người dùng.

- Rasa thực sự dễ tiếp cận cho người mới bắt đầu, ngay cả người không biết gì về chatbot hay NLP cũng có thể làm quen sử dụng. Hầu hết công việc của người sử dụng là tập trung xây dựng nội dung kịch bản, khả năng của chatbot chứ không cần thiết phải quan tâm đến công nghệ xây dựng.

- Rasa hoạt động khá tốt và mạnh mẽ, đặc biệt trong vấn đề xác định ý định người dùng (intent) và xác định thực thể (entity).

- Mã nguồn của Rasa là mã nguồn mở, do đó Rasa giúp ta biết chính xác được đang làm gì với chatbot của mình, thậm chí có thể tùy biến chatbot theo mong muốn của bản thân, ví dụ như: tùy biến hành động (action), quyết định nền tảng tin nhắn (messenger) của chatbot…

# CHƯƠNG 2: GIỚI THIỆU MỘT SỐ KỸ THUẬT SỬ DỤNG TRONG CHATBOT VÀ RASA FRAMEWORK

## 2.1. Một số kỹ thuật sử dụng trong chatbot

### *2.1.1 Xác định ý định người dùng*

#### a. Xác định ý định (intent)

Intent: điều người dùng mong muốn chatbot thực hiện (hỗ trợ) khi đưa ra câu hội thoại.

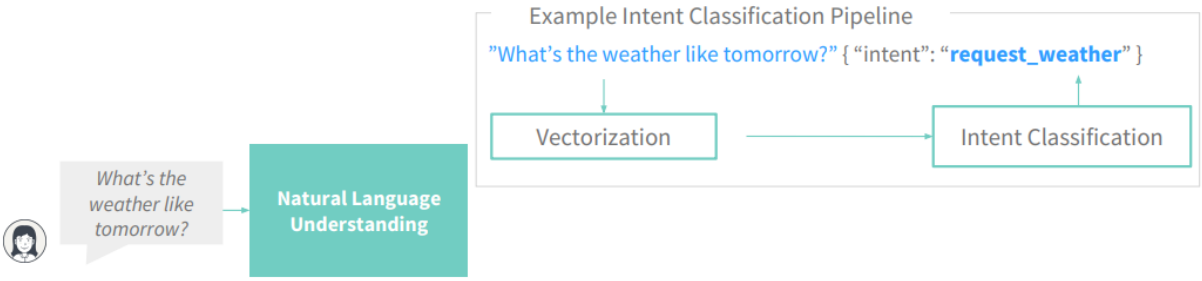
Ví dụ: Khi người dùng muốn hỏi về thông tin các loại phòng của khách sạn:

Khách sạn mình có những loại phòng nào vậy ad?

Khách sạn mình có phòng đơn không?

Khách sạn mình có phòng twin không?

Hoặc khi một người dùng hỏi "What's the weather like tomorrow?" thì chatbot cần hiểu được ý định của họ là hỏi về thời tiết (request weather) [4].




**Hình 2.1 Xác định ý định người dùng [4]**

Intent được xác định sẽ quyết định cấu trúc (frame) và kịch bản (script) của đoạn hội thoại tiếp theo. Việc xác định ý định là rất quan trọng đối với chatbot. Nếu chatbot xác định sai intent sẽ dẫn đến phản hồi không thích hợp dẫn đến người dùng không hài lòng và có thể rời bỏ hệ thống.

#### b. Các vấn đề khi xác định ý định

**- Thiếu nguồn dữ liệu:**

Với sự phát triển của công nghệ trí tuệ nhân tạo, các công ty Internet lớn đã ra mắt chatbot. Do trải nghiệm người dùng ít hơn, hầu hết các nhà nghiên cứu khó có được văn bản trò chuyện giữa người dùng và chatbot, dẫn đến số lượng văn bản đối

thoại được nghiên cứu hạn chế, trong đó trở thành một vấn đề lớn phải đối mặt với các nhiệm vụ phát hiện ý định. Trong quá trình thực tế phát hiện ý định, có rất ít văn bản có mục đích với chú thích và chúng rất khó lấy, điều này cũng mang lại những thách thức đối với nghiên cứu và phát triển phát hiện ý định [10].

**- Sự bất quy tắc trong diễn đạt của người dùng:**

Trong hệ thống chat, mục đích của người dùng nói chung được đặc trưng bởi diễn đạt kiểu dạng nói thông thường, câu ngắn và nội dung rộng, điều này gây khó khăn cho việc xác định mục đích của người dùng. Ví dụ, "tôi muốn tìm một địa điểm ăn tối", mục đích tương ứng của câu này là "tìm kiếm một nhà hàng", vì vậy câu với ý định thông tục làm cho miền chủ đề không rõ ràng, gây khó khăn cho việc xác định mục đích của người dùng. Hoặc với câu "tôi muốn đặt vé", có thể đặt vé máy bay, vé tàu, vé xe buýt ... Do diễn đạt của người dùng quá rộng, máy không thể đưa ra phản hồi cho người dùng kịp thời [10].

**- Phát hiện ý định ngầm:**

Với việc mở rộng liên tục phạm vi ứng dụng của hệ thống đối thoại giữa người và máy, ngày càng có nhiều cách để thể hiện ý định. Theo các loại diễn đạt, ý định có thể được chia vào ý định rõ ràng (explicit intents) và ý định ngầm định (implicit intents). Ý định rõ ràng đề cập đến việc người dùng chỉ ra rõ ràng hoặc yêu cầu ý định trong nội dung hội thoại đã bao gồm miền chủ đề (topic domain), thể loại ý định, ... Ý định ngầm định đề cập đến thực tế là người dùng không có yêu cầu mục đích rõ ràng và cần phải suy luận người dùng có ý định thực sự bằng cách phân tích ý định. Chẳng hạn "Book a hotel near the People's Park for one night" và câu ngầm định "I'm going to Shenzhen for two days next week". Mặc dù ý định của họ là đặt phòng khách sạn, nhưng đối với khách hàng sau cần đánh giá và suy đoán về ý định thực sự của người dùng. Do đó, phát hiện ý định ngầm mà không có chủ đề rõ ràng là rất khó khăn trong nhiệm vụ phát hiện ý định [10].

**- Phát hiện đa ý định (Multiple intents detection):**

Trong câu hội thoại con người sẽ có những nội dung với đa ý định. Ví dụ nếu bạn nói "xin chào, cho tôi đặt phòng nhé" thì bot phải xác định được 2 ý định "chào

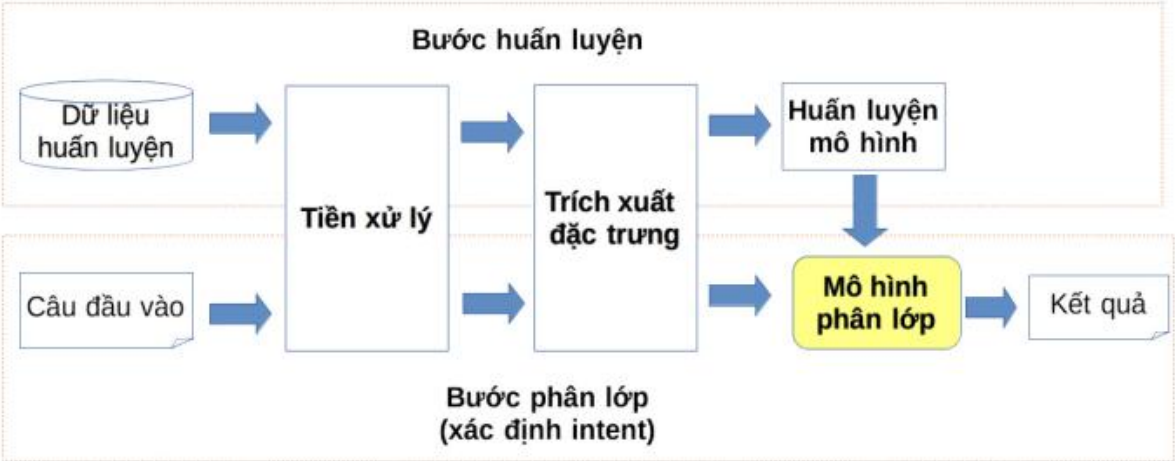
hỏi" và "đặt phòng" trong câu nói người dùng. Nếu bot có thể hiểu và trả lời được câu hỏi loại này sẽ giúp việc tương tác với bot trở nên tư nhiên hơn. Làm thế nào để phát hiện đa ý định của người dùng trong văn bản ngắn là một khó khăn của phát hiện ý định [10].

**- Phát hiện lỗi chính tả, bỏ dấu…**

Đối với tiếng Việt còn đối mặt với sự nhập nhằng chẳng hạn khi người dùng gõ tiếng Việt không dấu, nhầm lẫn s, x, l, n… hoặc gõ sai lỗi chính tả cũng sẽ gây khó khăn lớn cho phát hiện ý định [1].

### c. Xác định ý định dựa trên học máy (machine learning)

Các bước xác định ý định dựa trên học máy được minh họa như hình dưới [1].




**Hình 2.2: Xác định ý định dựa trên học máy [1]**

Hệ thống phân lớp ý định người dùng có một số bước cơ bản:

**- Tiền xử lý dữ liệu:**

Bước tiền xử lý dữ liệu chính là thao tác "làm sạch" dữ liệu như: loại bỏ các thông tin dư thừa, chuẩn hoá dữ liệu và chuyển các từ viết sai chính tả thành đúng chính tả, chuẩn hoá các từ viết tắt… Bước tiền xử lý dữ liệu có vai trò quan trọng trong hệ thống chatbot. Nếu dữ liệu đầu vào có xử lý ở bước này thì sẽ làm tăng khả năng năng độ chính xác cũng như sự thông minh cho bot.

Một số kỹ thuật tiền xử lý:

+ Tách từ (word segmentation): như tách câu thành các token.

+ Xử lý các từ đồng nghĩa

+ Xử lý từ gõ sai chính tả (ví dụ mạng chaamj)

+ Xử lý từ viết tắt (ví dụ: gõ ip thay vì iphone)

**- Trích xuất đặc trưng:**

Tiếp đến là bước trích xuất đặc trưng (feature extraction hay feature engineering) từ những dữ liệu đã được làm sạch. Trong mô hình học máy truyền thống (trước khi

mô hình học sâu được áp dụng rộng rãi), bước trích xuất đặc trưng ảnh hưởng lớn đến độ chính xác của mô hình phân lớp. Để trích xuất được những đặc trưng tốt, chúng ta cần phân tích dữ liệu khá tỉ mỉ và cần cả những tri thức chuyên gia trong từng miền ứng dụng cụ thể. Đây là bước biểu diễn ngôn ngữ loài người dưới dạng số sao cho có nghĩa và machine (máy) có thể hiểu được.

Một số kỹ thuật trích xuất đặc trưng:

+ Word2Vec

+ One-hot Encoding

+ Bag of Words

+ TD/IDF…

**- Huấn luyện mô hình và Mô hình phân lớp:**

Bước huấn luyện mô hình nhận đầu vào là các đặc trưng đã được trích xuất và áp dụng các thuật toán học máy để học ra một mô hình phân lớp. Các mô hình phân lớp có thể là các luật phân lớp (nếu sử dụng decision tree) hoặc là các vector trọng số tương ứng với các đặc trưng được trích xuất (như trong các mô hình logistic regression, SVM, hay mạng Neural).

Một số kỹ thuật phân lớp:

Support Vector Machines (SVM)

Random Forest

Neural Networks (LSTM)…

Sau khi có một mô hình phân lớp intent, chúng ta có thể sử dụng nó để phân lớp một câu hội thoại mới. Câu hội thoại này cũng đi qua các bước tiền xử lý và trích

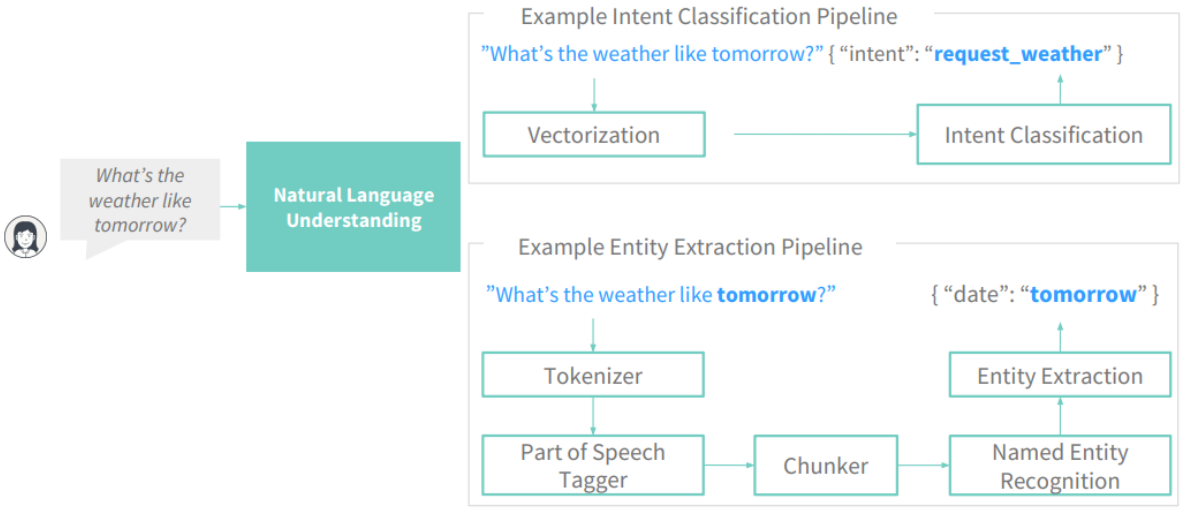
xuất đặc trưng, sau đó mô hình phân lớp sẽ xác định "điểm số" cho từng intent trong tập các intent và đưa ra intent có điểm cao nhất.

### *2.1.2 Trích xuất thông tin*

#### a. Giới thiệu Trích xuất thông tin thực thể (NER)

Named Entity Recognition (NER): đây là tác vụ cơ bản trong lĩnh vực xử lý ngôn ngữ tự nhiên. Vai trò chính của tác vụ này là nhận dạng các cụm từ trong văn bản và phân loại chúng vào trong các nhóm đã được định trước như tên người, tổ chức, địa điểm, thời gian, loại sản phẩm, nhãn hiệu, ...

Vẫn ở ví dụ đề cập ở trên, khi người dùng hỏi "What's the weather like tomorrow?" thì chatbot ngoài cần hiểu được ý định của họ là hỏi về thời tiết, còn cần xác định được thực thể xác định ở thời gian (date) là tomorrow (ngày mai) [4].




**Hình 2.3: Trích xuất thông tin thực thể [4]**

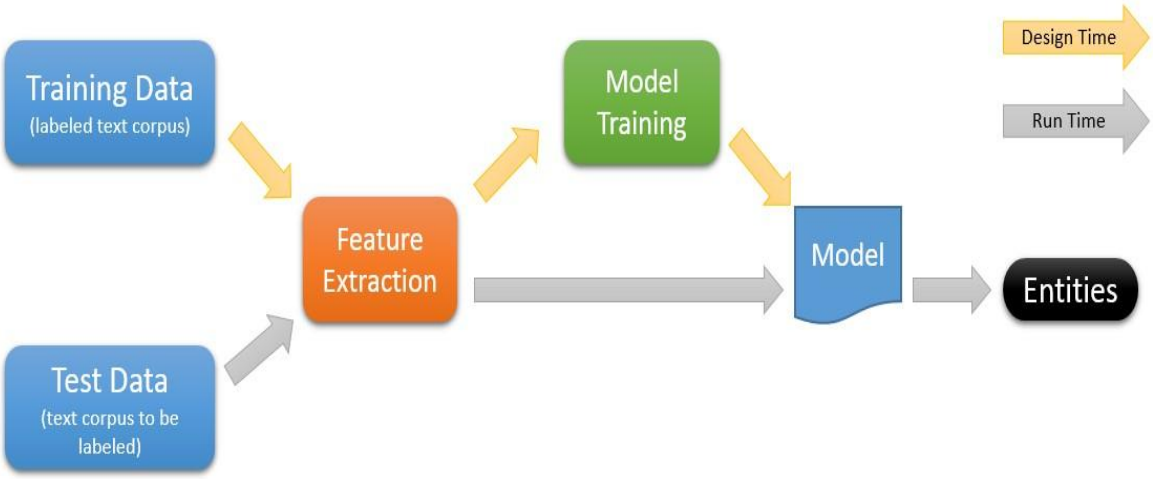
Các loại thực thể mà chatbot thường hỗ trợ:

- Vị trí (Location)
- Thời gian (Datetime)
- Số (Number)
- Địa chỉ liên lạc (Contact)
- Khoảng cách (Distance)
- Khoảng thời gian (Duration)

**b. Trích xuất thông tin thực thể dựa trên học máy**

Có một số phương pháp trích xuất thông tin thực thể như phương pháp tiếp cận từ vựng (lexicon approach), hệ thống dựa trên quy tắc (rule-based systems), và phương pháp dựa trên học máy cũng rất phổ biến.

Các hệ thống dựa trên học máy học cách nhận biết các thực thể trong văn bản dựa trên các ví dụ (examples) trước đây mà chúng đã thấy.




**Hình 2.4: Trích xuất thông tin thực thể dựa trên học máy**
*(Nguồn: https://docs.microsoft.com/en-us/archive/blogs/machinelearning/machine-learning-and-text-analytics)*

Để xây dựng một trình trích xuất thực thể, chúng ta cần cung cấp cho mô hình một khối lượng lớn dữ liệu huấn luyện có chú thích, để nó có thể tìm hiểu thực thể là gì. Chẳng hạn, nếu ta muốn xây dựng một mô hình để trích xuất các vị trí "locations", chúng ta cần gắn nhãn thủ công tên của các thành phố, quốc gia, địa điểm, v.v. Càng nhiều ví dụ được gắn nhãn, mô hình sẽ càng chính xác. Với phương pháp này, mô hình trở nên thông minh hơn theo thời gian khi nó học hỏi từ các ví dụ mới.

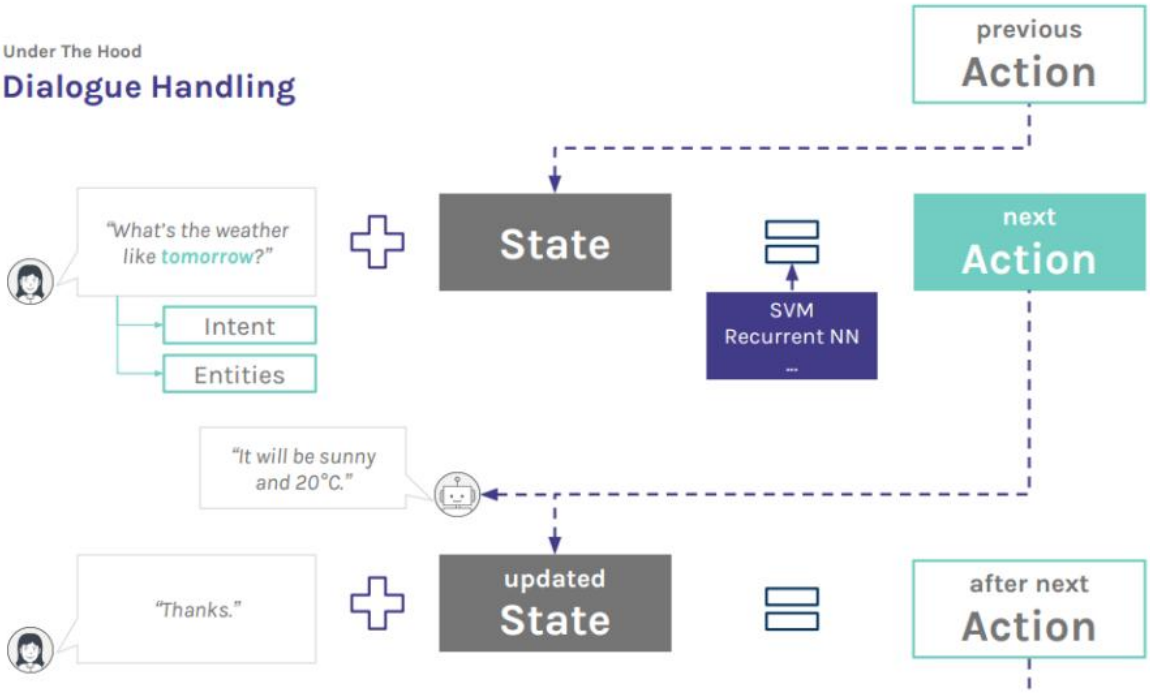
Các kỹ thuật thường được sử dụng:

- Hidden Markov Model
- Maximum Entropy
- Conditional Random Fields – CRFs…

### *2.1.3 Quản lý hội thoại*

**a. Vai trò của quản lý hội thoại**

Trong các phiên trao đổi dài (long conversation) giữa người và chatbot, chatbot sẽ cần ghi nhớ những thông tin về ngữ cảnh (context) hay quản lý các trạng thái hội thoại (dialog state). Vấn đề quản lý hội thoại (dialoge management) khi đó là quan trọng để đảm bảo việc trao đổi giữa người và máy là thông suốt [1]. Vẫn ở ví dụ về hỏi về thời tiết ở trên, chatbot cần có thông tin về trạng thái hội thoại, để có thể phản hồi đúng thông tin cần hỏi "It will be sunny and $20^{\circ}C$ " [4].




**Hình 2.5: Quản lý hội thoại [4]**

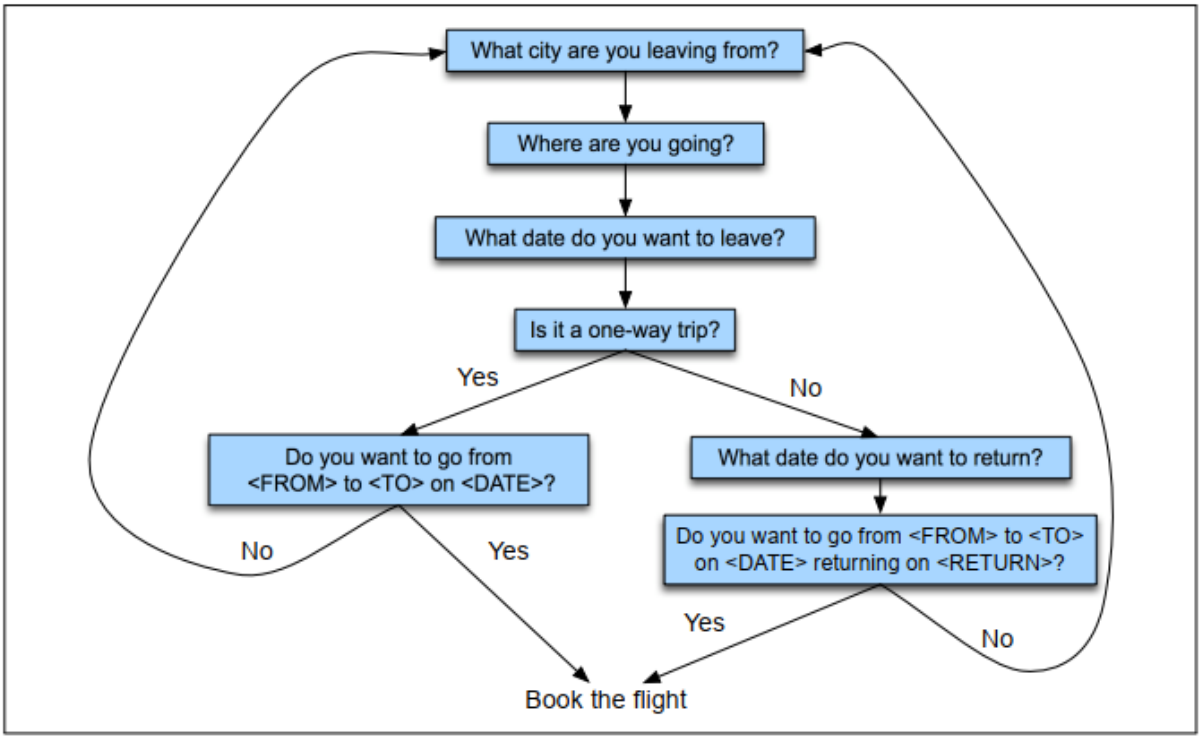
Chức năng của thành phần quản lý hội thoại là nhận đầu vào từ thành phần NLU, quản lý các trạng thái hội thoại (dialogue state), ngữ cảnh hội thoại (dialogue context), và truyền đầu ra cho thành phần sinh ngôn ngữ (Natural Language Generation - NLG) [1].

Trạng thái hội thoại (dialog state) được lưu lại và dựa vào tập luật hội thoại (dialog policy) để quyết định hành động tiếp theo cho câu trả lời của bot trong một kịch bản hội thoại, hay hành động (action) chỉ phụ thuộc vào trạng thái (dialog state) trước của nó.

### b. Các mô hình quản lý hội thoại phổ biến

Hiện nay, các chatbot thường dùng mô hình máy trạng thái hữu hạn (Finite State Machines – FSM), mô hình Frame-based (Slot Filling) hoặc kết hợp hai mô hình này [1].

- Mô hình máy trạng thái hữu hạn (Finite-State Machines): Các FSM đặt một trình tự được xác định trước các bước đại diện cho trạng thái của cuộc hội thoại tại bất cứ điểm nào. Các FSM là phương pháp thiết kế thủ công của DM. Các trạng thái của FSM tương ứng với các câu hỏi mà DM hỏi người dùng. Các cung nối giữa các trạng thái tương ứng với các hành động của chatbot sẽ thực hiện. Các hành động này phụ thuộc phản hồi của người dùng cho các câu hỏi.




**Hình 2.6: Mô hình máy trạng thái hữu hạn (Finite-State Machines) [10]**

- Mô hình Frame-based (hoặc tên khác là Form-based) có thể giải quyết vấn đề mà mô hình FSM gặp phải. Mô hình Frame-based dựa trên các frame định sẵn để định hướng cuộc hội thoại. Mỗi frame sẽ bao gồm các thông tin (slot) cần điền và các câu hỏi tương ứng mà dialogue manager hỏi người dùng. Mô hình này cho phép người dùng điền thông tin vào nhiều slot khác nhau trong frame.

| Slot | Question |
|---|---|
| ORIGIN CITY | "From what city are you leaving?" |
| DESTINATION CITY | "Where are you going?" |
| DEPARTURE TIME | "When would you like to leave?" |
| ARRIVAL TIME | "When do you want to arrive?" |

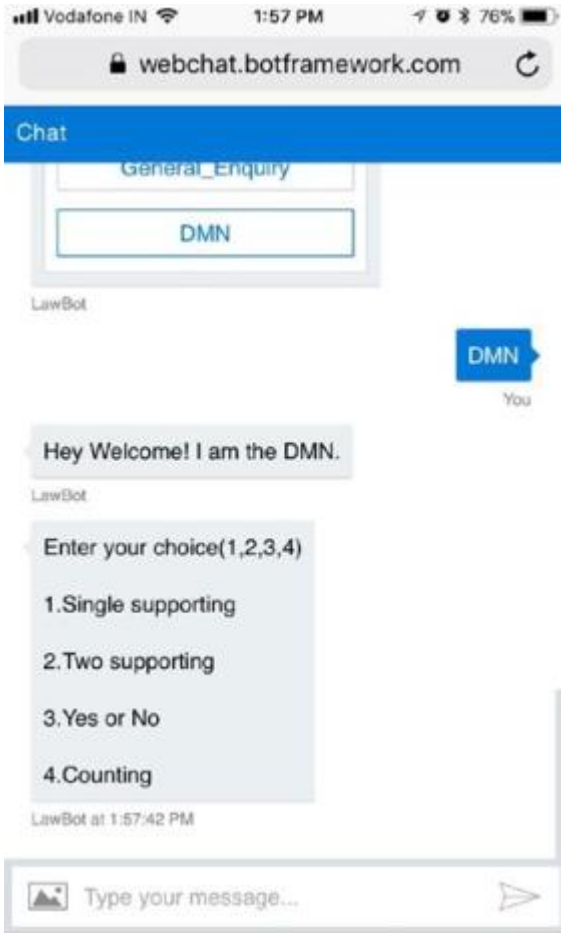
**Hình 2.7: Mô hình Frame-based [10]**

### *2.1.4 Mô hình sinh hội thoại cho chatbot*

Trên thực tế, có các loại ứng dụng NLG chính: NLG dựa trên quy tắc (rule-based), dựa trên truy xuất (retrieval-based) và dựa trên cơ sở sáng tạo (generative-based) [8].

**Các chatbot mô hình dựa trên quy tắc (rule-based)**: Đây là là kiểu kiến trúc mà hầu hết các chatbot đầu tiên đã được xây dựng. Chúng chọn câu trả lời của hệ thống dựa trên một tập hợp các quy tắc được xác định trước cố định, dựa trên việc nhận ra dạng từ vựng của văn bản đầu vào mà không tạo bất kỳ câu trả lời văn bản mới nào. Kiến thức được sử dụng trong chatbot được con người viết mã bằng tay (hard-coded) và được sắp xếp và trình bày bằng các mẫu hội thoại. Cơ sở dữ liệu quy tắc toàn diện cho phép chatbot trả lời nhiều loại đầu vào của người dùng hơn. Tuy nhiên, loại mô hình này không mạnh đối với các lỗi chính tả và ngữ pháp trong đầu vào của người dùng. Hầu hết các nghiên cứu hiện có về chatbots dựa trên quy tắc đều nghiên cứu lựa chọn phản hồi cho cuộc trò chuyện một lượt, chỉ xem xét tin nhắn cuối cùng. Trong các chatbot thông minh hơn, lựa chọn phản hồi nhiều lượt xem xét các phần trước của cuộc trò chuyện để chọn phản hồi phù hợp với toàn bộ bối cảnh hội thoại. Các hệ thống như vậy chủ yếu dựa vào các quy tắc, điều này khiến chúng kém linh hoạt hơn NLG nâng cao.

Cách tiếp cận dựa trên quy tắc, thông thường còn được gọi là cách tiếp cận theo hướng menu (menu-driven), trong đó các hành động được thực hiện bằng menu. Chatbot cố gắng hiểu câu hỏi người dùng và sau đó trình bày một menu để chọn hành động tiếp theo. Danh sách này đảm bảo backend biết hoạt động chính xác nào cần được thực hiện để thực hiện yêu cầu [12].




**Hình 2.8: Một chatbot hướng menu**

**Bảng 2.1: Ưu và nhược điểm của chatbot dựa trên quy tắc [12]**

| Ưu điểm | Nhược điểm |
|---|---|
| - Độ chính xác của phản hồi được xác nhận bởi thiết kế.<br>- Dựa trên heuristics chứ không phải NLP phức tạp, dễ hiểu và dễ thực hiện.<br>- Dễ mở rộng sang các mục menu mới mà không phải đào tạo lại core | - Chức năng được giới hạn trong các mã lệnh.<br>- Việc hoàn thành gồm hai bước: hiểu bối cảnh và đưa menu lên. Sau đó nhấp vào menu, thực hiện yêu cầu.<br>- Nó cung cấp hạn chế các cuộc trò chuyện theo ngôn ngữ tự nhiên bởi vì chatbot không hiểu các tình huống không được viết mã lệnh. |

**Mô hình dựa trên truy xuất (retrieval-based)**: Một chút khác biệt so với mô hình dựa trên quy tắc là mô hình dựa trên truy xuất (retrieval-based), cung cấp tính

linh hoạt hơn vì nó truy vấn và phân tích các tài nguyên có sẵn bằng cách sử dụng các API. Một chatbot dựa trên truy xuất lấy một số lựa chọn phản hồi từ một chỉ mục trước khi nó áp dụng phương pháp matching cho lựa chọn phản hồi.

**Mô hình sáng tạo (generative-based)**: tạo ra câu trả lời theo cách tốt hơn so các mô hình còn lại, dựa trên các tin nhắn hội thoại của người dùng hiện tại và trước đó. Các chatbot này giống con người hơn và sử dụng các thuật toán máy học (machine learning) hoặc kỹ thuật học sâu (deep learning) nên linh hoạt hơn. NLG nâng cao cho phép dự đoán khả năng xuất hiện từ này đến từ khác và sửa các lỗi ngôn ngữ, chẳng hạn như lỗi chính tả. Các thuật toán được sử dụng trong NLG nâng cao cũng tốt hơn trong việc xử lý các từ và biểu thức mới không có trong các mẫu đào tạo ban đầu.

Phương pháp dựa trên AI này dựa trên một công cụ NLP nâng cao để hỗ trợ ngôn ngữ tự nhiên và đáp ứng yêu cầu dựa trên các thuật toán ML và tích hợp hệ thống để truy xuất thông tin động. Độ chính xác của chatbot thấp hơn khi bắt đầu và tăng lên theo thời gian.

Sự khác biệt quan trọng giữa cách tiếp cận dựa trên menu và dựa trên AI là NLP engine. Engine này chịu trách nhiệm trích xuất thông tin có trong đầu vào của người dùng. Hơn nữa, dựa trên thông tin trích xuất, chatbot cần quyết định các bước tiếp theo [12].


**Hình 2.9: NLP engine trích xuất thông tin dựa trên kỹ thuật học máy**

**Bảng 2.2: Ưu và nhược điểm của chatbot dựa trên AI [12]**

| Ưu điểm | Nhược điểm |
|---|---|
| - Các cuộc trò chuyện nâng cao có thể xảy ra mà không cần người dùng thực hiện nhiều bước cho các hành động.<br>- NLP engine có thể giải quyết với các tình huống chưa biết trước và nhiều chữ.<br>- Chatbot có thể học cách tạo phản hồi tùy chỉnh từ đầu (tạo ngôn ngữ tự nhiên). | - NLP engine rất phức tạp để đào tạo, bảo trì và cải tiến.<br>- Độ chính xác của phản hồi bị ảnh hưởng bởi vì đầu ra NLP không chính xác 100%.<br>- Yêu cầu một lượng lớn dữ liệu cho một NLP engine |

## 2.2. Rasa framework

### *2.2.1. Giới thiệu*

Rasa Open source là một nền tảng học máy để tạo ra các trợ lý ảo dựa trên văn bản và giọng nói. Tính đến 8/2020, Rasa đã được download hơn 3 triệu lần, có cộng đồng diễn đàn hơn 10.000 thành viên và trên 450 người đóng góp vào mã nguồn. Các chức năng chính của Rasa:

Rasa Open Source là một nền tảng có khả năng hiểu ngôn ngữ tự nhiên, quản lý đối thoại và tích hợp. Rasa X là bộ công cụ miễn phí được sử dụng để cải thiện các trợ lý theo ngữ cảnh được xây dựng bằng Rasa Open Source. Cùng với nhau, chúng bao gồm rất nhiều các tính năng để tạo ra các trợ lý và chatbot dựa trên văn bản và giọng nói.

- Hiểu thông điệp (Understand messages): biến văn bản dạng tự do ở bất kỳ ngôn ngữ nào thành dữ liệu có cấu trúc. Hỗ trợ đơn và đa ý định (multiple intents) và cả các thực thể được đào tạo trước và tùy chỉnh (pre-trained and custom entities).

- Duy trì cuộc trò chuyện (Hold conversations): ghi nhớ ngữ cảnh bằng cách sử dụng quản lý hội thoại dựa trên máy học.

- Học tập tương tác (Interactive learning): tạo dữ liệu đào tạo bằng cách nói chuyện với chatbot của bạn và cung cấp phản hồi khi nó mắc lỗi.

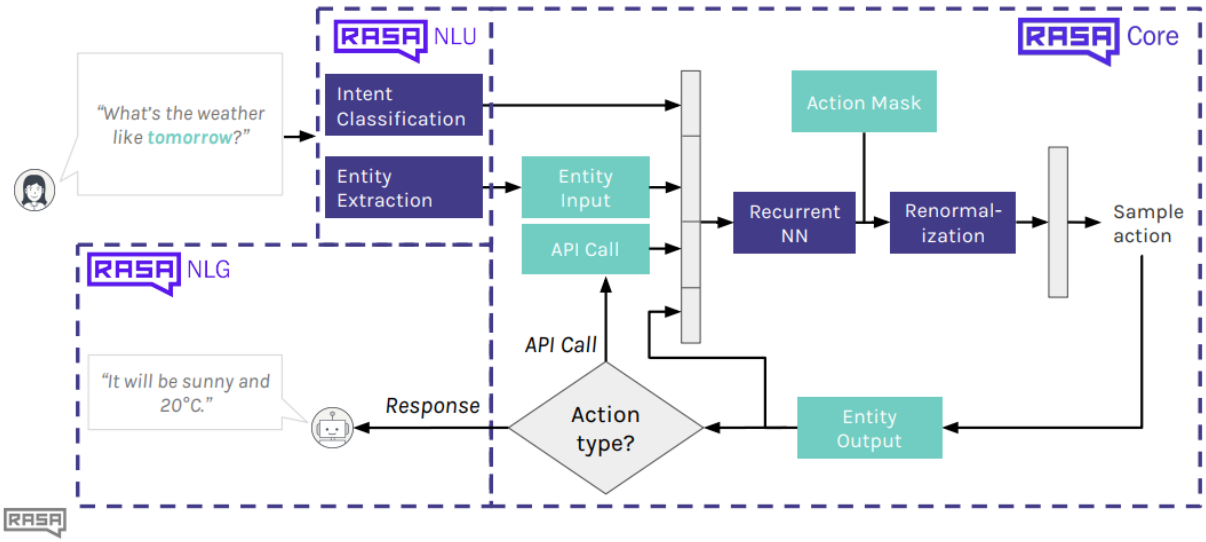
- Kết nối với các nền tảng nhắn tin thường dùng (Connect): tích hợp chatbot của bạn trên Slack, Facebook, Google Home, …

- Tích hợp các lệnh gọi API (Integrate): sử dụng các hành động tùy chỉnh của Rasa để tương tác với các API và các hệ thống khác.

- Xem và chú thích cuộc hội thoại (View and annotate conversations): lọc, gắn cờ và sửa các cuộc trò chuyện để liên tục cải thiện chatbot của bạn.

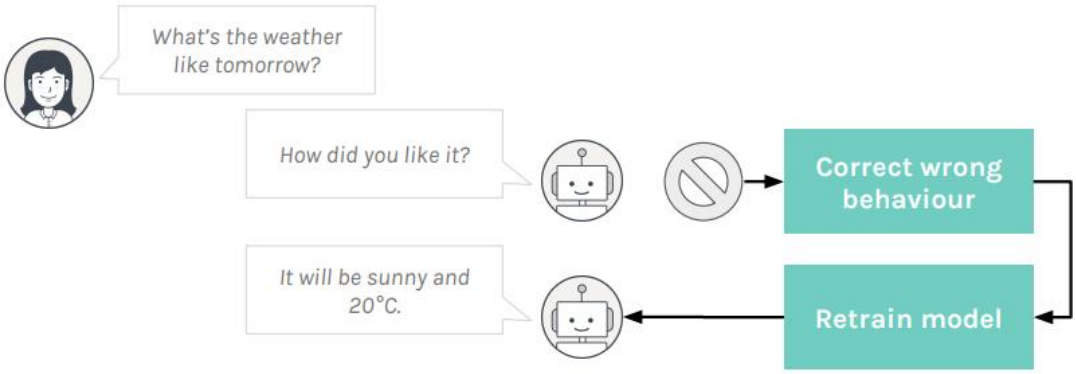
- Triển khai mọi nơi (Deploy): có khả năng triển khai Docker containers và điều phối để chạy Rasa on-premise hoặc thông qua nhà cung cấp đám mây ưa thích (cloud).

Rasa có đầy đủ các thành phần cơ bản của hệ thống chatbot bao gồm: NLU (hiểu ngôn ngữ tự nhiên), Dialogue Management (Quản lý hội thoại) và NLG (sinh ngôn ngữ tự nhiên) [4].



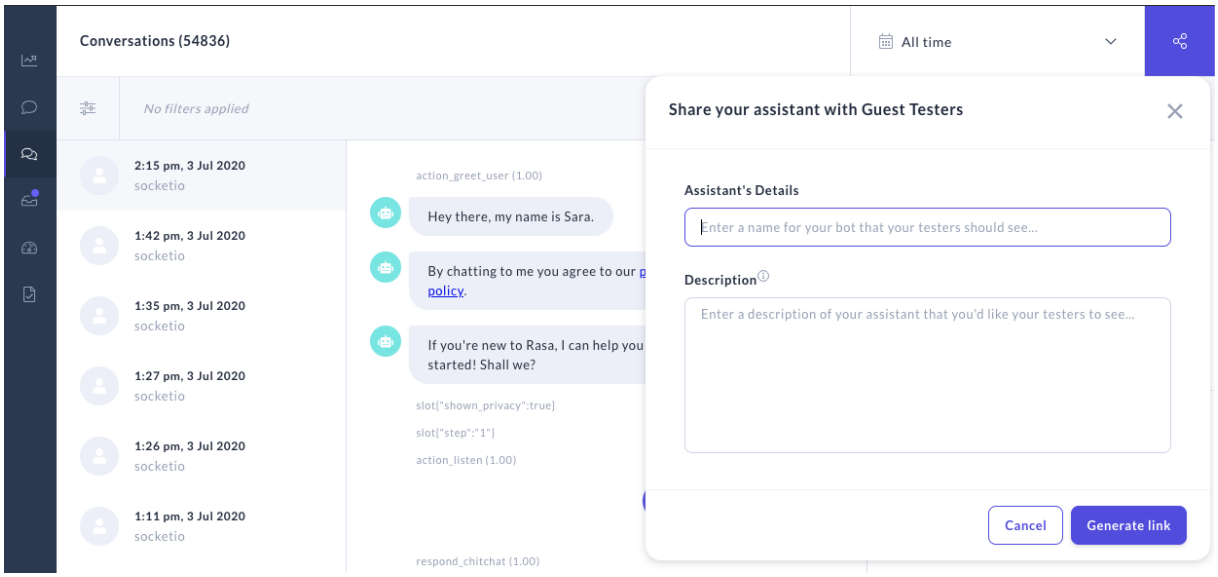

**Hình 2.10: Các thành phần của Rasa [4]**

Rasa có hỗ trợ chế độ học tương tác. Trong chế độ này, ta cung cấp phản hồi cho chatbot của mình trong khi nói chuyện với nó. Đây là một cách hiệu quả để khám phá những gì chatbot có thể làm và là cách dễ nhất để sửa chữa bất kỳ lỗi nào mà nó mắc phải. Một lợi thế của đối thoại dựa trên học máy là khi chatbot chưa biết cách làm điều gì đó, ta có thể dạy cho nó [4].



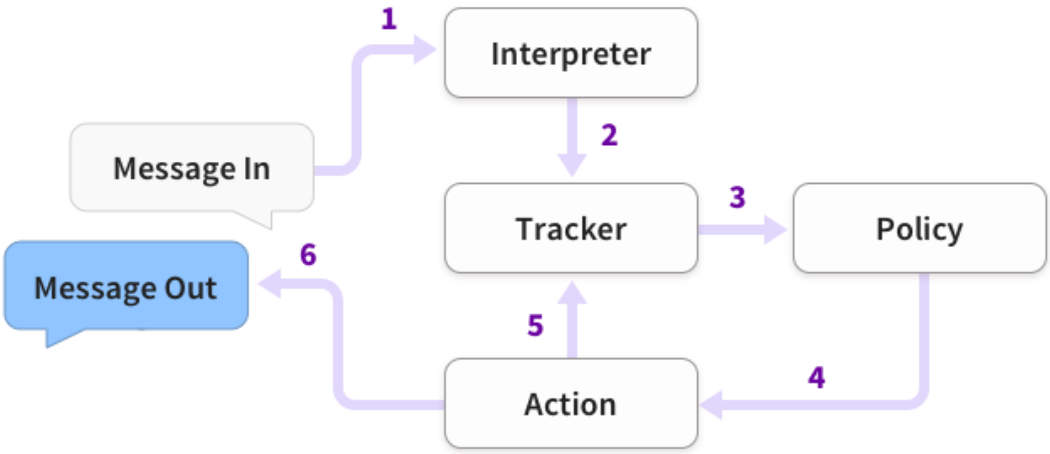

**Hình 2.11: Chế độ học tương tác của Rasa [4]**

Rasa còn cung cấp công cụ Rasa X dành cho phát triển theo hướng hội thoại (Conversation-Driven Development - CDD), đây là công cụ thông qua quá trình lắng nghe người dùng và sử dụng những thông tin chi tiết đó để cải thiện trợ lý AI chatbot.




**Hình 2.12: Công cụ Rasa X**

Sơ đồ dưới cho thấy các bước cơ bản về cách một trợ lý được xây dựng bằng Rasa phản hồi một thông báo:




**Hình 2.13: Cách thức Rasa phản hồi một tin nhắn**

- Tin nhắn được nhận và chuyển đến trình thông dịch (Interpreter), chuyển nó thành từ điển bao gồm văn bản gốc, ý định và các thực thể được tìm thấy. Phần này do NLU đảm nhận.

- Tracker là đối tượng theo dõi trạng thái hội thoại. Nó nhận được thông tin rằng một tin nhắn mới đã đến.

- Chính sách (Policy) nhận trạng thái hiện tại của Tracker.

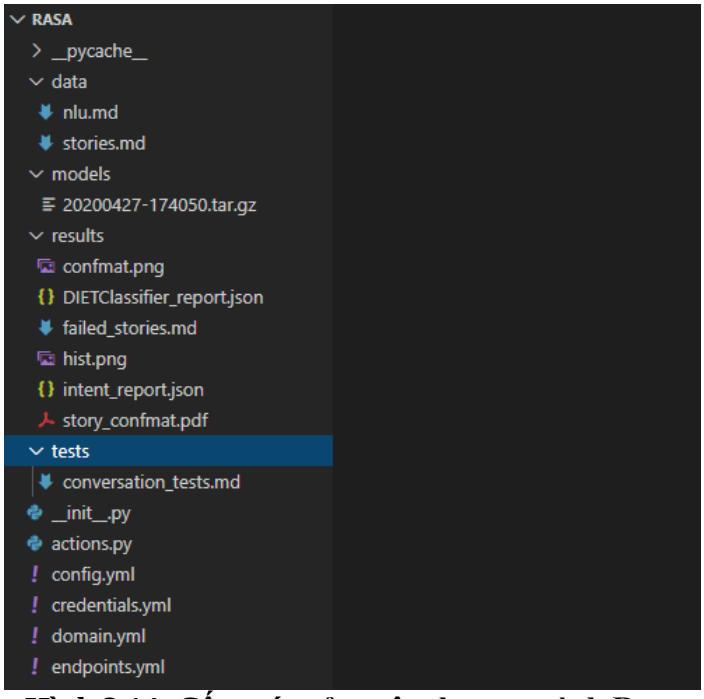
- Chính sách chọn hành động (Action) tiếp theo để thực hiện.

- Hành động đã chọn được ghi lại bởi Tracker.

- Một phản hồi được gửi đến người dùng.

### *2.2.2. Cấu trúc chương trình của Rasa*

Cấu trúc của một chương trình của Rasa như hình dưới:




**Hình 2.14: Cấu trúc của một chương trình Rasa**
*(Nguồn: https://rasa.com/docs/rasa/nlu/about/)*

Các thành phần chính trong chương trình được diễn giải như sau:

| | |
|---|---|
| __init__.py | một file trống giúp python tìm thấy hành động của chatbot |
| actions.py | mã cho các hành động tùy chỉnh của chatbot |
| config.yml '*' | cấu hình NLU và các mô hình Core của chatbot |
| credentials.yml | chi tiết để kết nối với các dịch vụ khác |
| data/nlu.md '*' | dữ liệu đào tạo NLU của chatbot |
| data/stories.md '*' | các stories |
| domain.yml '*' | miền của chatbot |
| endpoints.yml | chi tiết để kết nối với các kênh như fb messenger |
| models/<timestamp>.tar.gz | mô hình(model) ban đầu của bạn |

### 2.2.3. *Intent*

Trong Rasa, thực hiện khai báo các Intent (ý định người dùng) trong domain.yml. Chẳng hạn dưới đây liệt kê một số ý định của người dùng tương tác với chatbot như chào hỏi, tạm biệt, từ chối, đặt phòng, …

```
intents:
          - greet
          - affirm
          - deny
          - goodbye
          - book_room
          - num_rooms
          - type_rooms
          - book_number_room
          - clean_room
```

### 2.2.4. *Entity*

Trong Rasa, thực hiện khai báo các Entity (thực thể) trong domain.yml. Chẳng hạn dưới đây liệt kê một số thực thể như số, vị trí, kiểu phòng:

```
entities:
          - location
          - room_type
```

### 2.2.5. *Stories*

Rasa stories là một dạng dữ liệu đào tạo được sử dụng để đào tạo các mô hình quản lý hội thoại của Rasa.

Một story là sự trình bày cuộc trò chuyện giữa người dùng và trợ lý AI, được chuyển đổi thành một định dạng cụ thể trong đó thông tin đầu vào của người dùng được thể hiện dưới dạng ý định (intents) tương ứng (và các thực thể nếu cần) trong khi phản hồi của chatbot được thể hiện dưới dạng tên hành động (action) tương ứng.

Một ví dụ đào tạo cho hệ thống đối thoại Rasa Core được gọi là một câu chuyện. Đây là ví dụ về một đoạn hội thoại ở định dạng câu chuyện Rasa:

```
## greet + location/price + cuisine + num people    <!-- name of the
story - just for debugging -->
* greet
   - action_ask_howcanhelp
```

```
* inform{"location": "rome", "price": "cheap"}  <!-- user utterance, in
format intent{entities} -->
   - action_on_it
   - action_ask_cuisine
* inform{"cuisine": "spanish"}
   - action_ask_numpeople        <!-- action that the bot should execute
-->
* inform{"people": "six"}
   - action_ack_dosearch
```

- Một câu chuyện bắt đầu với một cái tên đứng trước hai dấu #. Ta có thể gọi câu chuyện là bất cứ thứ gì bạn thích, nhưng nó có thể rất hữu ích để gỡ lỗi khi đặt tên mô tả cho chúng.

- Kết thúc của một câu chuyện được biểu thị bằng một dòng mới và sau đó một câu chuyện mới bắt đầu lại với ##.

- Tin nhắn do người dùng gửi được hiển thị dưới dạng các dòng bắt đầu bằng * định dạng intent {"entity1": "value", "entity2": "value"}

- Các hành động được thực hiện bởi chabot được hiển thị dưới dạng các dòng bắt đầu bằng _ và chứa tên của hành động.

- Các sự kiện được trả về bởi một hành động sẽ hiển thị ngay sau hành động đó. Ví dụ, nếu một hành động trả về một sự kiện SlotSet, điều này được thể hiện như slot{"slot_name": "value"}

### *2.2.6. Actions*

Trong khi viết các story, ta sẽ gặp hai loại hành động: hành động phát biểu (utterance actions) và hành động tùy chỉnh (custom actions). Hành động phát biểu là các thông điệp được cố định (hardcoded) mà bot có thể phản hồi. Trong khi đó, các hành động tùy chỉnh liên quan đến mã tùy chỉnh (custom code) được thực thi.

Tất cả các hành động (cả hành động phát biểu và hành động tùy chỉnh) được thực hiện bởi chatbot được hiển thị dưới dạng các dòng bắt đầu – và theo sau là tên của hành động.

Các phản hồi cho các hành động phát biểu phải bắt đầu bằng tiền tố utter_và phải khớp với tên của phản hồi được xác định trong domain.yml.

```
responses:
  utter_greet:
```

```
  - text: "Hey! How are you?"
```

Đối với hành động tùy chỉnh, tên hành động là chuỗi ta chọn để trả về từ phương thức name của lớp (class) hành động tùy chỉnh. Mặc dù không có hạn chế nào về việc đặt tên cho các hành động tùy chỉnh (không giống như các hành động phát biểu), nhưng cách tốt nhất ở đây là đặt tiền tố tên bằng action_.

```
action_endpoint:
  url: "http://localhost:5055/webhook"
```

### *2.2.7. Policies*

Trong file config.yml có khóa policies mà ta có thể sử dụng để tùy chỉnh các chính sách mà chatbot của mình sử dụng. Ví dụ, trong ví dụ bên dưới, có tham số max_history để kiểm soát lượng lịch sử đối thoại mà mô hình xem xét để quyết định hành động nào cần thực hiện tiếp theo.

```
policies:
  - name: "KerasPolicy"
    featurizer:
    - name: MaxHistoryTrackerFeaturizer
      max_history: 5
      state_featurizer:
        - name: BinarySingleStateFeaturizer
  - name: "MemoizationPolicy"
    max_history: 5
  - name: "FallbackPolicy"
    nlu_threshold: 0.4
    core_threshold: 0.3
    fallback_action_name: "my_fallback_action"
  - name: "path.to.your.policy.class"
    arg1: "..."
```

### *2.2.8. Slots*

Slots là bộ nhớ của chatbot. Chúng hoạt động như một kho lưu trữ khóa-giá trị (key-value) có thể được sử dụng để lưu trữ thông tin mà người dùng cung cấp (ví dụ: thành phố của họ) cũng như thông tin thu thập được về thế giới bên ngoài (ví dụ: kết quả của một truy vấn cơ sở dữ liệu).

Ví dụ: nếu người dùng đã cung cấp thành phố của họ, ta có thể có một slot dạng text được gọi là home_city. Nếu người dùng yêu cầu thời tiết và chatbot không

biết thành phố của họ, nó sẽ phải hỏi họ. Slot chỉ cho Rasa Core biết liệu slot đó có giá trị hay không. Giá trị cụ thể của một slot (ví dụ: Bangalore hoặc New York hoặc Hồng Kông) không tạo ra bất kỳ sự khác biệt nào.

## 2.3. Kết luận chương

Trong chương này luận văn đã giới thiệu các kỹ thuật quan trọng nhất được sử dụng trong chatbot, ngoài ra cũng đề cập đến các thành phần cơ bản của Rasa framework. Đây là các cơ sở để áp dụng xây dựng bài toán chatbot trả lời thông tin khách sạn.

# CHƯƠNG 3: XÂY DỰNG CÔNG CỤ HỎI ĐÁP THÔNG TIN KHÁCH SẠN

## 3.1. Giới thiệu bài toán

Trong chương này tác giả lựa chọn bài toán trả lời thông tin khách sạn nhằm cung cấp thông tin, tư vấn và hỗ trợ bán hàng cho khách sạn, với khả năng hoạt động liên tục 24/7, hỗ trợ con người trong việc trả lời các câu hỏi liên quan đến khách sạn. Trên cơ sở nghiên cứu các câu hỏi thường gặp [13] [14], bài toán tập trung vào một số chức năng chính của chatbot như sau:

- Chào hỏi
- Tạm biệt
- Thông tin về các loại phòng khách sạn
- Thông tin liên quan về wifi trong khách sạn
- Thông tin hỏi về đưa vật nuôi vào khách sạn
- Thông tin về bữa ăn sáng
- Thông tin về thời gian check-in
- Thông tin về thời gian check-out
- Thông tin về khu vực hút thuốc trong khách sạn
- Thông tin về két sắt trong khách sạn
- Thông tin về số điện thoại hotline
- Thông tin về nhà hàng trong khách sạn
- Thông tin về thời gian mở cửa nhà hàng trong khách sạn
- Thông tin về món chay trong nhà hàng
- Thông tin về bể bơi trong khách sạn
- Thông tin về phòng tập gym trong khách sạn
- Thông tin về chỗ đậu xe trong khách sạn
- Các câu hỏi FAQ khác
- Thông tin chung của khách sạn
- Đặt phòng

### *3.1.1. Mô hình huấn luyện cho chatbot*

Trong Rasa, các messages được xử lý bởi một chuỗi các thành phần (components). Các thành phần này được thực thi lần lượt trong "pipeline" được xác định trong file config.yml. Việc lựa chọn một NLU pipeline cho phép ta tùy chỉnh mô hình của mình và kết hợp nó trên tập dữ liệu của mình [20].

Có các Components để trích xuất thực thể, để phân loại ý định, lựa chọn phản hồi, tiền xử lý và các thành phần khác. Nếu muốn thêm thành phần của riêng mình, chẳng hạn như để chạy kiểm tra chính tả (spell-check) hoặc để phân tích quan điểm (sentiment analysis), có thể thực hiện custom component.

Một pipeline thường bao gồm ba phần chính:

**- Tokenization:**

Tách mỗi câu thành một danh sách các từ tố (token), mỗi câu được tách ra thành một danh sách các từ có nghĩa. Đối với ngôn ngữ tiếng Việt, các từ được phân tách bằng dấu cách , ở đây tác giả lựa chọn tách từ bằng WhitespaceTokenizer. Ngoài ra tác giả có thực hiện custom tokenization cho tiếng Việt với thư viện xử lý tiếng Việt của tác giả Vũ Anh [21].

Nhằm chuẩn hóa từ đồng nghĩa bằng việc đồng nhất từ đồng nghĩa, từ địa phương, tiếng lóng về một từ chuẩn hóa, tác giả sử dụng EntitySynonymMapper.

**- Featurization:**

Ta cần quyết định xem có nên sử dụng các thành phần cung cấp tính năng nhúng từ được đào tạo trước (pre-trained word embeddings) hay nhúng được giám sát (Supervising Embeddings).

+ Pre-trained Embeddings: 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 hay biểu diễn ngôn ngữ dưới dạng vector. Lợi thế của việc sử dụng tính năng nhúng từ được đào tạo trước là nếu ta có một ví dụ đào tạo như: "Tôi muốn mua táo" và Rasa được yêu cầu dự đoán ý định cho "lấy lê", thì mô hình

của ta đã biết rằng từ "táo" và "lê" rất giống nhau. Điều này đặc biệt hữu ích nếu không có đủ dữ liệu đào tạo…

+ Supervised Embeddings: Với phương pháp nhúng được giám sát này thì ta sẽ tự tạo tập dữ liệu training riêng của mình từ đầu. Với phương pháp nhúng được giám sát, Rasa có khả năng huấn luyện với bất kì ngôn ngữ nào (bao gồm tiếng Việt), vì sẽ training lại mọi thứ từ đầu, chỉ phụ thuộc vào dữ liệu huấn luyện. Với việc khó tìm ra được mô hình đào tạo trước cho ngôn ngữ tiếng Việt, cùng với bài toán trong một miền lĩnh vực đóng như trả lời thông tin tin khách sạn thì nó sẽ đảm bảo tính chính xác hơn nhiều và tránh dư thừa dữ liệu. Do đó, ở đây tác giả lựa chọn phương pháp này cùng với một số thành phần cụ thể là:

CountVectorsFeaturizer: trích xuất đặc trưng cho phân loại ý định và lựa chọn phản hồi, tạo túi từ (BoW:bag-of-words) đại diện cho tin nhắn người dùng, ý định và phản hồi.

RegexFeaturizer: Tạo biểu diễn vectơ của thông điệp người dùng bằng cách sử dụng biểu thức chính quy(regular expressions)

LexicalSyntacticFeaturizer: Tạo các đặc trưng từ vựng và cú pháp cho tin nhắn của người dùng để hỗ trợ trích xuất thực thể.

**- Entity Recognition / Intent Classification / Response Selectors:**

Tùy thuộc vào dữ liệu, ta có thể chỉ muốn thực hiện phân loại ý định, nhận dạng thực thể hoặc lựa chọn phản hồi. Hoặc ta có thể muốn kết hợp nhiều nhiệm vụ đó. Rasa hỗ trợ một số thành phần cho mỗi nhiệm vụ. Trong Rasa, việc nhận diện ý định thông thường sử dụng mô hình máy vector hỗ trợ (Support Vector Machines-SVM), trích xuất thông tin thực thể sử dụng mô hình trường ngẫu nhiên có điều kiện (Conditional Random Fields - CRF). Ở đây tác giả lựa chọn các thành phần như sau:

DIETClassifier: DIET (Dual Intent và Entity Transformer) là một kiến trúc đa tác vụ để phân loại ý định và nhận dạng thực thể. Kiến trúc dựa trên một bộ chuyển đổi được chia sẻ cho cả hai nhiệm vụ. Một chuỗi các nhãn thực thể được dự đoán thông qua một lớp gắn thẻ trường ngẫu nhiên có điều kiện (Conditional Random Field - CRF) tương ứng với chuỗi đầu vào của tokens. Đối với nhãn ý định, đầu ra bộ

chuyển đổi cho __CLS__ token và nhãn ý định được nhúng vào một không gian vectơ ngữ nghĩa duy nhất. DIETClassifier cũng hỗ trợ đa ý định (multi-intent) tách các Intent thành nhiều nhãn.

ResponseSelector: Thành phần này có thể được sử dụng để xây dựng mô hình truy xuất phản hồi nhằm dự đoán trực tiếp phản hồi của bot từ một tập hợp các phản hồi. Dự đoán của mô hình này được sử dụng bởi Retrieval Actions. Nó nhúng đầu vào của người dùng và nhãn phản hồi vào cùng một không gian và tuân theo cùng một kiến trúc và tối ưu hóa mạng thần kinh giống hệt như DIETClassifier.

Sau khi xây dựng xong mô hình và tạo một số dữ liệu đào tạo NLU, có thể huấn luyện (train) mô hình với rasa train nlu. Sau khi quá trình huấn luyện kết thúc, ta có thể kiểm tra (test) khả năng của mô hình diễn giải các thông điệp đầu vào khác nhau qua: rasa shell nlu.

### *3.1.2. Đánh giá hiệu quả của chatbot*

Dưới đây là một số phương pháp đánh giá các mô hình phân loại của Rasa.

**+ Accuracy:**

Cách đánh giá này đơn giản tính tỉ lệ các lớp đã phân loại đúng / tổng số dự đoán.

**+ Confusion matrix:**

Là một phương pháp đánh giá kết quả của những bài toán phân loại với việc xem xét cả những chỉ số về độ chính xác và độ bao quát của các dự đoán cho từng lớp. Confusion matrix là một ma trận tổng quát thể hiện kết quả phân loại chính xác và quả phân loại sai được tạo ra bởi một mô hình phân loại. Đây là một ma trận vuông với kích thước các chiều bằng số lượng lớp dữ liệu. Giá trị tại hàng thứ i, cột thứ j là số lượng điểm lẽ ra thuộc vào class i nhưng lại được dự đoán là thuộc vào class j.

**+ True/False Positive/Negative:**

Xét một confusion matrix gồm 4 chỉ số sau đối với mỗi lớp phân loại:

**Bảng 3.1: Bảng confusion matrix**

| | | **Dự đoán (predicted)** | |
|---|---|---|---|
| | | True | False |
| **Thực tế (Actual)** | Positive | True Positive | False Negative |
| | Negative | False Positive | True Negative |

- TP (true positive) – mẫu mang nhãn dương được phân lớp đúng vào lớp dương.
- FN (false negative) – mẫu mang nhãn dương bị phân lớp sai vào lớp âm.
- FP (false positive) – mẫu mang nhãn âm bị phân lớp sai vào lớp dương.
- TN (true negative) – mẫu mang nhãn âm được phân lớp đúng vào lớp âm.

**+ Precision / Recall:**

- Precision là tỉ lệ số điểm Positive mô hình dự đoán đúng (TP) trên tổng số điểm mô hình dự đoán là Positive (TP+FP), theo công thức:

$$\text{Precision} = \frac{TP}{TP+FP}$$

0< Precision <=1, Precision càng lớn có nghĩa là độ chính xác của các điểm tìm được càng cao.

Precision = 1, tức là tất cả số điểm mô hình dự doán là Positive đều đúng, hay không có điểm nào có nhãn là Negative mà mô hình dự đoán nhầm là Positive.

- Recall là tỉ lệ số điểm Positive mô hình dự đoán đúng (TP) trên tổng số điểm thật sự là Positive (hay tổng số điểm được gán nhãn là Positive ban đầu TP+FN), chỉ số này được tính theo công thức:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Recall càng cao, tức là số điểm là positive bị bỏ sót càng ít. Recall = 1, tức là tất cả số điểm có nhãn là Positive đều được mô hình nhận ra.

**+ F1- Score:**

Chỉ sử dụng Precision hay chỉ có Recall thì không đánh giá được chất lượng mô hình.

Khi đó F1-score được sử dụng. F1-score là trung bình điều hòa của precision và recall (giả sử hai đại lượng này khác 0). F1-score được tính theo công thức:

$$F1 = \frac{2*Precision*Recall}{Precesion+Recall}$$

F1-Score có giá trị nằm trong khoảng (0, 1], F1-Score càng cao thì mô hình phân loại càng tốt.

## 3.2. Xây dựng Chương trình

### *3.2.1. Nguồn dữ liệu xây dựng*

Nguồn dữ liệu thực nghiệm, tác giả đã thu thập chính từ bộ câu hỏi, câu trả lời, là những câu hỏi thường gặp của một số khách sạn [13] [14], ngoài ra có tham khảo thêm một số chatbot facebook fanpage.

Để làm giàu cho tập dữ liệu huấn luyện, tác giả bổ sung thêm các câu hỏi mới trong mỗi Intent, đảm bảo các Intent quan trọng mang tính hỏi đáp nhiều có ít nhất 10 câu hỏi, các intent khác cũng có tối thiểu 5 câu hỏi, mỗi câu hỏi là một các cách diễn đạt khác nhau với cùng mục đích với câu hỏi (ban đầu). Các câu hỏi này đã được gán nhãn cùng với Intent của câu hỏi ban đầu.

### *3.2.2. Xây dựng ý định*

Nhiệm vụ xây dựng các tập ý định sẽ theo nguyên tắc là những mẫu câu hỏi mà người dùng khi có ý định đó hay sử dụng nhất có thể. Cần định nghĩa các ý định khớp với ngôn ngữ tự nhiên nhất, trong hệ thống này tác giả định nghĩa các ý định như sau với nguồn dữ liệu thực nghiệm.

**Bảng 3.2: Bảng các ý định (intent) của chabot**

| **Tên ý định** | **Mô tả** | **Số lượng câu hỏi** | **Số lượng câu trả lời** |
|---|---|---|---|
| greet | Chào hỏi | 10 | 1 |
| goodbye | Tạm biệt | 10 | 1 |

| | | | |
|---|---|---|---|
| Deluxe_room_details | Ý định liên quan về loại phòng Deluxe | 5 | 1 |
| premium_room_details | Ý định liên quan về loại phòng Premium | 5 | 1 |
| suite_room_details | Ý định liên quan về loại phòng Suite | 5 | 1 |
| family_room_details | Ý định hỏi về đưa vật nuôi vào khách sạn | 5 | 1 |
| book_room_now | Ý định đặt phòng | 10 | 1 |
| faq_breakfast | Ý định hỏi về bữa sáng | 10 | 2 |
| faq_breakfast_time | Ý định hỏi về thời gian bữa sáng | 10 | 1 |
| faq_cancellation_policy | Ý định hỏi về chính sách hủy phòng | 10 | 1 |
| faq_check_in_time | Ý định hỏi về thời gian check-in | 10 | 2 |
| faq_check_out_time | Ý định hỏi về thời gian check-out | 10 | 2 |
| faq_check_out_late | Ý định hỏi về thời gian check-out muộn | 10 | 2 |
| faq_check_in_soon | Ý định hỏi về thời gian check-in sớm | 10 | 2 |
| faq_reception_time | Ý định hỏi về thời gian lễ tân hoạt động | 10 | 1 |
| faq_pet_policy | Ý định hỏi về chính sách vật nuôi | 10 | 2 |
| faq_nearest_airport | Ý định hỏi về sân bay gần nhất | 10 | 1 |
| faq_meeting_room | Ý định hỏi về phòng họp | 10 | 1 |
| faq_luggage | Ý định hỏi về trông giữ đồ | 10 | 1 |
| faq_contact_number | Ý định hỏi về số hotline | 10 | 1 |
| faq_smoking_room | Ý định hỏi về khu vực hút thuốc | 10 | 1 |
| faq_safety_box | Ý định hỏi về khách sạn có két sắt không | 10 | 1 |
| ask_hotline | Ý định hỏi về số điện thoại hotline | 10 | 1 |

| | | | |
|---|---|---|---|
| faq_restaurant | Ý định hỏi về khách sạn có nhà hàng không | 10 | 1 |
| faq_restaurant_time | Ý định hỏi về thời gian mở cửa nhà hàng | 10 | 2 |
| faq_restaurant_vegetarian | Ý định hỏi về có món chay trong nhà hàng không | 10 | 1 |
| faq_massage | Ý định hỏi về dịch vụ massage | 10 | 1 |
| faq_delegation_policy | Ý định hỏi về chính sách cho khách đoàn | 10 | 2 |
| faq_wifi | Ý định hỏi về Internet trong khách sạn | 10 | 1 |
| faq_parking | Ý định hỏi về chỗ đậu xe trong khách sạn | 10 | 1 |
| faq_parking_cost | Ý định hỏi về giá đậu xe trong khách sạn | 10 | 1 |
| faq_baby_cot | Ý định hỏi về có cũi cho trẻ em | 10 | 1 |
| faq_baby_care | Ý định hỏi về dịch vụ trông trẻ | 10 | 1 |
| faq_baby_play_area | Ý định hỏi về chỗ chơi cho trẻ em | 10 | 1 |
| faq_baby_chair | Ý định hỏi về ghế cho trẻ em | 10 | 1 |
| faq_baby_age | Ý định hỏi về quy định tuổi trẻ em | 10 | 1 |
| faq_baby_food | Ý định hỏi về đồ ăn cho trẻ em | 10 | 1 |
| faq | Ý định hỏi thêm các câu faq | 10 | 1 |
| hotel_info | Ý định hỏi về thông tin của khách sạn | 10 | 1 |
| rooms | Ý định hỏi về các loại phòng | 10 | 1 |
| faq_people_rooms | Ý định hỏi về số người ở được trong phòng | 10 | 1 |
| faq_bank | Ý định hỏi về ngân hàng gần nhất | 10 | 1 |
| faq_hospital | Ý định hỏi về bệnh viện gần nhất | 10 | 1 |
| faq_healthcare | Ý định hỏi về dịch vụ bác sỹ trong khách sạn | 10 | 1 |
| faq_transport | Ý định hỏi về thông tin di chuyển đến khách sạn | 10 | 2 |

| | | | |
|---|---|---|---|
| faq_swimming_pool | Ý định hỏi về bể bơi trong khách sạn | 10 | 1 |
| faq_deposit | Ý định hỏi về ký quỹ/đặt cọc khi đặt phòng | 10 | 2 |
| faq_gym | Ý định hỏi về thông tin phòng tập gym | 10 | 1 |

Ví dụ cho một ý định muốn hỏi về thông tin thời gian check-in như sau:

```
## intent:faq_check_in_time
- thời gian nhận phòng như thế nào?
- bắt đầu nhận phòng từ mấy giờ được vậy bạn?
- có thể lấy phòng từ mấy giờ?
- thời gian nhận phòng lúc nào?
- thời gian nhận phòng?
- bạn có thể cho tôi biết thời gian nhận phòng không?
- cho biết thời gian nhận phòng của bạn
- khi nào tôi có thể nhận phòng?
- bạn có thể cho tôi biết khi nào nhận phòng không?
- khi nào nhận phòng?
```

### *3.2.3. Xây dựng thực thể*

Entities là các thực thể thông tin đặc trưng quan trọng được trích xuất theo các ý định người dùng. Slots là các thông tin được trích chọn trong câu nói của người dùng được hệ thống lưu lại trong bộ nhớ hệ thông để dùng trong các hành động hoặc để đưa ra các câu trở lời phù hợp theo ngữ cảnh, tránh việc phải hỏi lại những thông tin mà người dùng đã cung cấp từ trước.

```
entities:
- location
- room_type
slots:
  adults:
    type: unfeaturized
    auto_fill: false
  check_in:
    type: unfeaturized
    auto_fill: false
  check_out:
    type: unfeaturized
    auto_fill: false
  child:
    type: unfeaturized
    auto_fill: false
```

```
  email:
    type: unfeaturized
    auto_fill: false
  name:
    type: unfeaturized
    auto_fill: false
  number:
    type: unfeaturized
  phno:
    type: unfeaturized
    auto_fill: false
  room:
    type: unfeaturized
    auto_fill: false
  room_type:
    type: unfeaturized
```

Bài toán xây dựng 2 thực thể: vị trí (location) và loại phòng (room_type), mỗi loại thực thể được xây dựng 20 câu training như dưới.

```
- [nhà hàng] (location) mở cửa đến mấy giờ vậy em?
- [nhà hàng] (location) bắt đầu lúc nào vậy?
- [nhà hàng] (location) phục vụ từ mấy giờ đến mấy giờ?
- thời gian [nhà hàng] (location) mở cửa là gì?
- thời gian của [nhà hàng] (location)
- thời gian mở cửa [nhà hàng] (location)
- thời gian [nhà hàng] (location)
- [nhà hàng] (location) mở cửa cho đến khi nào?
- [nhà hàng] (location) sẽ mở cửa đến mấy giờ?
- bạn có thể chia sẻ thời gian của [nhà hàng] (location) không?
- [nhà hàng] (location) mở cửa lúc nào
- [nhà hàng] (location) mở cửa đến mấy giờ
```

### *3.2.4. Xây dựng câu trả lời*

Khi người dùng đưa ra các câu hỏi, yêu cầu thì Chatbot phải có nhiệm vụ đưa ra được câu trả lời đáp ứng được các câu hỏi và yêu cầu đó.

Để tạo tính tự nhiên và phù hợp với độ tuổi trong cuộc hội thoại thì ta có thể xây dựng nhiều tập mẫu câu trả lời để Chatbot lựa phù hợp với lứa tuổi, giới tính khi có được thông tin về khách hàng hoặc nếu không thì sẽ chọn ngẫu nhiên các mẫu câu trả lời để tạo cảm giác không bị nhàm chán.

Có thể xây dựng các phản hồi cho Chatbot thông qua action. Khi có yêu cầu đầu vào của người dùng thì Chatbot có thể lựa chọn các hành động phù hợp để đáp

ứng nhu cầu đó. Hành động này có thể cung cấp thông tin mong muốn cho người dùng dựa vào các ý định, slot và dữ liệu lấy từ hệ thống cơ sở dữ liệu thông qua các API kết nối. Bên cạnh đó action của rasa còn hỗ trợ tùy biến qua ngôn ngữ python nên ta có thể điều hướng các action tiếp theo dựa vào dialog state tracker, policy và dispatcher của rasa. Có các loại hành động trong Rasa Core:

**- Utterance Actions:**

Để xác định một hành động phát biểu (ActionUtterTemplate), ta thêm một phản hồi của chatbot vào domain.yml bắt đầu bằng utter_:

```
utter_faq_parking:
  - text: Bãi đậu xe của khách sạn có sức chứa đủ cho 20 xe hơi với 2 khu vự
c đậuxe phía trước và phía sau sau khách sạn. Khách sạn cũng có chỗ đậu xe g
ắn máy tại tầng hầm. Số lượng phục vụ có thể lên đến 35 chiếc.
utter_faq_parking_cost:
  - text: Phí đậu xe đã bao gồm trong giá phòng.
```

Thông thường, bắt đầu tên của một hành động phát biểu bằng utter_. Nếu tiền tố này bị thiếu, ta vẫn có thể sử dụng phản hồi (response) trong các hành động tùy chỉnh của mình (custom actions), nhưng phản hồi không thể dự đoán trực tiếp như hành động của chính nó.

**- Default Actions:**

Các hành động default như lắng nghe người dùng, restart lại hội thoại hoặc trả lời mặc định khi không phân loại được ý định người dùng.

```
  utter_default:
  - text: Xin lỗi tôi không hiểu. Tôi có thể hỗ trợ bạn các chức năng như th
ông tinkhách sạn, các loại phòng, đặt phòng, thông tin các dịch vụ ẩm thực t
rong kháchsạn, thông tin các dịch vụ cung cấp khác như truyền hình, internet
, massage...
```

**- Custom actions:**

Khi các tập câu trả lời mẫu không áp dụng được với các câu trả lời cần có kết quả lấy từ một nguồn dữ liệu khác thì action tùy biến được sử dụng, nó sẽ trỏ đến một hàm trong lớp action (python). Trong đây sẽ tùy biến câu trả lời như lấy dữ liệu qua API rồi điền vào tham số trong câu trả lời. Ở đây thực hiện việc tương tác đặt phòng với chatbot qua API bằng cách xây dựng custom actions trong actions.py.

```
class BookRoomForm(FormAction):
 def name(self):
  return "book_room_form"

 def required_slots(self,tracker) -> List[Text]:
  return ["room_type","check_in","check_out","adults","child","room","name",
"phno","email"]
 def slot_mappings(self) -> Dict[Text, Union[Dict, List[Dict]]]:
    return {
            "room_type": [
                self.from_text(),
            ],
            "check_in": [
                self.from_text(),
            ],
            "check_out": [
                self.from_text(),
            ],

            "adults": [
                self.from_text(),
            ],
            "child": [
                self.from_text(),
            ],
            "room": [
                self.from_text(),
            ],
            "name": [
                self.from_text(),
            ],
            "phno": [
                self.from_text(),
            ],
            "email": [
                self.from_text(),
            ],
        }
```

### *3.2.5. Xây dựng khung kịch bản*

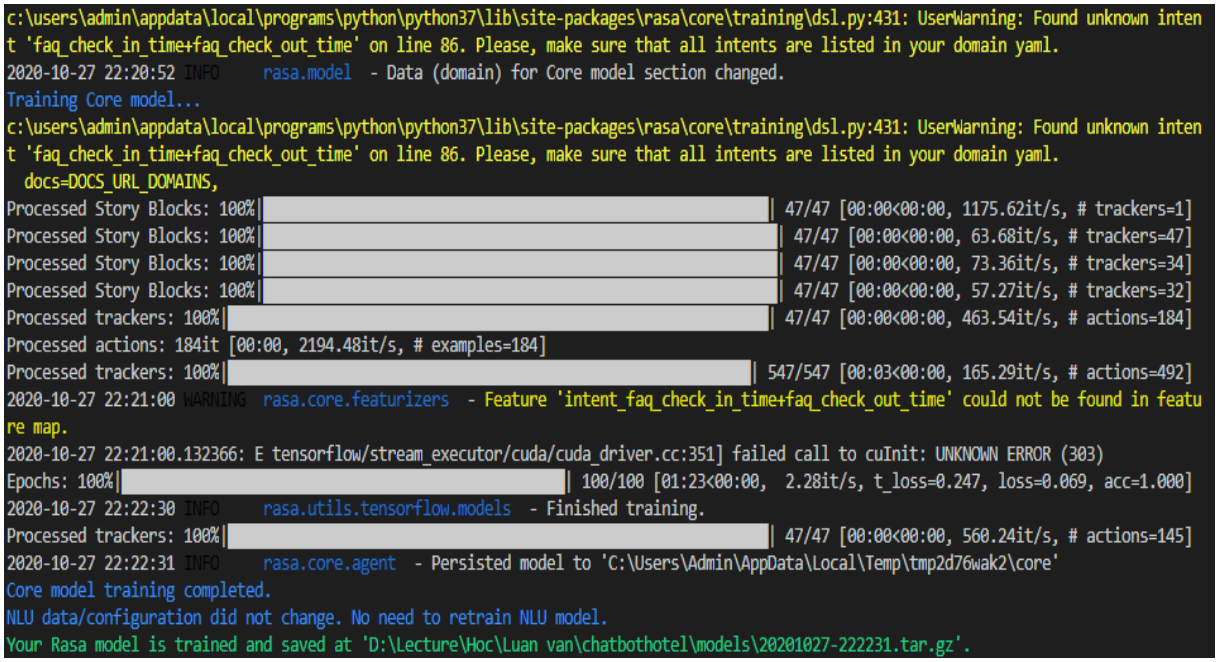
Với mỗi một ý định của người dùng thì tương ứng với một tập các mẫu câu trả lời đã được xây dựng sẵn, ta xây dựng các khung kịch bản cho Chatbot dựa trên việc sắp xếp thành đoạn đối thoại.

```
## hi
* greet
 - utter_greet

## bye
* goodbye
 - utter_goodbye

## deluxe_room_details
* greet
 - utter_greet
* rooms
 - utter_rooms
* deluxe_room_details
 - utter_deluxe_details

## premium_room_details
* greet
 - utter_greet
* rooms
 - utter_rooms
* premium_room_details
 - utter_premium_details

## suite_room_details
* greet
 - utter_greet
* rooms
 - utter_rooms
* suite_room_details
 - utter_suite_details

## family_room_details
* greet
 - utter_greet
* rooms
 - utter_rooms
* family_room_details
 - utter_family_details

## book_room_now2
* greet
 - utter_greet
* book_room_now2
 - book_room_form2
 - action_book_rooms_details2
```

```
## faq_breakfast
* greet
 - utter_greet
* faq
 - utter_faq_prompt
* faq_breakfast
 - utter_faq_breakfast

## faq_cancellation_policy
* greet
 - utter_greet
* faq
 - utter_faq_prompt
* faq_cancellation_policy
 - utter_faq_cancellation_policy
```

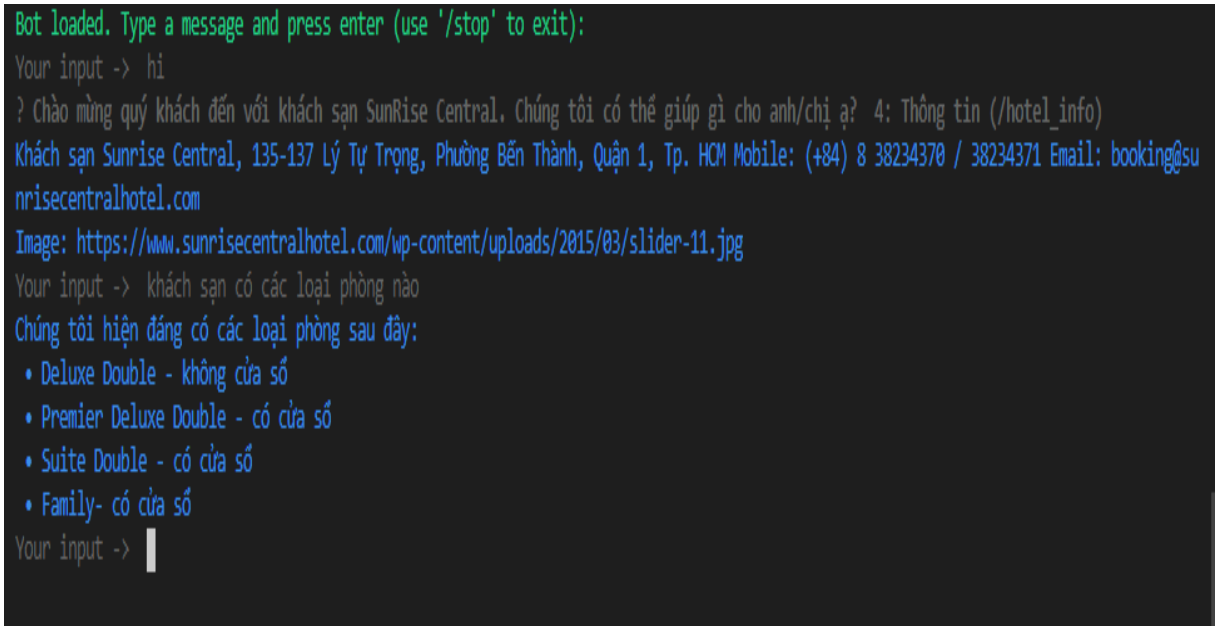
### *3.2.6. Đào tạo cho chatbot*

Tiến hành train cho một model sử dụng NLU data và các kịch bản (stories), mô hình được đào tạo sẽ được lưu dưới dạng thư mục /models.



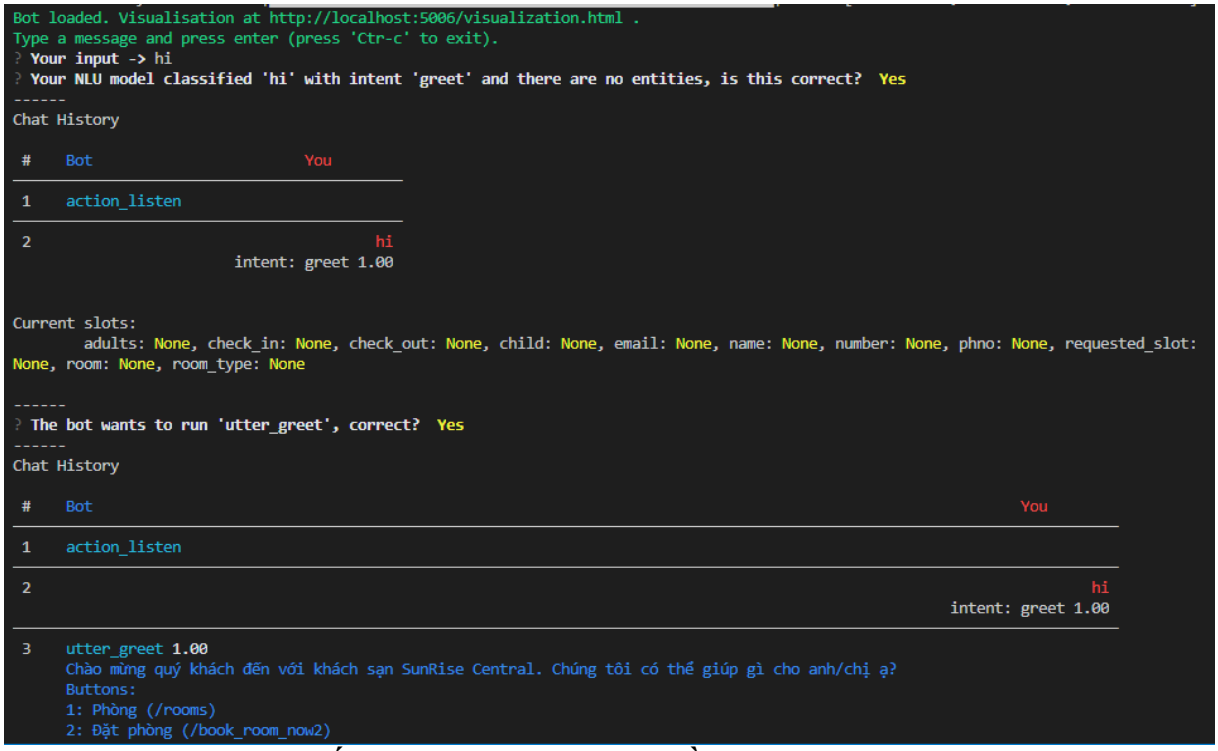

**Hình 3.1: Đào tạo một model**

Việc xây dựng đoạn hội thoại này có thể viết bằng tay. Thông qua cửa sổ dòng lệnh người dùng có thể sử dụng để đào tạo cho chatbot.




**Hình 3.2: Đào tạo cho chatbot dạng shell**

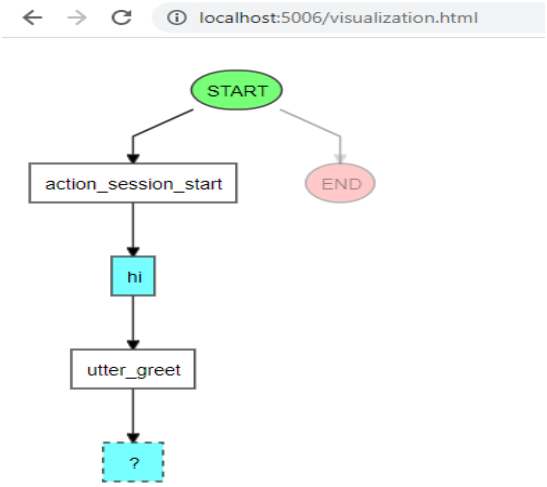
Việc xây dựng đoạn hội thoại này có thể viết bằng tay hoặc thông qua việc học tương tác (Interactive Learning) với Chatbot: Đây là một cách khác để xây dựng khung câu truyện là việc học tương tác với bot. Chế độ này cho phép người dùng tự động tạo ra các hội thoại sau khi chat trực tiếp với bot. Nếu bot nhận định các intent hay slot sai thì người dùng có huấn luyện lại cho bot đúng.




**Hình 3.3: Chế độ đào tạo cho chatbot bằng Interactive Learning**

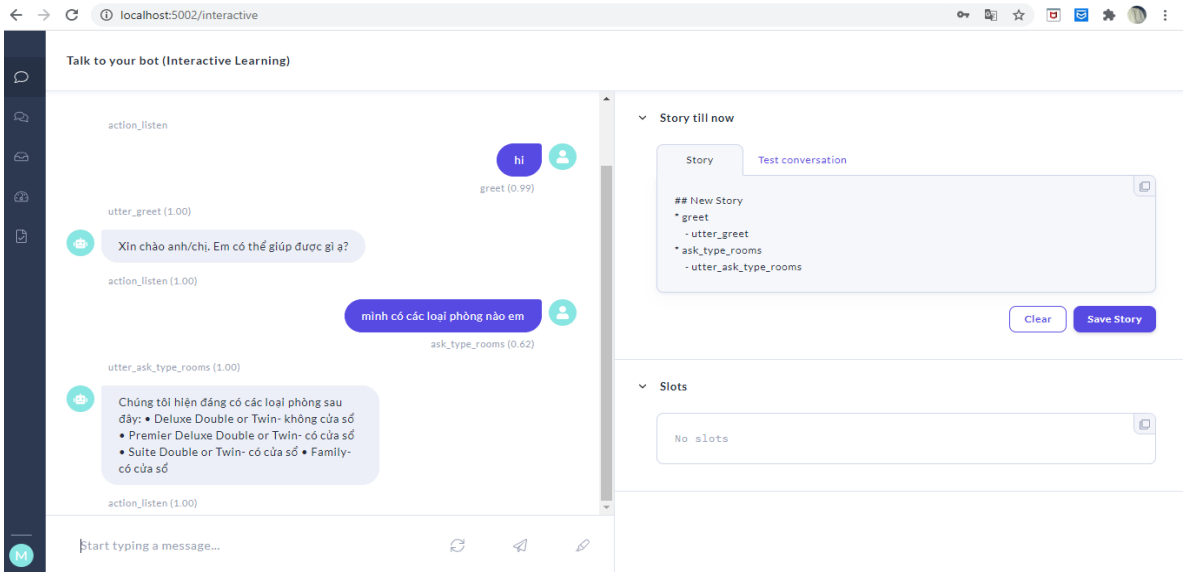
Trong ví dụ trên đây, khi người dùng gõ vào từ "hi", NLU sẽ phân loại ý định đây là dạng "greet" (chào hỏi), và hỏi xem có đúng không (Yes/No). Xác định ý định hợp lý thì chúng ta xác nhận đúng (Yes) để tiếp tục giúp đào tạo cho chatbot đưa ra câu phản hồi cho người dùng tương ứng (utter_greet).

Trong quá trình học tập tương tác, Rasa sẽ vẽ sơ đồ cuộc hội thoại hiện tại và một vài cuộc hội thoại tương tự từ dữ liệu đào tạo. Ta có thể xem trực quan hóa cuộc hội thoai ngay sau khi bắt đầu học tương tác.




**Hình 3.4: Trực quan hóa cuộc hội thoại**

Ngoài ra, ta cũng có thể sử dụng Rasa X để thực hiện chế độ học tập tương tác với chatbot. Đây là công cụ rất thuận tiện để đào tạo, kiểm tra chatbot.

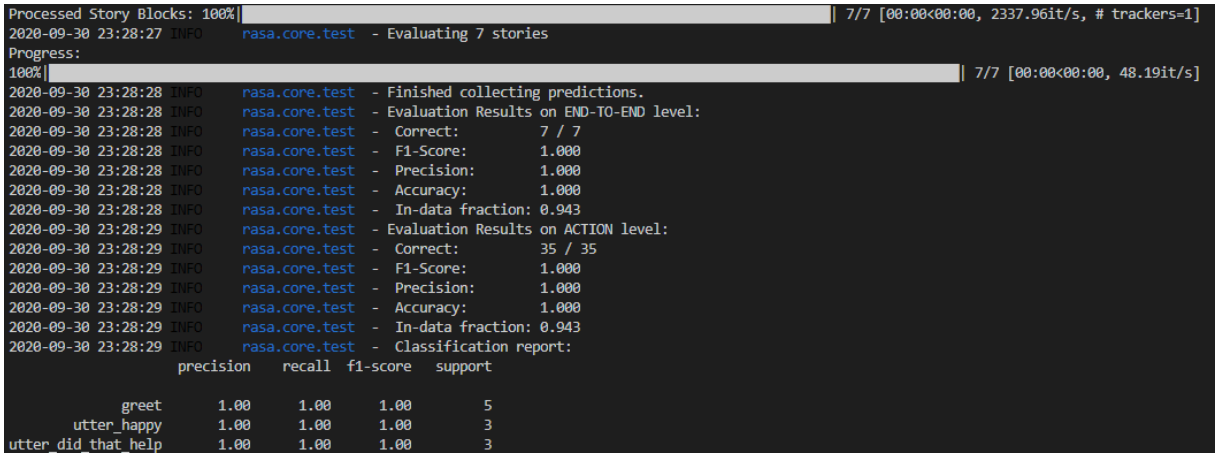



**Hình 3.5: Học tương tác qua Rasa X**

### *3.2.7. Kiểm tra chatbot*

Rasa Open Source cho phép ta kiểm tra các cuộc hội thoại từ đầu đến cuối bằng cách chạy các cuộc hội thoại kiểm tra để đảm bảo rằng cả NLU và Core đều đưa ra dự đoán chính xác.

Để làm điều này, ta cần một số stories ở định dạng end-to-end, bao gồm cả đầu ra của NLU và văn bản gốc. Theo mặc định, Rasa lưu các bài hội thoại kiểm tra vào tests/conversation_tests.md. Ta có thể kiểm tra chatbot của mình bằng cách chạy lệnh: $ rasa test




**Hình 3.6: End-to-end testing với Rasa**

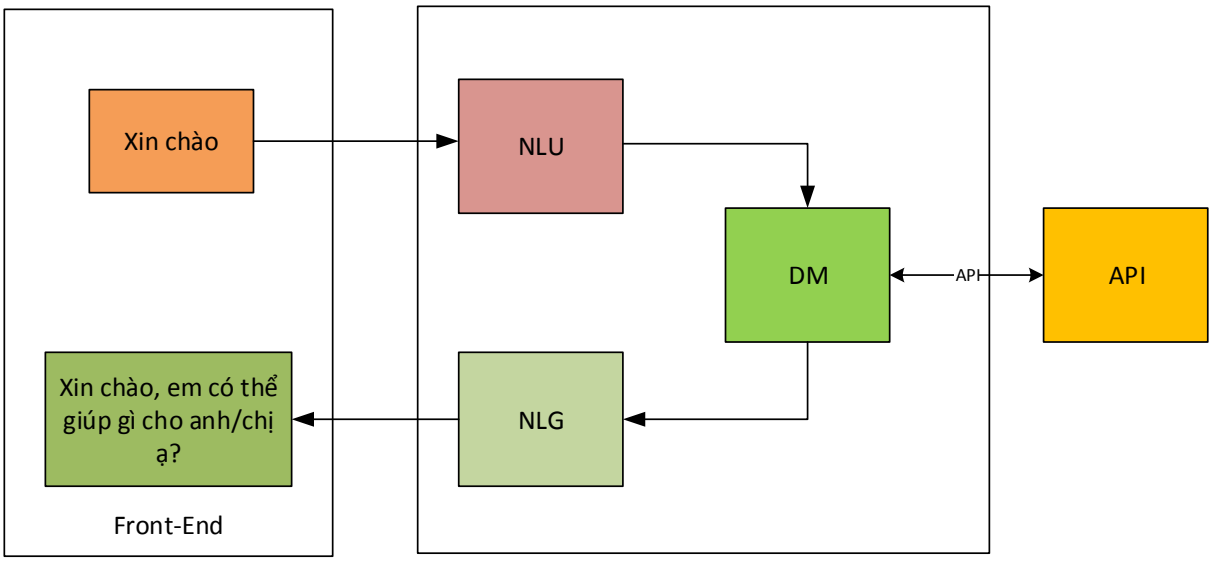
## 3.3. Kết quả thực nghiệm

### *3.3.1. Môi trường thực nghiệm*

Chương trình thử nghiệm được thiết kế, xây dựng và thực hiện trên môi trường hệ điều hành Windows với nền tảng framework Rasa 1.10.12, dựa trên ngôn ngữ lập trình python 3.7. Giao điện người dùng sử dụng nền tảng web/ứng dụng chat Telegram.

### *3.3.2. Thiết kế*

Hình dưới đây minh hoạ kiến trúc chung của bài toán.




**Hình 3.7: Kiến trúc chung của hệ thống**

Front-end: sử dụng giao diện web hoặc các trình nhắn tin phổ biến (Facebook Messenger/Telegram…). Với mục tiêu minh họa, ở đây sử dụng một giao diện web và chat Telegram.

- Mỗi khi có một người dùng gửi tin nhắn cho chatbot thì nội dung tin này sẽ gửi một POST request đến webhook được sử dụng để lắng nghe sự kiện. Webhoook này sẽ chuyển tiếp đến bộ NLU của RASA.

- RASA nhận diện ý định. Sau khi đã thu được message của người dùng thì sử dụng RASA để hiểu được ý định của người dùng cùng các thông tin thực thể.

- Thông tin này tiếp tục được chuyển đến DM của Rasa, tại đây tùy theo ý định và thông tin thực thể cùng với các thông tin theo dõi của cuộc trò chuyện đã xảy ra cho đến nay, để dự đoán một phản ứng thích hợp, bao gồm cả việc gọi API để lấy thông tin trả lời thích hợp.

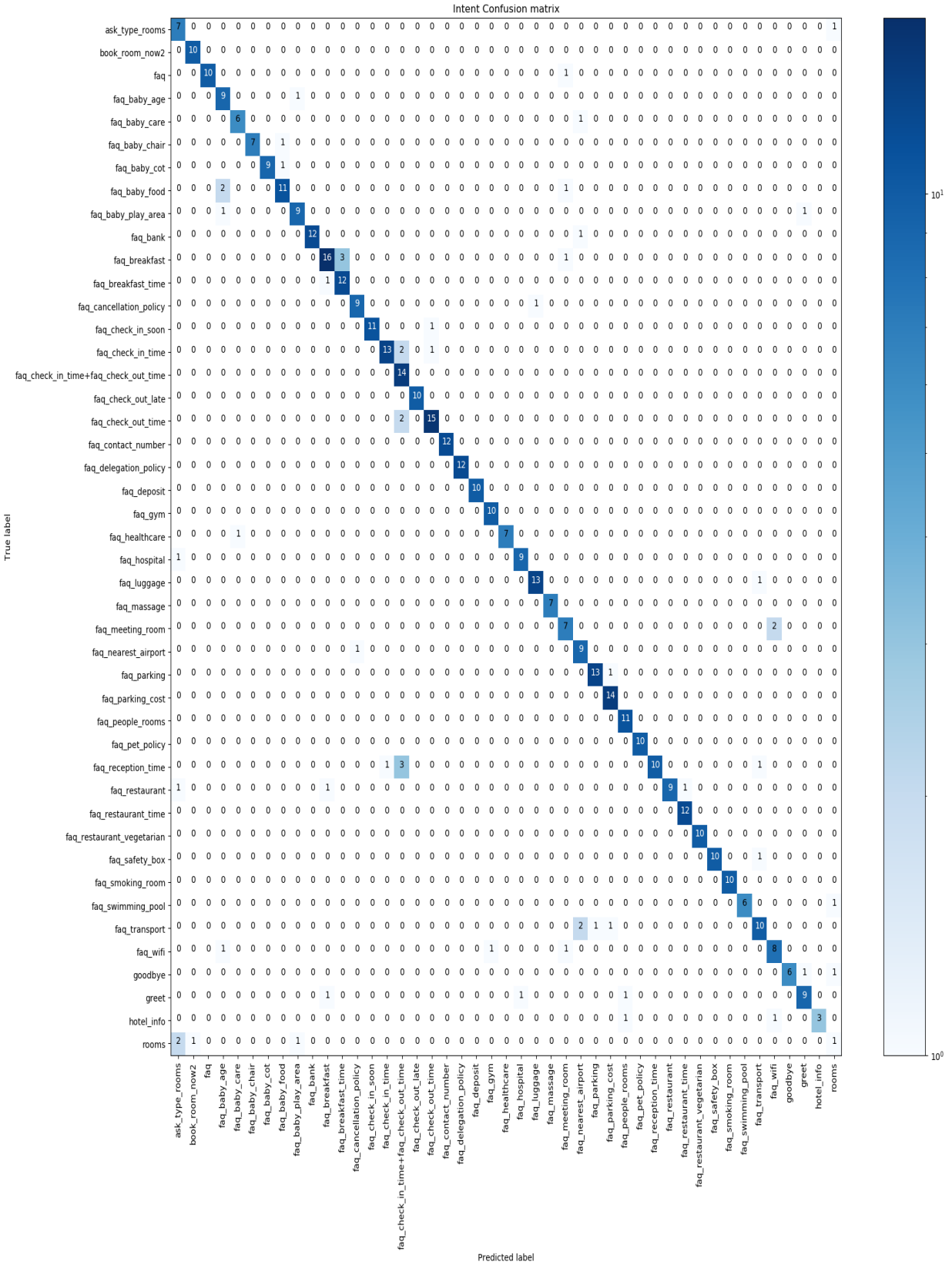
- NLG sinh ra câu trả lời dựa vào dữ liệu từ thành phần DM theo các mẫu câu template đã được xây dựng trước hoặc là kết quả của API.

- Gửi tin nhắn qua phản hồi trả về cho người dùng.

### *3.3.3. Kết quả thực nghiệm*

a) Kết quả thử nghiệm

Kết quả đánh giá NLU model và Rasa Core sau khi thực hiện đào tạo chatbot và kiểm tra trên dữ liệu test có 814 câu dữ liệu người dùng nhập vào:




**Hình 3.8: Intent Confusion matrix**

**Bảng 3.3: đánh giá trích chọn thông tin thực thể (entity)**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| room_type | 0.85 | 0.75 | 0.79 | 8 |
| location | 1.00 | 0.58 | 0.74 | 17 |

**Bảng 3.4: đánh giá mô hình Rasa Core**

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| utter_faq_restaurant_time | 1.00 | 0.64 | 0.78 | 11 |
| utter_faq_safety_box | 1.00 | 0.75 | 0.86 | 4 |
| utter_faq_bank | 1.00 | 1.00 | 1.00 | 8 |
| utter_rooms | 0.80 | 0.80 | 0.80 | 5 |
| utter_faq_contact_number | 1.00 | 1.00 | 1.00 | 10 |
| utter_faq_deposit | 1.00 | 0.67 | 0.80 | 9 |
| utter_ask_type_rooms | 0.86 | 1.00 | 0.92 | 6 |
| utter_faq_hospital | 1.00 | 1.00 | 1.00 | 9 |
| utter_faq_baby_food | 1.00 | 1.00 | 1.00 | 14 |
| utter_faq_check_out_late | 1.00 | 0.78 | 0.88 | 9 |
| utter_faq_transport | 1.00 | 0.83 | 0.91 | 6 |
| utter_faq_restaurant | 1.00 | 0.75 | 0.86 | 8 |
| utter_faq_reception_time | 0.70 | 1.00 | 0.82 | 14 |
| utter_faq_pet_policy | 1.00 | 1.00 | 1.00 | 8 |
| utter_deluxe_details | 1.00 | 1.00 | 1.00 | 2 |
| utter_faq_check_in_time | 0.71 | 0.94 | 0.81 | 16 |
| utter_faq_massage | 1.00 | 1.00 | 1.00 | 13 |
| utter_faq_healthcare | 1.00 | 1.00 | 1.00 | 5 |
| utter_hotel_info | 0.92 | 1.00 | 0.96 | 11 |
| utter_faq_luggage | 1.00 | 0.90 | 0.95 | 10 |
| utter_faq_smoking_room | 1.00 | 1.00 | 1.00 | 8 |
| book_room_form2 | 1.00 | 0.95 | 0.97 | 37 |
| utter_greet | 1.00 | 1.00 | 1.00 | 7 |
| utter_faq_baby_age | 1.00 | 0.91 | 0.95 | 11 |
| utter_faq_nearest_airport | 1.00 | 1.00 | 1.00 | 5 |
| utter_faq_meeting_room | 1.00 | 0.88 | 0.93 | 8 |

| | | | | |
|---|---|---|---|---|
| utter_faq_baby_chair | 1.00 | 1.00 | 1.00 | 8 |
| utter_faq_people_rooms | 1.00 | 1.00 | 1.00 | 4 |
| utter_faq_baby_play_area | 0.89 | 1.00 | 0.94 | 8 |
| action_listen | 0.98 | 1.00 | 0.99 | 401 |
| utter_faq_delegation_policy | 1.00 | 1.00 | 1.00 | 8 |
| utter_faq_breakfast_time | 0.78 | 0.64 | 0.70 | 11 |
| utter_faq_check_in_soon | 1.00 | 0.57 | 0.73 | 14 |
| utter_faq_swimming_pool | 1.00 | 1.00 | 1.00 | 5 |
| utter_faq_cancellation_policy | 1.00 | 1.00 | 1.00 | 9 |
| utter_suite_details | 1.00 | 1.00 | 1.00 | 2 |
| utter_faq_baby_cot | 1.00 | 1.00 | 1.00 | 8 |
| utter_family_details | 1.00 | 0.50 | 0.67 | 8 |
| utter_faq_parking | 1.00 | 0.80 | 0.89 | 10 |
| utter_faq_check_out_time | 0.62 | 0.59 | 0.61 | 17 |
| utter_faq_parking_cost | 0.73 | 1.00 | 0.85 | 11 |
| utter_faq_breakfast | 0.89 | 1.00 | 0.94 | 8 |
| utter_faq_restaurant_vegetarian | 1.00 | 1.00 | 1.00 | 7 |

Tính chung, kết quả test trên tập dữ liệu test end-to-end cho độ chính xác khoảng 89%.

Correct: 731 / 814

F1-Score: 0.906

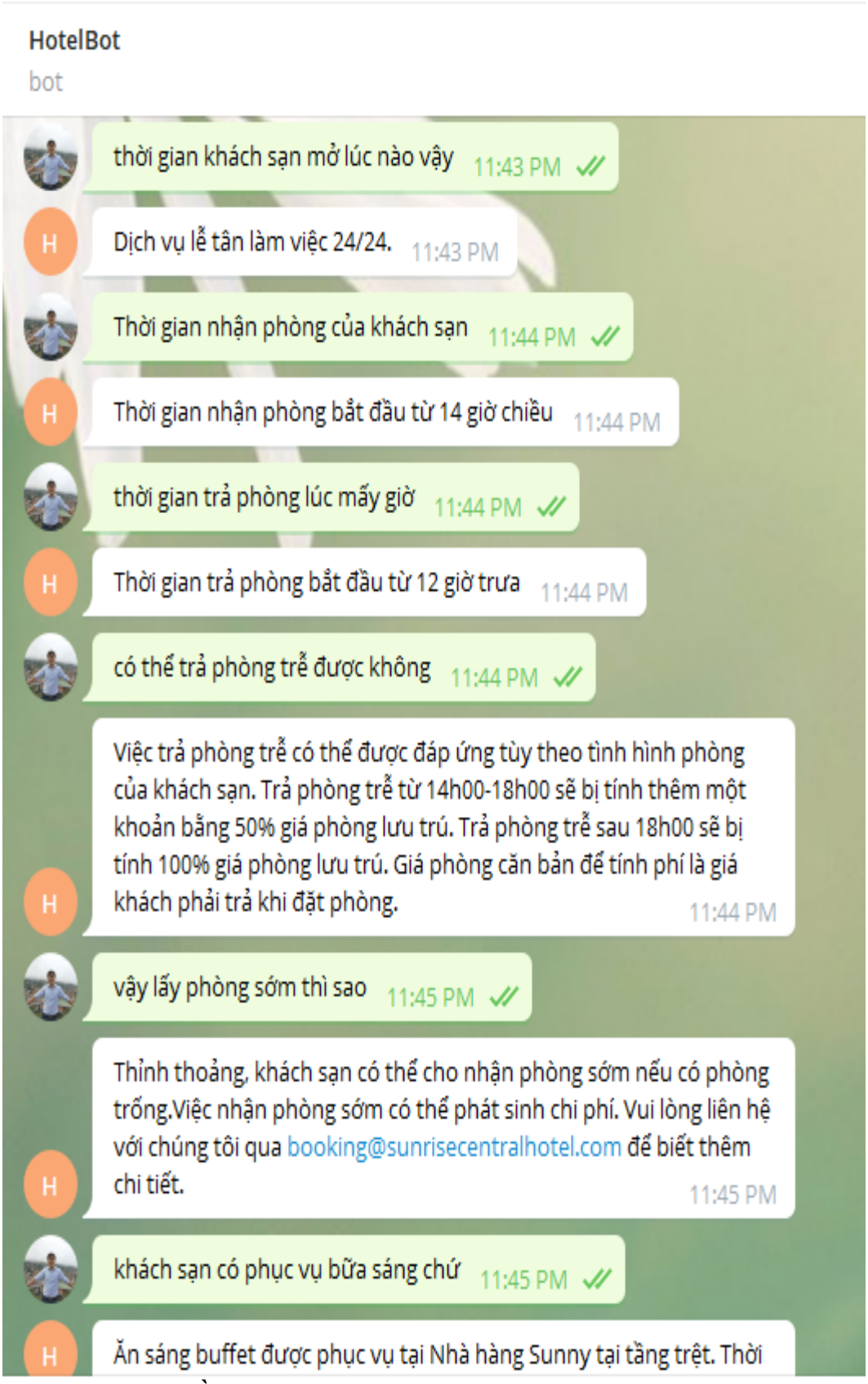
Precision: 0.926

Accuracy: 0.899

b) Test trên giao diện người dùng cuối

Thực hiện thử nghiệm tương tác với Chatbot với một số câu câu hỏi gần với kịch bản đã đào tạo cho Chatbot.


**Hình 3.9: Hỏi về các loại phòng của khách sạn**




**Hình 3.10: Hỏi về các thông tin thời gian check-in, check-out của khách sạn**


**Hình 3.11: Hỏi về các thông tin dạng FAQ khác của khách sạn**


HotelBot
bot
Đặt phòng 11:54 PM
Bạn muốn đặt phòng nào? 11:54 PM
Phòng Delu...
Phòng Pre...
Phòng Suite
Phòng Family
Bạn muốn nhận phòng (check-in) vào ngày nào? Ví dụ: 20/11/2020 11:55 PM
18/10 11:55 PM
Xin vui lòng điền đúng định dạng ngày: dd/yy/yyyy (Ngày/Tháng/Năm) lớn hơn ngày hiện tại 11:55 PM
Bạn muốn nhận phòng (check-in) vào ngày nào? Ví dụ: 20/11/2020 11:55 PM
19/10/2020 11:55 PM
Bạn muốn trả phòng (check-out) vào ngày nào? Ví dụ: 21/11/2020 11:55 PM
20/10/2020 11:55 PM
Có bao nhiêu người lớn (trên 12 tuổi) ? 11:56 PM
1
2
3
Có bao nhiêu trẻ em (dưới 12 tuổi)? 11:56 PM
0
1
2


**Hình 3.12: Đặt phòng**

## 3.4. Đánh giá

Từ kết quả thực nghiệm rút ra một số đánh giá như sau:

- Xác định đúng được ý định (intent) có ý nghĩa quan trọng nhất đối với chatbot. Đối với bài toán trong miền đóng cần xác định rõ ràng các intent, xây dựng tập dữ liệu đủ lớn, gán nhãn và tiến hành training.

- Xây dựng dữ liệu đào tạo, training cho chatbot với các kịch bản là rất cần thiết để cho độ chính xác cao của chatbot.

- Chatbot ứng dụng AI có khả năng đáp ứng tốt với các kịch bản dựng sẵn, và được đào tạo. Đối với các kịch bản nằm ngoài kịch bản dựng sẵn, có thể tăng cường khả năng cho chatbot bằng cách điều hướng người dùng về các câu mặc định hoặc các dạng giao diện menu lựa chọn.

- Việc xác định và phản hồi đa ý định có thể thực hiện bằng việc kết hợp các ý định.

- Qua bài toán thực nghiệm có thể thấy rằng áp dụng bài toán Chatbot cho việc hỗ trợ trả lời thông tin khách sạn là khả thi, có tính thực tiễn cao, và hoàn toàn áp dụng được ngay trong thực tiễn.

## 3.5. Kết luận chương

Trong chương này luận văn đã mô tả các bước chính trong xây dựng chương trình thực nghiệm từ việc lựa chọn nguồn dữ liệu, xây dựng ý định, thực thể và các kịch bản trả lời. Kết quả của việc huấn luyện và test dữ liệu cho kết quả khá cao trên tập dữ liệu huấn luyện và test. Huấn luyện càng nhiều dữ liệu thì độ chính xác sẽ càng cao hơn.

# KẾT LUẬN

Đề tài thực hiện nghiên cứu một số kiến thức bao gồm kiến trúc và nhiệm vụ các thành phần chatbot, một số thuật toán cơ bản áp dụng vào việc xây dựng chatbot để giải quyết các bài toán theo hướng tiếp cận miền đóng, cụ thể là lĩnh vực khách sạn du lịch. Dựa vào đó ta có thể áp dụng xây dựng chatbot giải quyết các bài toán hỗ trợ người dùng trong nhiều lĩnh vực thực tế.

**Các vấn đề mà luận văn đã đạt được:**

- Nghiên cứu và tìm hiểu, trình bày một cách khái quát nhất về hệ thống chatbot, các thành phần, kỹ thuật được áp dụng trong chatbot.

- Tìm hiểu lựa chọn mô hình chatbot phù hợp, từ đó có thể quyết định xây dựng bot theo mô hình nào, phương pháp nào phù hợp hơn cho từng yêu cầu bài toán. Cụ thể với bài toán trả lời thông tin khách sạn sử dụng phương pháp tiếp cận dạng chat văn bản (text-based), trong miền đóng (closed-domain), hướng mục tiêu (task-oriented), thiết kế dựa trên AI.

- Khảo sát, nghiên cứu, xây dựng chatbot thực nghiệm xử lý ngôn ngữ tự nhiên (NLP) bằng tiếng Việt sử dụng Rasa framework. Thử nghiệm và đánh giá mô hình RASA trên bộ dữ liệu xây dựng cho kết quả độ chính xác khá cao (khoảng 89%). Chatbot hỗ trợ tương đối nhiều ý định (hơn 50 intents) và cũng đã có kịch bản minh họa hỗ trợ đa ý định (multi-intent).

- Cài đặt và triển khai ứng dụng trên môi trường hệ điều hành Windows với nền tảng Rasa framework 1.10.12, ngôn ngữ lập trình python 3.7, sử dụng chat client dạng web hoặc telegram. Sản phẩm demo có được sẽ làm tiền đề cho việc phát triển, hoàn thiện sản phẩm trong thời gian tới. Sản phẩm cũng cho thấy khả năng xây dựng chatbot ứng dụng AI trong các lĩnh vực khác là hoàn toàn khả thi.

**Một số hạn chế:**

- Chưa xây dựng được đa dạng hóa câu trả lời

- Chatbot mới chỉ hỗ trợ dạng hội thoại text, chưa hỗ trợ voice.

- Chưa có khả năng trả lời các câu hỏi phức tạp.

- Chưa xây dựng được giao diện quản trị để tự tổ chức xây dựng intent, entity, kịch bản… nhằm mục đích cung cấp cho nhiều khách sạn khác nhau.

**Định hướng nghiên cứu tiếp theo:**

- Xây dựng thêm đa dạng hóa các câu trả lời ngẫu nhiên theo các ý định
- Tích hợp speech to text và text to speech cho chatbot để hỗ trợ voice.
- Xây dựng bot có khả năng trả lời các câu hỏi phức tạp hơn.
- Xây dựng giao diện quản trị tự tổ chức và quản lý intent, entity, kịch bản… để hỗ trợ cho các khách sạn tự xây dựng chatbot của riêng mình mà không cần hiểu nhiều về lập trình.

# TÀI LIỆU THAM KHẢO

[1] Phạm Quang Nhật Minh (2017). Các bài toán xử lý ngôn ngữ tự nhiên trong phát triển hệ thống chatbot. Viện nghiên cứu công nghệ FPT (FTRI).

[2] Nguyễn Thành Thủy (2018). Ứng dụng thuật toán học có giám sát multi-class SVM trong xây dựng hệ thống chatbot hỏi đáp tiếng Việt. The 7th conference on information technology and its applications, 177-184.

[3] Yun-Nung (Vivian) Chen, Asli Celikyilmaz and Dilek Hakkani-Tur (2018). Deep Learning for Dialogue Systems. Association for Computational Linguistics.

[4] Tom Bocklisch (2018). Conversational AI with Rasa NLU & Rasa Core.

[5] Denis Rothman. Artificial Intelligence By Example (2018). Develop machine intelligence from scratch using real artificial intelligence use cases. Packt Publishing.

[6] Sumit Raj (2019). Building Chatbots with Python Using Natural Language Processing and Machine Learning, Apress.

[7] Hussain S., Ameri Sianaki O., Ababneh N. (2019). A Survey on Conversational Agents/Chatbots Classification and Design Techniques. In: Barolli L., Takizawa M., Xhafa F., Enokido T. (eds) Web, Artificial Intelligence and Network Applications. WAINA 2019. Advances in Intelligent Systems and Computing, vol 927. Springer, Cham. https://doi.org/10.1007/978-3-030-15035-8_93

[8] Adamopoulou E., Moussiades L. (2020). An Overview of Chatbot Technology. In: Maglogiannis I., Iliadis L., Pimenidis E. (eds) Artificial Intelligence Applications and Innovations. AIAI 2020. IFIP Advances in Information and Communication Technology, vol 584. Springer, Cham. https://doi.org/10.1007/978-3-030-49186-4_31

[9] Galitsky B. (2019). Chatbot Components and Architectures. In: Developing Enterprise Chatbots. Springer, Cham

[10] Daniel Jurafsky, James Martin (2008). Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.

[11] Liu, Jiao & Li, Yanling & Lin, Min. (2019). Review of Intent Detection Methods in the Human-Machine Dialogue System. Journal of Physics: Conference Series. 1267. 012059. 10.1088/1742-6596/1267/1/012059.

[12] Singh, Abhishek & Ramasubramanian, Karthik & Shivam, Shrey. (2019). Building an Enterprise Chatbot: Work with Protected Enterprise Data Using Open Source Frameworks. 10.1007/978-1-4842-5034-1.

[13] Sunrise Central Hotel, Những câu hỏi thường gặp. Truy nhập ngày 10/10/2020: https://www.sunrisecentralhotel.com/vi/ve-chung-toi/nhung-cau-hoi-thuong-gap/

[14] Palm Hotel, Những câu hỏi thường gặp. Truy nhập ngày 10/10/2020: https://palmhotel.vn/cau-hoi-khach-san-thuong-gap/

[15] Cambridge Dictionary, Meaning of chatbot in English. Truy nhập ngày 10/10/2020: https://dictionary.cambridge.org/dictionary/english/chatbot

[16] A brief history of chatbots – Timeline. Truy nhập ngày 10/10/2020: https://roboticsbiz.com/a-brief-history-of-chatbots-timeline/

[17] Lịch sử hình thành và phát triển của chatbot. Truy nhập ngày 10/10/2020: https://congdongchatbot.com/lich-su-hinh-thanh-va-phat-trien-cua-chatbot/

[18] Top 8 Hotels in the World that Use Chatbots in Innovative Way. Truy nhập ngày 10/10/2020: https://www.trilyo.com/blog/top-8-hotels-in-the-world-that-use-chatbots-in-innovative-way/

[19] Which industries have the greatest potential for chatbot disruption? Truy nhập ngày 10/10/2020: https://www.clickz.com/which-industries-have-the-greatest-potential-for-chatbot-disruption/112840/

[20] Rasa document. Truy nhập ngày 10/10/2020: https://rasa.com/docs/

[21] Underthesea (Open-source Vietnamese Natural Language Process Toolkit) Truy nhập ngày 10/10/2020: https://github.com/undertheseanlp/underthesea