

**BỘ CÔNG THƯƠNG
TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP.HCM
KHOA CÔNG NGHỆ ĐIỆN TỬ**



BÁO CÁO THỰC TẬP DOANH NGHIỆP

ĐƠN VỊ THỰC TẬP

CÔNG TY CỔ PHẦN VIỄN THÔNG TECAPRO

**GVHD Cao Văn Kiên
SINH VIÊN Lê Huy Phát
LỚP DHDTMT12A**

**MSSV 16052431
KHÓA 12**

Tp HCM, tháng 09 năm 2020

LỜI CẢM ƠN

Lời nói đầu tiên, cho phép em được gửi lời cảm ơn chân thành và sâu sắc nhất đến toàn thể các anh chị công ty TECAPRO và đặc biệt là anh Sử, đã tạo điều kiện và hướng dẫn tận tình cho em trong quá trình thực tập và làm việc tại công ty.

TP. Hồ Chí Minh, ngày 20 tháng 09 năm 2020

Sinh viên thực hiện

Lê Huy Phát

NHẬN XÉT THỰC TẬP

Họ và tên sinh viên :

Mã sinh viên :

1. Thời gian thực tập :

Từ ngày tháng năm 2019 đến ngày tháng năm 2020

2. Bộ phận thực tập :

.....
.....
.....

3. Tinh thần trách nhiệm với công việc và ý thức chấp hành kỷ luật :

.....
.....
.....
.....

4. Kết quả thực tập:

.....
.....

5. Nhận xét chung :

.....
.....
.....
.....

6. Điểm (Thang 10):

- Tinh thần trách nhiệm và ý thức chấp hành kỷ luật (1):
- Kiến thức (2):
- Kỹ năng (3):
- Kết quả thực tập (4):

Điểm trung bình $[(1)+(2)+(3)+(4) \times 2]/5$:

Cán bộ hướng dẫn của cơ quan đến thực tập
(Ký và ghi rõ họ tên)

Ngày tháng năm

Thủ trưởng cơ quan
(Ký tên và đóng dấu)

[illegible]

GIÁO VIÊN HƯỚNG DẪN
(Ký và ghi rõ họ tên)

NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN :

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Ngày tháng năm

GIÁO VIÊN PHẢN BIỆN

(Ký và ghi rõ họ tên)

MỤC LỤC

LỜI CẢM ƠN	1
NHẬN XÉT THỰC TẬP	2
NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN :	3
NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN :	4
A. LỜI NÓI ĐẦU :	6
B. NỘI DUNG :	7
CHƯƠNG 1 : GIỚI THIỆU ĐƠN VỊ TIẾP NHẬN.....	7
CHƯƠNG 2: NỘI DUNG THỰC TẬP.....	13
I. MÔ TẢ CÔNG VIỆC ĐƯỢC GIAO	13
II. NỘI DUNG THỰC TẬP	13
1. Chatbot là gì ?	13
2. Tìm hiểu các thư viện python hỗ trợ chatbot	14
a) Thư viện Chatterbot	14
b) Thư viện Rasa	15
c) Tại sao chọn RASA Framework.	15
3. Xây dựng chatbot với Rasa	16
3.1. Trước tiên muốn xây dựng chatbot với rasa chúng ta cần hiểu về cấu trúc rasa. .	16
3.2 Cài đặt Rasa và cấu trúc cơ bản của một project.	18
3.3 Cây thư mục làm việc:	19
3.4 Xây dựng chatbot hỗ trợ tìm kiếm đồ trong kho.....	20
3.4.1 Rasa Natural Language Understanding - rasa nlu	20
3.4.2 Rasa Core	23
3.4.3 Kết thúc công đoạn setup, tiến hành tạo các file python.py để train models..	36
3.4.4 Cuối cùng chúng ta sẽ tạo các file run project.	39
3.4.5 Đưa chatbot-assistant lên docker-linux.....	46
4. Tìm hiểu về KubeEdge- mô hình Edge Computing.	52
CHƯƠNG 3 : KẾT QUẢ ĐẠT ĐƯỢC.....	56
I. TRÌNH BÀY VÀ ĐÁNH GIÁ CÁC KẾT QUẢ THỰC TẬP	56
II. ƯU ĐIỂM	56
III. NHƯỢC ĐIỂM.....	57
C. KẾT LUẬN	58
TÀI LIỆU THAM KHẢO	59

A. LỜI NÓI ĐẦU :

Là sinh viên đang học năm 3 của trường Đại học Công nghiệp TpHCM, sau 3 năm ngồi trên ghế nhà trường và được sự dẫn dắt tận tình của các thầy cô, em và các bạn sinh viên khác đã dần trưởng thành và có kiến thức ngày càng sâu rộng hơn. Qua đó nhà trường đã tạo điều kiện và đưa chúng em đi thực tập ở doanh nghiệp, nơi giúp em và các bạn sinh viên khác vận dụng những kiến thức được học vào thực tế, từ đó giúp em có thể nâng cao năng lực thực hành của bản thân mình, và có cơ hội trải nghiệm cọ xát với thực tế và tiếp cận với môi trường làm việc chuyên nghiệp. Được sự đồng ý của nhà trường và sự giúp đỡ của ban lãnh đạo công ty Cổ phần Viễn thông TECAPRO (TECAPRO Telecom), đã giúp em hoàn thành tốt đợt thực tập này. Trong thời gian thực tập tại công ty, em đã được phân công và tìm hiểu về các thư viện hỗ trợ chatbot-assistant và mục tiêu là xây dựng chatbot hỗ trợ nhân viên công ty trong việc tìm kiếm đồ trong kho. Báo cáo này là tổng hợp những hiểu biết của em về công việc em được giao và quá trình thực hiện từng bước về lập trình xây dựng chatbot Tiếng Việt có sử dụng các model “Named Entity Recognition” (NLP) và “Long short term memory” (LSTM) trong công nghệ trí tuệ nhân tạo(AI). Công nghệ AI (viết tắt của Artificial Intelligence) hoặc trí thông minh nhân tạo là công nghệ mô phỏng các quá trình suy nghĩ và học tập của con người cho máy móc, đặc biệt là các hệ thống máy tính. Các quá trình này bao gồm việc học tập (thu thập thông tin và các quy tắc sử dụng thông tin), lập luận (sử dụng các quy tắc để đạt được kết luận gần đúng hoặc xác định), và tự sửa lỗi. Các ứng dụng đặc biệt của AI bao gồm các hệ thống chuyên gia, nhận dạng tiếng nói và thị giác máy tính (nhận diện khuôn mặt, vật thể hoặc chữ viết). Khái niệm về công nghệ AI xuất hiện đầu tiên bởi John McCarthy, một nhà khoa học máy tính Mỹ, vào năm 1956 tại Hội nghị The Dartmouth. Ngày nay, công nghệ AI là một thuật ngữ bao gồm tất cả mọi thứ từ quá trình tự động hoá robot đến người máy thực tế. Công nghệ AI gần đây trở nên nổi tiếng, nhận được sự quan tâm của nhiều người là nhờ Dữ liệu lớn (Big Data), mối quan tâm của các doanh nghiệp về tầm quan trọng của dữ liệu cùng với công nghệ phần cứng đã phát triển mạnh mẽ hơn, cho phép xử lý công nghệ AI với tốc độ nhanh hơn bao giờ hết.

B. NỘI DUNG :

CHƯƠNG 1 : GIỚI THIỆU ĐƠN VỊ TIẾP NHẬN

GIỚI THIỆU CHUNG VỀ CÔNG TY

Tên công ty	CÔNG TY CỔ PHẦN VIỄN THÔNG TECAPRO
Tên giao dịch nước ngoài	TECAPRO TELECOM JOINT STOCK
Tên viết tắt	TECAPRO TELECOM
Trụ sở chính	18A Cộng Hòa, Phường 12, Quận Tân Bình, Tp. Hồ Chí Minh.
Ngày đăng ký kinh doanh	29/08/2011
Điện thoại	(84-8) 3811 9306



Tiền thân là xí nghiệp TOCA được thành lập năm 2000. Được công nhận là xí nghiệp sản xuất quốc phòng của Tổng Cục kỹ thuật.

Tecapro Telecom là một trong hai công ty con của Công ty TNHH MTV Ứng dụng Kỹ Thuật và Sản xuất (TECAPRO) trực thuộc Bộ Quốc Phòng, có

chức năng, nhiệm vụ nghiên cứu, ứng dụng chuyển giao công nghệ, sản xuất kinh doanh, phục vụ nhiệm vụ quốc phòng và kinh tế.

Tecapro Telecom đặt trụ sở chính tại 18A Cộng Hòa, phường 12, quận Tân Bình, Tp. Hồ Chí Minh. Ngoài ra, có văn phòng đại diện đặt tại 89B Lý Nam Đế, quận Hoàn Kiếm, thành phố Hà Nội.

Hiện có hơn 40 cán bộ công nhân viên, trong đó có hơn 20 kỹ sư Điện tử - Viễn thông – Tin học nghiên cứu phát triển sản phẩm.

Có khu vực nghiên cứu phát triển, chế thử rộng 4000 m².

Công ty đã sản xuất và cung cấp ra thị trường hơn 10.000 đơn vị sản phẩm, trong đó có hơn 1000 thiết bị phục vụ quốc phòng an ninh như: tổng đài, thiết bị mã hóa, mô phỏng.

Đã đào tạo cho hơn 500 lượt cán bộ kỹ thuật trong toàn quân.

Trong chiến lược phát triển của mình, TECAPRO tập trung phát triển các lĩnh vực công nghệ mũi nhọn:

○ Công nghệ thông tin

TECAPRO là nhà cung cấp có uy tín các sản phẩm và dịch vụ triển khai các giải pháp tổng thể của các hãng: DELL TECHNOLOGIES, VERTIV, JUNIPER, FUJITSU, IBM, HP, ORACLE, NEC, HITACHI, CISCO, MICROSOFT, FORTINET, CHECKPOINT, PALO ALTO, F5, FIRE EYE, VMWARE, PURE STORAGE, PULSE SECURE, NETSCOUT, BLUE COAT, RIELO, SPLUNK, BARACUDA, CYBERARK, RAPID 7, TREND MICRO, IMPERVA, MITSUBISHI, ABBYY, NUCLEUS SOFT-WARET, DATASTAX...

Tư vấn, thiết kế, triển khai các hệ thống công nghệ thông tin, các giải pháp phần mềm và các hệ thống tích hợp, bao gồm:

- Tư vấn, thiết kế, triển khai các giải pháp phần mềm tích hợp cho các ngân hàng, tổ chức tài chính: giải pháp code banking, internet banking, mobile banking...
- Cung cấp và triển khai các giải pháp, hệ thống thông tin tổng thể cho Chính phủ điện tử và dịch vụ công điện tử của các Bộ, Ngành như: hệ thống tích hợp dữ liệu tập trung, hệ thống quản trị CSDL tập trung, các hệ thống phần mềm nghiệp vụ lõi, hệ thống số hóa hồ sơ lưu trữ điện tử, hệ thống quản lý email, hệ thống quản lý văn bản điều hành, hệ thống đào tạo trực tuyến, hệ thống và các dịch vụ công trên cổng thông tin điện tử, hệ thống chăm sóc khách hàng ...
- Cung cấp và triển khai các giải pháp chuyển đổi số (Digital Transformation), quản lý dữ liệu lớn (Big Data), Trí tuệ nhân tạo (AI).
- Cung cấp và triển khai các giải pháp an ninh, bảo mật đường truyền, bảo mật cơ sở dữ liệu và các giải pháp bảo mật chuyên dụng khác.
- Cung cấp và triển khai các hệ thống máy chủ, máy trạm, các hệ thống lưu trữ dữ liệu và các thiết bị tin học.

○ Điện tử viễn thông

TECAPRO là đơn vị nghiên cứu, chế tạo, sản xuất, cung cấp, đào tạo và chuyển giao công nghệ các thiết bị, phần mềm quản lý thiết bị:

- Tổng đài chuyển mạch kỹ thuật số quân sự đa năng cấp chiến dịch, chiến lược và dã chiến dung lượng từ 16 đến 5000 số, 128 luồng E1.
- Tổng đài dân sự chuyển mạch IP dung lượng từ 100 đến 10.000 thuê bao, chuẩn quốc tế.
- Thiết bị truyền dẫn quang tốc độ cao từ 2,5Gb tới >10Gb công nghệ 5G ready.
- Thiết bị liên lạc chỉ huy, điện thoại đa hướng, giao ban xa phục vụ an ninh – quốc phòng.
- Thiết bị giám sát hành trình ô tô S-box/NFT và thiết bị cảm biến nhiên liệu F-box (giaiphapdinhvi.com).
- Cung cấp, lắp đặt các thiết bị điện tử, viễn thông, tin học và các loại camera giám sát.
- Cung cấp giải pháp an toàn thông tin, thiết bị bảo mật, mã hóa, máy tích nhúng tốc độ cao quân sự/dân sự.
- Thiết kế, chế tạo, cung cấp giải pháp, chuyển giao công nghệ điện tử, viễn thông, tin học, Iot/AI.



Các thế hệ tổng đài



Thiết bị mô phỏng Palma



**Thiết bị bảo mật
luồng E1**



Tổng đài T64SIP



Tổng đài dã chiến TIP24



Tổng đài chuyển tiếp TZ



Thiết bị bảo mật đầu cuối

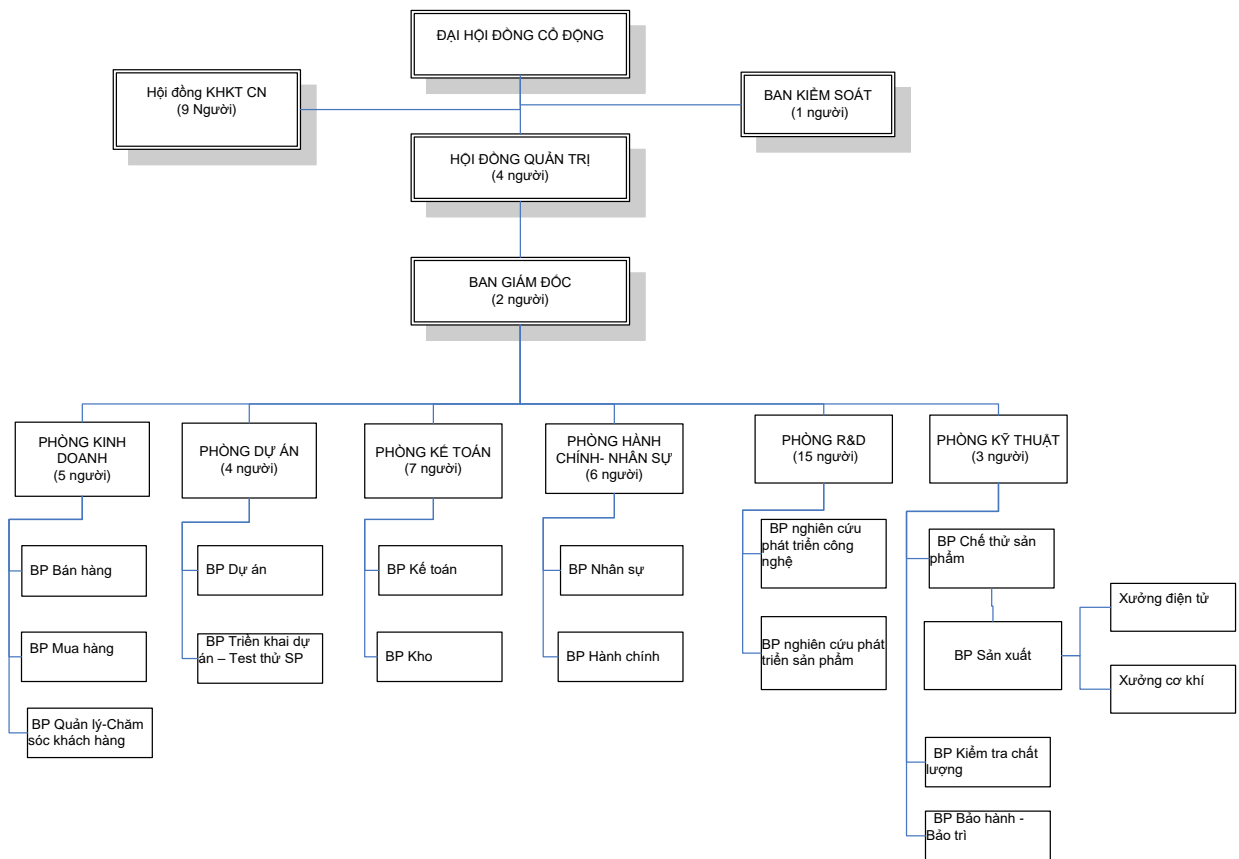


Thiết bị đa kết nối TRS

○ Công nghệ mô phỏng

- Nghiên cứu, làm chủ công nghệ nền, phát triển các giải pháp công nghệ, ưu tiên tập trung vào lĩnh vực mô phỏng thực tại ảo, tự động hóa và điều khiển, phục vụ Quốc phòng – An ninh và dịch vụ kinh tế xã hội.
- Tư vấn, xây dựng và phát triển các giải pháp công nghệ ứng dụng trong huấn luyện chiến đấu.
- Nghiên cứu, thiết kế, chế tạo thử nghiệm và sản xuất một số nhóm vật tư kỹ thuật, phụ tùng thay thế.

SƠ ĐỒ TỔ CHỨC CÔNG TY



CHƯƠNG 2: NỘI DUNG THỰC TẬP

I. MÔ TẢ CÔNG VIỆC ĐƯỢC GIAO

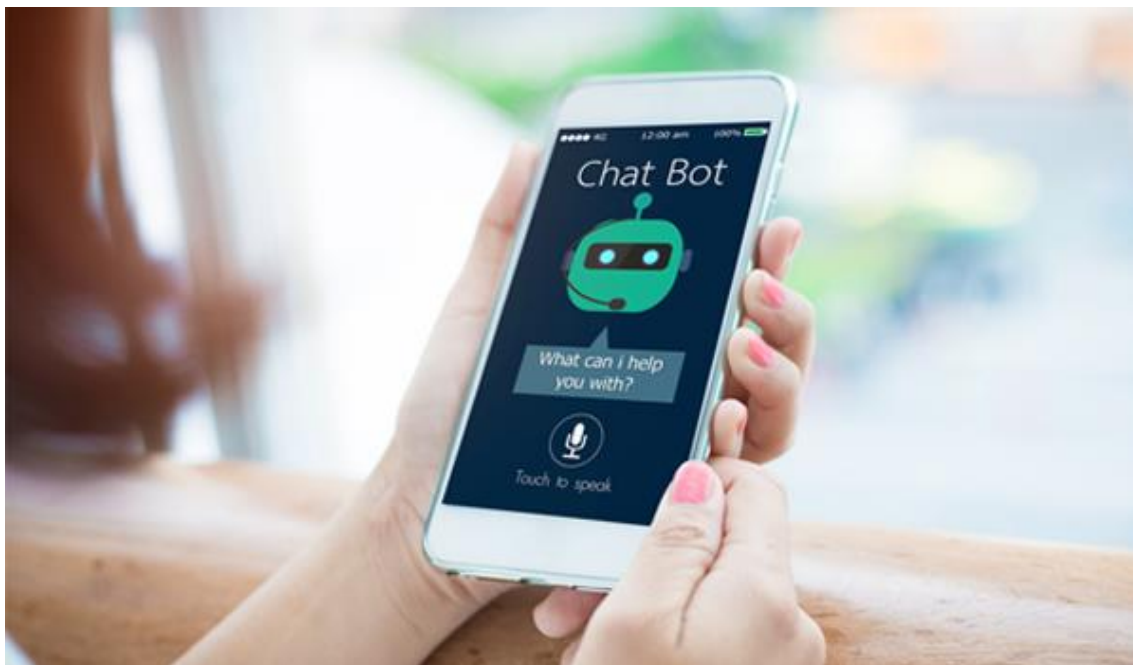
🌈 Công việc được giao gồm 3 phần :

- 1) Tìm hiểu các thư viện python hỗ trợ chatbot
- 2) Viết ứng dụng nhận yêu cầu từ người dùng, truy vấn vào cơ sở dữ liệu và trả lời kết quả đến người dùng
- 3) Tìm hiểu và làm việc với mô hình KubeEdge để xây dựng 1 hệ thống IOT với các Edge

II. NỘI DUNG THỰC TẬP

1. Chatbot là gì ?

Chatbot là một chương trình kết hợp với trí tuệ nhân tạo (AI) để tương tác với con người. Công cụ này thay thế cho nhân viên để tư vấn trả lời những gì khách hàng thắc mắc. Chatbot thường trao đổi với người dùng qua hình thức tin nhắn (Textual) hoặc âm thanh (Audiotory).



- Chatbot là 1 công cụ ứng dụng công nghệ AI để tương tác với con người
 - Các loại chatbot hiện nay
- Có nhiều cách để phân loại chatbot. Nếu xét theo khía cạnh dịch vụ thì có thể chia chatbot thành 2 loại, đó là:
- Chatbot bán hàng

- Chatbot chăm sóc khách hàng

2. Tìm hiểu các thư viện python hỗ trợ chatbot

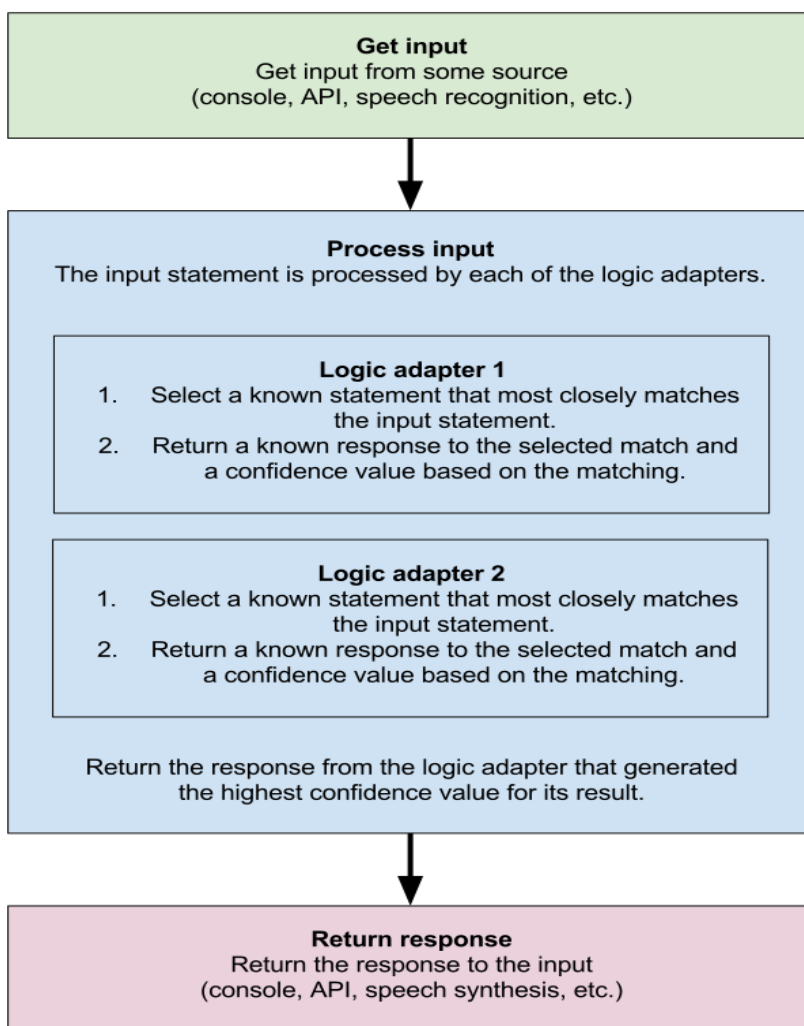
a) Thư viện Chatterbot

ChatterBot là một thư viện của Python giúp chúng ta dễ dàng tạo các phản hồi tự động cho đầu vào của người dùng. ChatterBot sử dụng các thuật toán machine learning để xử lý dữ liệu với nhiều ngữ cảnh khác nhau. Điều này giúp các nhà phát triển dễ dàng tạo ra các bot trò chuyện và tự động hóa các cuộc hội thoại với người dùng.

Để biết thêm chi tiết chúng ta có thể đọc Document của chatterbot tại link:

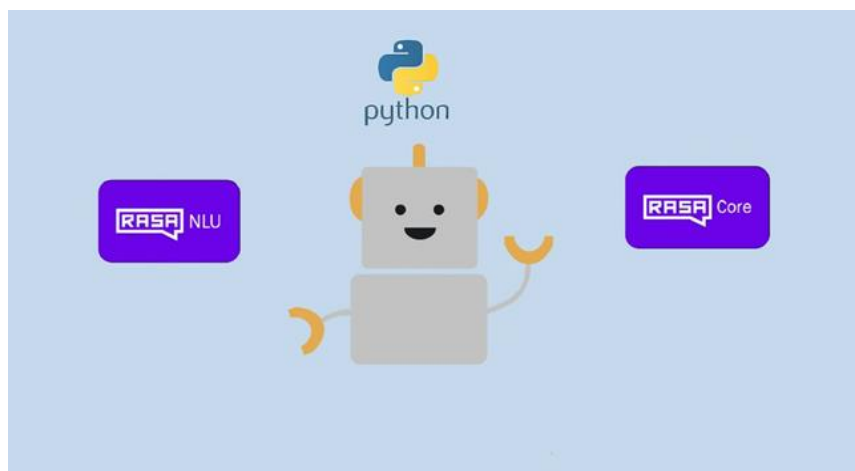
<https://chatterbot.readthedocs.io/en/stable/>

- Cách thức hoạt động của chatterbot:



b) Thư viện Rasa

- Rasa là một open source machine learning framework cho các cuộc hội thoại dựa trên văn bản và giọng nói tự động. Hiểu tin nhắn, tổ chức cuộc trò chuyện và kết nối với các kênh nhắn tin và API.
- Rasa có những framework hỗ trợ rất mạnh mẽ như 'rasa core', 'rasa nlu'
- Rasa thực sự dễ tiếp cận cho người mới bắt đầu, thậm chí là người không biết gì về chatbot hay NLP. 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.
- Mã nguồn của Rasa là mã nguồn mở, do đó Rasa giúp bạn biết chính xác được bạn đang làm gì với chatbot của mình, thậm chí có thể custom chatbot theo ý thích của bản thân, ví dụ như: *custom action*, *custom tokenizer*, hay cả việc quyết định nền tảng tin nhắn của chatbot *custom connector*.



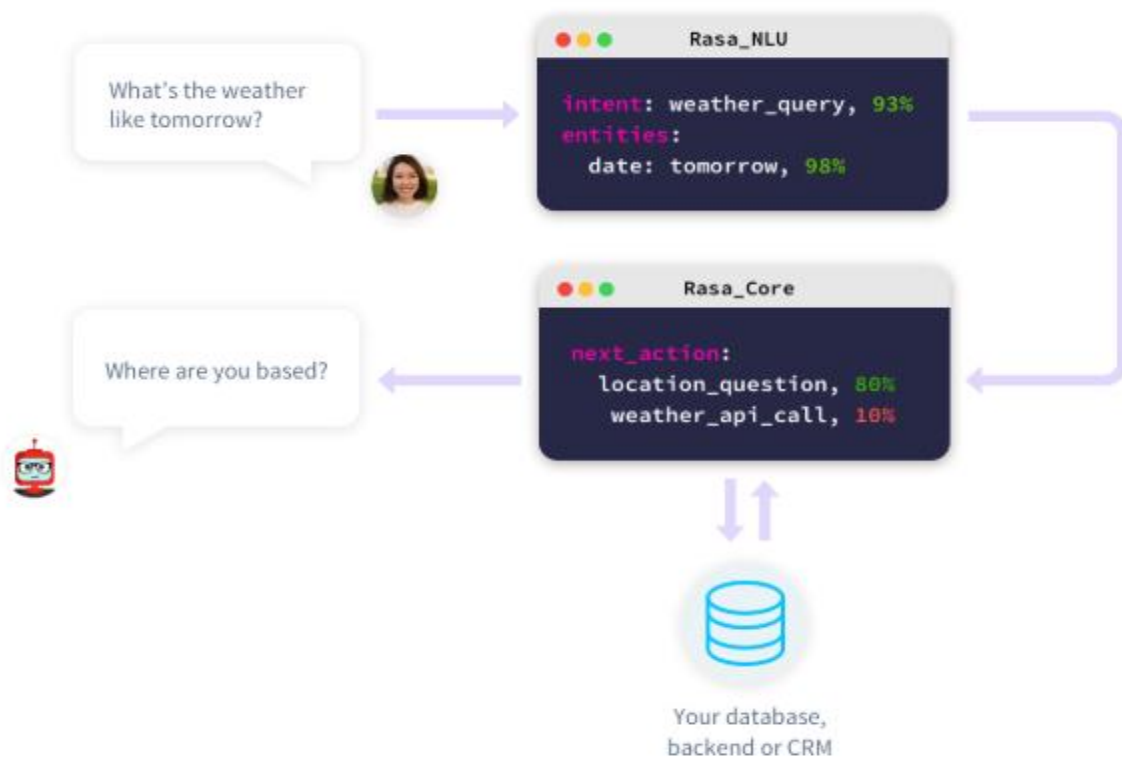
c) Tại sao chọn RASA Framework.

- Rasa dễ tiếp cận với những người mới bắt đầu tiếp cận và nghiên cứu về chatbot, hay chỉ đơn giản là nảy ra một tưởng xây dựng một chú "bot" thus vj cos theer "chat", hoặc cập nhật tin tức, hoặc làm một tác vụ gì đó phức tạp nhưng chưa biết bắt đầu từ đâu.
- Rasa có cộng đồng hỗ trợ lớn, rasa hỗ trợ tiếng Việt.
- Rasa hoạt động khá tốt và mạnh mẽ, đặc biệt trong vấn đề xác định ý định người dùng (intent) và đối tượng được nhắc đến trong câu (entity) dù dữ liệu bạn thu thập và cung cấp cho rasa là vô cùng ít.

- Rasa là nguồn mở

3. Xây dựng chatbot với Rasa

3.1. Trước tiên muốn xây dựng chatbot với rasa chúng ta cần hiểu về cấu trúc rasa.



Rasa là một nền tảng chatbot bao gồm :

- Natural Language Unit (NLU)
- The Rasa Core Dialogue Engine
- Rasa X

Một số thuật ngữ chúng ta cần nắm khi làm việc với Rasa:

1. **Rasa NLU** - một thư viện để hiểu ngôn ngữ tự nhiên (NLU) thực hiện phân loại ý định và trích xuất thực thể từ đầu vào của người dùng và giúp bot hiểu những gì người dùng đang nói.

2. **Rasa Core** - một khung chatbot với quản lý hội thoại, lấy đầu vào có cấu trúc từ NLU. NLU và Core là độc lập và người ta có thể sử dụng NLU mà không cần Core, và ngược lại.
3. **Rasa X** : là một tool giúp chúng ta xây dựng, cải thiện và triển khai model chatbot vừa tạo.
4. **Intent**: RASA cần biết người dùng muốn gì, vì vậy cần nhận ra ý định của họ. Ví dụ: Người dùng nói rằng : "Tôi muốn đặt bàn cho 2 người tối nay tại nhà hàng ABC" thì intent ở đây là việc đặt bàn.
5. **Entity**: thực thể là để trích xuất thông tin từ đầu vào của người dùng. Như ví dụ ở trên thì entity ở đây chính là thời gian và địa điểm đặt bàn
6. **Stories** : Câu chuyện xác định sự tương tác giữa người dùng và chatbot theo intent và action được thực hiện bởi bot. Giống như trong ví dụ trên, bot có ý định đặt bàn và các thực thể như địa điểm, thời gian và điều đó sẽ thực hiện hành động tiếp theo từ bot.
7. **Action**: Actions về cơ bản là các hoạt động được thực hiện bởi bot hoặc yêu cầu thêm một số chi tiết để có được tất cả các thực thể hoặc tích hợp với một số API hoặc truy vấn cơ sở dữ liệu để nhận / lưu một số thông tin.

3.2 Cài đặt Rasa và cấu trúc cơ bản của một project.

- **Yêu cầu :** Hệ điều hành linux hoặc windows

+ Linux : ubuntu 18.04

+ Windows : window 10

+ Python 3.6+

- **Các bước cài đặt rasa framework trênUbuntu:**

+ Cài đặt môi trường ảo trong thư mục làm việc (chatbot):

```
:~$ cd Desktop/  
: ~/Desktop$ mkdir chatbot  
: ~/Desktop/chatbot$ sudo pip3 install virtualenv  
: ~/Desktop/chatbot$ virtualenv -p python3 envcb
```

+ Truy cập và làm việc trong môi trường ảo:

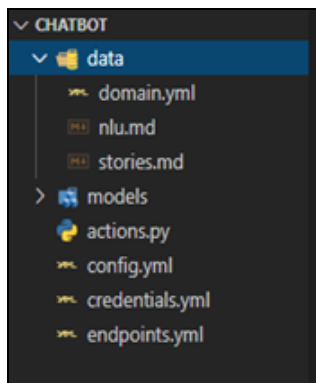
```
: ~/Desktop/chatbot$ source envcb/bin/activate  
(envcb) /Desktop/chatbot$
```

+ Cài đặt các gói framework của RASA và thư viện cần thiết :

```
(envcb) /Desktop/chatbot$ pip install --upgrade pip setuptools wheel  
(envcb) /Desktop/chatbot$ pip install rasa_core rasa_nlu rasa_core_sdk  
feedparser spacy sklearn_crfsuite  
(envcb) /Desktop/chatbot$ python -m spacy download en  
(envcb) /Desktop/chatbot$ pip install tornado  
(envcb) /Desktop/chatbot$ pip install --pre requests  
(envcb) /Desktop/chatbot$ pip install bs4  
(envcb) /Desktop/chatbot$ pip install pyvi
```

3.3 Cây thư mục làm việc:

- Cây thư mục rasa chatbot của chúng ta sẽ có cấu trúc các thành phần chính như sau :



actions.py: Nơi chúng ta sẽ code tất cả mọi hành động tùy chỉnh mà bạn muốn bot làm

config.yml: Nơi chúng ta cấu hình các thông tin liên quan tới mô hình NLU và Core, cách mà chúng hoạt động.

credentials.yml: Thông tin chi tiết về cách kết nối chatbot với các dịch vụ như Facebook, Slack, Telegram,...

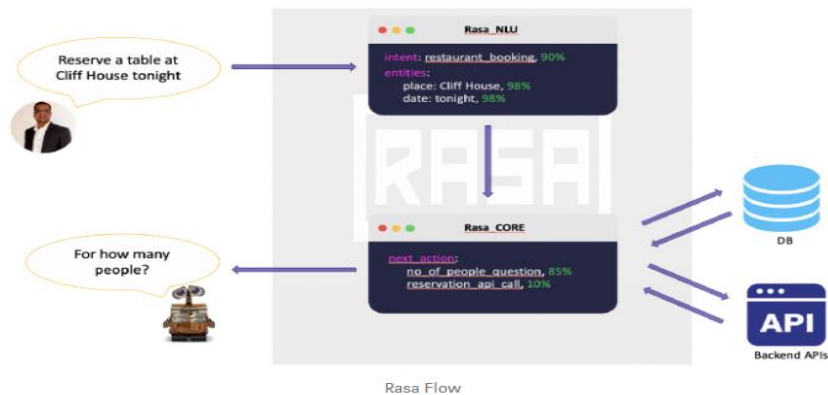
data/nlu.md: Dữ liệu huấn luyện cho NLU, bao gồm các câu được gán nhãn intent và entities theo định dạng cho trước. Nó chứa dữ liệu đào tạo về mặt đầu vào của người dùng cùng với việc ánh xạ các intent và entity có trong mỗi câu. Nó có thêm các ví dụ khác nhau mà chúng ta cung cấp, giúp cho khả năng của bot sẽ trở nên tốt hơn. Việc bạn tạo ra nhiều dữ liệu train và đa dạng cho con bot thì độ chính xác của nó càng cao.

data/stories.md: Dữ liệu huấn luyện cho Rasa core, là các kịch bản mà chúng ta muốn bot làm theo.

Data/domain.yml: Đây coi như phần khai báo tất cả mọi thứ mà chatbot của chúng ta sử dụng, bao gồm các intent, entities, actions,...

endpoints.yml: Các endpoints mà chúng ta muốn chatbot trả ra

models/: Nơi lưu trữ các model bạn đã huấn luyện



3.4 Xây dựng chatbot hỗ trợ tìm kiếm đồ trong kho.

3.4.1 Rasa Natural Language Understanding - rasa nlu

- Như đã tóm tắt về NLU ở trên, chúng ta cần dạy con bot của chúng ta để nó có thể hiểu các thông điệp của chúng ta trước. Vì vậy, chúng ta phải đào tạo mô hình

```
language: "en"  
pipeline: "spacy_sklearn"
```

NLU với các đầu vào ở định dạng văn bản đơn giản và trích xuất dữ liệu có cấu trúc. Chúng ta sẽ đạt được điều này bằng cách xác định intent và cung cấp một vài cách người dùng có thể thể hiện chúng.

B1) Đầu tiên chúng ta tạo file *config.yml*

- Trên cửa sổ Terminal chúng ta tạo file *config.yml* như sau:

```
(envcb) /Desktop/chatbot$ mkdir config &&cd config &&touch config.yml
```

Đây là phần cấu hình cho NLU, nơi chúng ta lựa chọn ngôn ngữ, model cần thiết.

- Chúng ta gõ 2 dòng sau vào cuối file.

Đó là một quy trình hoàn chỉnh từ lựa chọn *Tokenizer*, *Featurizer*, *Extractor* đến *Classifier*. Tất nhiên chúng ta hoàn toàn có thể lựa chọn thay thế bất cứ một công đoạn nào trong *pipeline* này nếu chúng ta cảm thấy nó sẽ đạt hiệu quả tốt hơn.

Bên cạnh *spacy_sklearn* thì Rasa cũng cung cấp sẵn thêm một vài pipeline template khác là *supervised_embeddings*, *pretrained_embeddings_spacy*, *pretrained_embeddings_convert*, ...

spaCy + sklearn : spaCy là một thư viện NLP chỉ thực hiện trích xuất thực thể. sklearn được sử dụng với spaCy để thêm các khả năng ML để phân loại ý định.

- Chúng ta có thể tham khảo các pipeline ở đây:

<https://rasa.com/docs/rasa/core/policies/#policies>

B2) Tạo file *nlu.md*

- Trên cửa sổ Terminal chúng ta tạo file nlu.md như sau:

```
(envcb) /Desktop/chatbot$ mkdir data &&cd data &&touch nlu.md
```

- Chúng ta custom file này dạng như sau:

```
## intent:ask_func_list
- bạn có thể làm được những gì
- bạn giúp được gì nào
- chức năng của bạn là gì
- bạn có thể làm được mấy chức năng
- bạn giỏi nhất làm gì
- bạn có tư vấn giúp mình được không
## intent:ask_name
- bạn tên gì
- tên gì
- chị tên gì
- anh tên gì
- tên của bạn là gì
- tên của chị là gì
## intent:bye
- tạm biệt
- chào tạm biệt
- chào tạm biệt em
- tạm biệt em
- tạm biệt em nhé
## intent:give_name
- [Anh](cust_sex) là [Phát](cust_name)
- [Anh](cust_sex) là [Sử](cust_name)
- [Chị](cust_sex) là [Hằng](cust_name)
- [Em](cust_sex) là [Trang](cust_name)
- [Cô](cust_sex) [Vân](cust_name)
- [Chú](cust_sex) [Hùng](cust_name)
## intent:greet
- xin chào
- chào bạn
- hello
- hi
- hey
## intent:linh_kien
- c1815 giá bao nhiêu?
- bc337 giá bao nhiêu?
- 2n3904 giá bao nhiêu?
- điện trở r100K giá bao nhiêu?
- điện trở r10K giá bao nhiêu?
- điện trở r50K giá bao nhiêu?
## intent:loai_linh_kien
- transistor trong kho có những loại gì?
- điện trở trong kho có những loại nào?
- transistor
- điện trở

## intent:thank
- cảm ơn
- thanks
- thank you
```

Phía trên là file **nlu.md** tương đối ngắn, mục đích chỉ là để demo nên lượng data training phần NLU khá là ít, chúng ta có thể thêm các dạng câu hỏi khác vào phần này, càng nhiều câu hỏi thì chatbot của chúng ta sẽ thông minh hơn. Và đó là công việc của chúng ta. Với **pipeline** đã được chúng ta duyệt qua trong file **config.yml** thì chúng ta cần thêm data để **train model** và file **nlu.md** chịu trách nhiệm về data đó. Ở đây chúng ta có các câu message người dùng có thể hỏi đã được gán nhãn là intent tương ứng.

3.4.2 Rasa Core

Rasa Core là nơi thực hiện quản lí luồng hội thoại. Dựa vào các intent, entity đã được detect ra ở phần NLU, Rasa Core tiến hành lấy các kết quả này làm đầu vào, rồi quyết định message đầu ra.

B1) Đầu tiên, vẫn là cần cấu hình cho Rasa Core trong file **config.yml**

- Chúng ta thêm đoạn mã sau vào phần policies:

```
#Configuration for Rasa Core.
policies:
- name: MemoizationPolicy
  max_history: 1
- name: KerasPolicy
  epochs: 200
  batch_size: 20
- name: MappingPolicy
- name: FallbackPolicy
  nlu_threshold: 0.3
  core_threshold: 0.3
  fallback_action_name: 'utter_unclear'
```

- Chúng ta lần lượt khai báo cái Policy cần thiết. Ở đây, chúng ta cần dùng một số policy như: **MemoizationPolicy** (quyết định message đầu ra dựa vào thông tin của những đoạn hội thoại trước đó), **KerasPolicy** (sử dụng mạng LSTM để tính xác suất đưa ra lựa chọn cho message tiếp theo), **MappingPolicy** (quyết định message dựa vào dữ liệu đã mapping) và trong trường hợp việc tính xác suất đầu ra không thể vượt được ngưỡng mà **FallbackPolicy** đề ra, message trả ra sẽ là một **utter_unclear** kiểu như: "Xin lỗi anh chị ạ, em không hiểu được nội dung anh chị nói ạ".
- Chúng ta có thể tham khảo các Action Selection tại link sau:
<https://rasa.com/docs/rasa/core/policies/#id11>

- Ngoài ra chúng ta có thể xử lý phần policies này trực tiếp khi train rasa core trong file test_dialog.py (sẽ được đề cập sau).

B2) Tiếp theo, chúng ta cần khai báo các thông tin cần thiết trong file *domain.yml*

- Trên cửa sổ Terminal chúng ta tạo file domain.yml như sau:

```
(envcb) /Desktop/chatbot$ mkdir data &&cd data &&touch domain.yml
```

- Chúng ta custom file này như sau:

```
session_config:
  session_expiration_time: 0.0
  carry_over_slots_to_new_session: true

intents:
  # Ý đồ khách hàng
  - greet
  - thank
  - bye
  - ask_func_list
  - ask_name
  - give_name
  - loai_linh-kien
  - linh_kien

entities:
  - cust_sex
  - cust_name
  - linh_kien

slots:
  cust_sex:
    auto_fill: true
    type: text
  cust_name:
    auto_fill: true
    type: text
  linh_kien:
    auto_fill: false
    type: unfeaturized

templates:
  # Bot trả lời
  utter_greet:
    - text: "Xin chào, hãy cho tôi biết tên của bạn để tôi dễ xưng hô nhé."
  utter_greet_with_name:
    - text: "Chào {cust_sex} {cust_name}. Tecapro-Telecom có thể giúp gì được {cust_sex} {cust_name} ạ?"
```

```
utter_bye:
- text: "Chào tạm biệt {cust_sex} {cust_name} và hẹn gặp lại!"
- text: "Kính chào tạm biệt và chúc quý khách một ngày tốt lành!"
utter_thank:
- text: "Cảm ơn bạn!"
- text: "Không có gì, đó là trách nhiệm của tôi!"
utter_func_list:
- text: "Tôi có thể tìm mọi thứ có trong kho cho bạn!"
utter_ask_name:
- text: "Tôi tên Bot họ Chat, luôn vui vẻ và nhiệt tình, bạn cần tôi giúp gì không?"
- text: "Em tên Bot họ Chat!"
utter_unclear:
- text: "Tôi vẫn chưa hiểu yêu cầu."

actions:
# templates (as they are reply actions),
# also custom actions if any
- utter_greet
- utter_greet_with_name
- utter_bye
- utter_unclear
- utter_func_list
- utter_thank
- utter_ask_name
- action_custom_loai_linh_kien
- action_custom_linh_kien
```

Ở đây:

- **intent**: là các thông tin đã nêu trong file nlu, (có thể có cả entity),
- **action**: là phần liệt kê các hành động, message đầu ra mà chúng ta định nghĩa.
- **templates**: là phần chúng ta định nghĩa các message dạng text, hoặc hình ảnh, ... (các response này thường có dạng utter_{{}}) mà bot sẽ gửi lại cho người dùng.
- **slot** : về cơ bản là bộ nhớ của bot. Chúng hoạt động như một kho lưu trữ key-value có thể được sử dụng để lưu trữ thông tin mà người dùng đã cung cấp (ví dụ: thành phố quê hương 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).
- Với các action cần thao tác với database, chúng ta định nghĩa trong file action.py (lát nữa mình sẽ nói cụ thể hơn)
- Cuối cùng là session_config, là phần cấu hình cho một session như thời gian để restart lại một session, có mang slot từ session cũ sang session mới hay không, ...

B3) Sau khi khai báo trong domain, chúng ta xây dựng các kịch bản cần thiết cho việc trò chuyện của "bot". Nhìn chung, phần này có logic khá giống if-else. Chúng ta hãy nhìn vào file *stories.md*

- Trên cửa sổ Terminal chúng ta tạo file stories.md như sau:

```
(envcb) /Desktop/chatbot$ mkdir data &&cd data &&touch stories.md
```

- Chúng ta custom file này như sau:

```
## Chào hỏi đưa tên
* greet
  - utter_greet
* give_name
  - utter_greet_with_name
* ask_name
  - utter_ask_name
* bye
  - utter_bye

## Đưa tên luôn
* give_name
  - utter_greet_with_name

## Tam biệt luôn
* bye
  - utter_bye

## Chào - tên - hỏi chức năng - chào
* greet
  - utter_greet
* ask_name
  - utter_ask_name
* ask_func_list
  - utter_func_list
* bye
  - utter_bye

## Chào - hỏi chức năng - chào
* greet
  - utter_greet
* ask_func_list
  - utter_func_list
* bye
  - utter_bye
```

```
## Chào - hỏi tên - chào
* greet
  - utter_greet
* ask_name
  - utter_ask_name
* bye
  - utter_bye

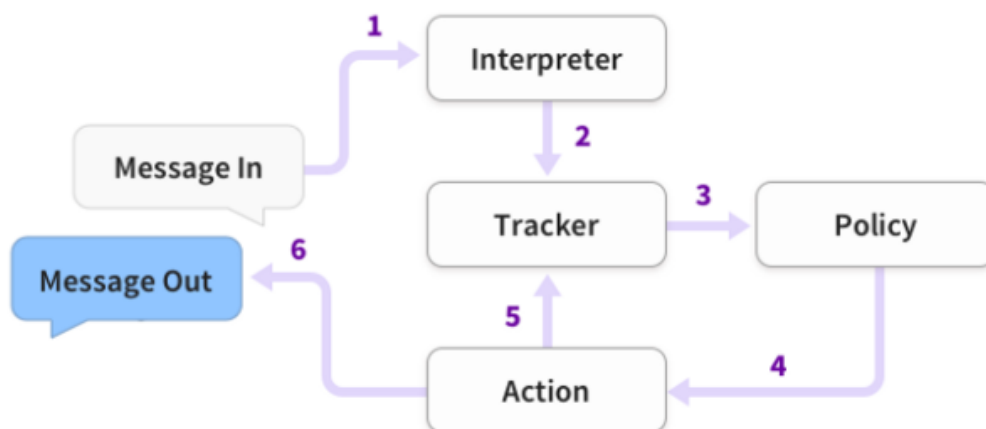
## Hỏi tên - hỏi chức năng
* ask_name
  - utter_ask_name
* ask_func_list
  - utter_func_list

## Cảm ơn
* thank
  - utter_thank

## loại_linh_kien
* loại_linh_kien
  - action_custom_loai_linh_kien

## linh_kien
* linh_kien
  - action_custom_linh_kien
```

- Việc dự tính trước các luồng hội thoại và xây dựng sẵn một kịch bản sẽ giúp con bot của chúng ta xử lý một cách trơn tru hơn và vó vẻ là thông minh hơn.



Rasa Core High Level Architecture (Source — Rasa Core)

B4) Custom file *actions.py*

- Trên cửa sổ Terminal chúng ta tạo file actions.py như sau:

```
(envcb) /Desktop/chatbot$ touch actions.py
```

- Về cơ bản, một chatbot sẽ luôn cần có một database để lưu trữ thông tin: đó có thể là ngân hàng câu hỏi, thông tin về lĩnh vực bot được hỏi, các thao tác với database, ...do đó, chúng ta cũng cần có những xử lý riêng theo từng tác vụ. Rasa hỗ trợ điều đó trong file *actions.py*
- Trước khi custom file actions.py, chúng ta sẽ xây dựng gói cơ sở dữ liệu trên local:
- Ở đây chúng ta sử dụng cơ sở dữ liệu SQLite, trên cửa sổ Terminal chúng ta tạo file db_sqlite.py như sau:

```
(envcb) /Desktop/chatbot$ touch db_sqlite.py  
(envcb) /Desktop/chatbot$ sudo nano db_sqlite.py
```

Vì dữ liệu demo lấy là nhỏ nên chúng ta sẽ tạo database bằng dòng lệnh, đối với dữ liệu lớn cần nhập vào chúng ta cần sử dụng đến tool “DB Browser for SQLite”.

+ Khởi tạo file khohangrasa.db và tạo table “KHOHANG”

```
##khohangrasa.db  
import sqlite3  
  
conn = sqlite3.connect('khohangrasa.db')  
  
print ("Opened database successfully")  
  
conn.execute("CREATE TABLE KHOHANG(  
    linh_kien    TEXT PRIMARY KEY NOT NULL,  
    loai_linh_kien    TEXT NOT NULL,  
    gia        INTEGER NOT NULL);")  
print ("Table created successfully")  
  
conn.close()
```

+ Trên cửa sổ Terminal thực hiện:

```
(envcb) /Desktop/chatbot/$chmod +x db_sqlite.py  
(envcb) /Desktop/chatbot$python ./db_sqlite.py
```

Nếu output giống như sau thì đã tạo table “KHOHANG” thành công:

```
Opened database successfully
Table created successfully
```

+ Thêm giá trị mẫu vào các cột đã được tạo, chúng ta thực hiện trong file khohangrasa.db:

```
##khohangrasa.db
import sqlite3

conn = sqlite3.connect('khohangrasa.db')

print ("Opened database successfully")

#conn.execute("CREATE TABLE KHOHANG(
#    linh_kien    TEXT PRIMARY KEY NOT NULL,
#    loai_linh_kien TEXT NOT NULL,
#    gia          INTEGER NOT NULL);")
#print ("Table created successfully")

conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
VALUES ('c1815', 'transistor', 100 )")

conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
VALUES ('bc337', 'transistor', 150 )")

conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
VALUES ('2n3904', 'transistor', 200 )")

conn.commit()
print ("Records created successfully")
conn.close()
```

+ Trên cửa sổ Terminal thực hiện:

```
(envcb) /Desktop/chatbot$python ./db_sqlite.py
```

Nếu output giống như sau thì đã Insert các giá trị thành công:

```
Opened database successfully
Records created successfully
```

+ Kiểm tra database của chúng ta xem dữ liệu nhập vào đúng theo ý muốn chưa, chúng ta thực hiện trong file khohangrasa.db:

```
##tao khohang.db
import sqlite3

conn = sqlite3.connect('khohangrasa.db')

print ("Opened database successfully")

#conn.execute("CREATE TABLE KHOHANG(
#    linh_kien    TEXT PRIMARY KEY NOT NULL,
#    loai_linh_kien TEXT NOT NULL,
#    gia          INTEGER NOT NULL);")
#print ("Table created successfully")

#conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
#    VALUES ('c1815', 'transistor', 100 )")

#conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
#    VALUES ('bc337', 'transistor', 150 )")

#conn.execute("INSERT INTO KHOHANG (linh_kien, loai_linh_kien, gia) \
#    VALUES ('2n3904', 'transistor', 200 )")

#conn.commit()
#print ("Records created successfully")

cursor = conn.execute("SELECT linh_kien, loai_linh_kien, gia from KHOHANG")
for row in cursor:
    print ("linh_kien = ", row[0])
    print ("loai_linh_kien = ", row[1])
    print ("gia = ", row[2], "\n")

print ("Operation done successfully")

conn.close()
```

+ Trên cửa sổ Terminal thực hiện:

```
(envcb) /Desktop/chatbot$python ./db_sqlite.py
```

Nếu output có dạng giống như sau thì đã thành công:

```
Opened database successfully
linh_kien = c1815
loai_linh_kien = transistor
gia = 100

linh_kien = bc337
loai_linh_kien = transistor
gia = 150

linh_kien = 2n3904
loai_linh_kien = transistor
gia = 200

Operation done successfully
```

- Sau khi đã có database cần thiết chúng ta tiến hành custom file actions.py như sau:

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function
from __future__ import unicode_literals

from rasa_core_sdk import Action

# from rasa_core_sdk.events import SlotSet
from rasa_core_sdk.events import UserUtteranceReverted
from rasa_core_sdk.events import AllSlotsReset
from rasa_core_sdk.events import Restarted

import requests

# import json
from bs4 import BeautifulSoup
from pyvi import ViTokenizer, ViPosTagger

###
from typing import Text, List, Dict, Any
from rasa_core_sdk import Tracker
from rasa_core_sdk.events import SlotSet, ActionExecuted, EventType
from rasa_core_sdk.executor import CollectingDispatcher
from rasa_core_sdk.events import SlotSet
import json
import re
import sqlite3
```



```
# from rasa_core_sdk.events import SessionStarted
# connect to database
sqliteConnection = sqlite3.connect("khohangrasa.db")
cursor = sqliteConnection.cursor()
print("Database created and Successfully Connected to SQLite")
# cursor.execute("SELECT linh_kien, loai_linh_kien, gia from KHOHANG")
# record = cursor.fetchall()
# for index in record:
#     linh_kien = index[0].lower()
#     loai_linh_kien = index[1].lower()
#     gia = index[2]
#     print(linh_kien)
#     print(loai_linh_kien)
#     print(gia)

def name_cap(text):
    tarr = text.split()
    for idx in range(len(tarr)):
        tarr[idx] = tarr[idx].capitalize()
    return " ".join(tarr)

class action_save_cust_info(Action):
    def name(self):
        return "action_save_cust_info"

    def run(self, dispatcher, tracker, domain):
        user_id = (tracker.current_state())["sender_id"]
        print(user_id)
        cust_name = next(tracker.get_latest_entity_values("cust_name"), None)
        cust_sex = next(tracker.get_latest_entity_values("cust_sex"), None)
        bot_position = "Tecapro-Telecom"

        if cust_sex is None:
            cust_sex = "Quý khách"

        if (cust_sex == "anh") | (cust_sex == "chị"):
            bot_position = "em"
        elif (cust_sex == "cô") | (cust_sex == "chú"):
            bot_position = "cháu"
        else:
            cust_sex = "Quý khách"
            bot_position = "Tecapro-Telecom"

        if not cust_name:
            dispatcher.utter_template("utter_greet_name", tracker)
            return []

        print(name_cap(cust_name))
        return [
            SlotSet("cust_name", " " + name_cap(cust_name)),
            SlotSet("cust_sex", name_cap(cust_sex)),
            SlotSet("bot_position", name_cap(bot_position)),
        ]
```

```

class action_save_mobile_no(Action):
    def name(self):
        return "action_save_mobile_no"

    def run(self, dispatcher, tracker, domain):
        user_id = (tracker.current_state())["sender_id"]
        print(user_id)
        mobile_no = next(tracker.get_latest_entity_values("inp_number"), None)

        if not mobile_no:
            return [UserUtteranceReverted()]

        mobile_no = mobile_no.replace(" ", "")
        # print (cust_name)
        return [SlotSet("mobile_no", mobile_no)]

# class clear cac bo nho slot cua Bot sau khi 1 messenger toi
class action_reset_slot(Action):
    def name(self):
        return "action_reset_slot"

    def run(self, dispatcher, tracker, domain):
        return [
            SlotSet("transfer_nick", None),
            SlotSet("transfer_amount", None),
            SlotSet("transfer_amount_unit", None),
        ]

#### Database class
## class xu ly event loai_linh_kien
class ActionAskKnowledgeBaseLoaiLinhKien(Action):
    def name(self) -> Text:
        return "action_custom_loai_linh_kien"

    def run(
        self,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any],
    ) -> List[Dict[Text, Any]]:
        text = tracker.latest_message["text"]
        text_input = text.lower()
        cursor.execute("SELECT linh_kien, loai_linh_kien, gia from KHOHANG")
        record = cursor.fetchall()
        check = False
        for result in record:
            linh_kien = result[0].lower()
            loai_linh_kien = result[1].lower()
            # gia = str(result[2])
            if loai_linh_kien in text_input:
                check = True
                dispatcher.utter_message(loai_linh_kien + ": " + linh_kien)
        if not check:
            dispatcher.utter_message("Hiện tại trong kho không có loại linh kiện này.")
        return [SlotSet("linh_kien", loai_linh_kien)]

```

```
## class event xử lý linh_kien
class ActionAskKnowledgeBaseLinhKien(Action):
    def name(self) -> Text:
        return "action_custom_linh_kien"

    def run(
        self,
        dispatcher: CollectingDispatcher,
        tracker: Tracker,
        domain: Dict[Text, Any],
    ) -> List[Dict[Text, Any]]:
        text = tracker.latest_message["text"]
        text_input = text.lower()
        cursor.execute("SELECT linh_kien, loai_linh_kien, gia from KHOHANG")
        record = cursor.fetchall()
        linh_kien_slot = str(
            tracker.get_slot("linh_kien")
        ).lower() # lấy ra slot được khai báo trong domain
        check = False
        for index in record:
            linh_kien = index[0].lower()
            # loai_linh_kien = index[1].lower()
            gia = str(index[2])
            if linh_kien in text_input:
                check = True
                dispatcher.utter_message(linh_kien + " có giá: " + gia)
        if linh_kien_slot in text_input:
            check = True
            dispatcher.utter_message(
                "Xin lỗi hình như bạn đang nhầm giữa tên 'linh kiện' với 'loại linh kiện'"
            )
        elif not check:
            dispatcher.utter_message("Hiện tại trong kho không có loại linh kiện này.")
        return [SlotSet("linh_kien", gia)]
```

Với mỗi action cụ thể, chúng ta xây dựng riêng một class. Class này có đặc điểm sau: chỉ bao gồm 2 method là `name()` và `run()`

- `name()` sẽ trả về tên của action, cái mà chúng ta khai báo trong file domain và file stories
- `run()` là nơi chúng ta thỏa sức sáng tạo, làm điều ta muốn (cụ thể là code python cho cái action này hoạt động).

Ngoài **Action**, chúng ta còn có các khái niệm khác về **Slot**, **Form Action**, **Tracker**, ... những công cụ hữu ích cho việc custom cho Rasa.

- Chúng ta có thể tham khảo thêm các khái niệm trên ở link sau:

<https://rasa.com/docs/rasa/api/rasa-sdk/https://rasa.com/docs/rasa/core/actions/>

B5) Sau khi đã có file *actions.py*, muốn action hoạt động được chúng ta cần thêm file *endpoints.yml*

- Chúng ta thêm 2 dòng dưới vào cuối file:

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

- Action của Rasa sẽ chạy trên một cổng hoàn toàn riêng biệt với rasa, vậy nên, nếu quên thì action của chúng ta sẽ không thể hoạt động được. Rasa sẽ run trên cổng 5005 trong khi đó action sẽ run trên cổng 5055

B6) để kết nối rasa với các nền tảng nhắn tin khác (vd: Mattermost) chúng ta xem đến file *credentials.yml* :

- Công việc của chúng ta chỉ cần uncomment phần code đó, sau đó điền thông tin về chatbot của mình vào. Với những nền tảng chưa được hỗ trợ mặc định, chúng ta cũng có thể custom nó theo ý muốn.
- Chúng ta có thể tham khảo link sau để có thể custom được file này:

<https://viblo.asia/p/cach-ket-noi-chatwork-voi-rasa-va-5-phut-mac-niem-latency-tren-troi-924IJb0IPM>

VD: Để có thể kết nối nền tảng nhắn tin mattermost với chatbot của chúng ta, chúng ta điền thông tin về chatbot của mình vào các dòng sau trong file *credentials.yml*:

```
mattermost:  
  url: "https://<mattermost instance>/api/v4"  
  token: "<bot token>"  
  webhook_url: "<callback URL>"
```

3.4.3 Kết thúc công đoạn setup, tiến hành tạo các file python.py để train models.

B1) Train rasa nlu.

- Chúng ta tạo file **train_nlu.py (train rasa nlu)**:

```
(envcb) /Desktop/chatbot$ touch train_nlu.py
(envcb) /Desktop/chatbot$ sudo nano train_nlu.py
```

- Chúng ta code trong file này như sau:

```
# Imports
# -----
# rasa nlu
from rasa_nlu.training_data import load_data
from rasa_nlu.config import RasaNLUModelConfig
from rasa_nlu.model import Trainer
from rasa_nlu import config
from rasa_nlu.model import Metadata, Interpreter

# Ham train NLU
# -----
def train(data, config_file, model_dir):
    training_data = load_data(data)
    trainer = Trainer(config.load(config_file))
    trainer.train(training_data)
    model_directory = trainer.persist(model_dir, fixed_model_name="chat")

# Tien hanh train modul NLU
# Input : File nlu.md
# Output: Model NLU trong thu mục models/nlu
train("data/nlu.md", "config/config.yml", "models/nlu")

# Load modul NLU
interpreter = Interpreter.load("./models/nlu/default/chat")

# Ham test NLU
def ask_question(text):
    print(interpreter.parse(text))

ask_question("c1815 giá bao nhiêu?")
```

- Tiến hành train nlu bằng lệnh sau:

```
(envcb) /Desktop/chatbot$ python train_nlu.py
```

- Output sẽ có dạng:

```
{'intent': {'name': 'linh_kien', 'confidence': 0.3944010665984753}, 'entities': [],  
'intent_ranking': [{'name': 'linh_kien', 'confidence': 0.3944010665984753},  
{'name': 'ask_name', 'confidence': 0.1550437550070706},  
{'name': 'greet', 'confidence': 0.14596000325938815},  
{'name': 'loai_linh_kien', 'confidence': 0.0864044501054772},  
{'name': 'give_name', 'confidence': 0.06495691996489647},  
{'name': 'thank', 'confidence': 0.05978404434933791},  
{'name': 'ask_func_list', 'confidence': 0.05856452915083669},  
{'name': 'bye', 'confidence': 0.03488523156451796}], 'text': 'c1815 giá bao nhiêu?'}
```

B2) Train rasa core.

- Chúng ta tạo file **train_dialog.py (train rasa core)**:

```
(envcb) /Desktop/chatbot$ touch train_dialog.py  
(envcb) /Desktop/chatbot$ sudo nano train_dialog.py
```

- Chúng ta code trong file này như sau:

```
# Imports  
#-----  
# rasa core  
import logging  
from rasa_core import training  
from rasa_core.actions import Action  
from rasa_core.agent import Agent  
from rasa_core.domain import Domain  
from rasa_core.policies.keras_policy import KerasPolicy  
from rasa_core.policies import FallbackPolicy  
from rasa_core.policies.memoization import MemoizationPolicy  
from rasa_core.featurizers import MaxHistoryTrackerFeaturizer,  
BinarySingleStateFeaturizer  
from rasa_core.interpreter import RegexInterpreter  
from rasa_core.interpreter import RasaNLUInterpreter
```

```
# Function
#-----
def train_dialog(dialog_training_data_file, domain_file, path_to_model = 'models/dialogue'):
    logging.basicConfig(level=INFO)
    fallback = FallbackPolicy(fallback_action_name="utter_unclear", core_threshold=0.3,
nlu_threshold=0.3)

    agent = Agent(domain_file,
        policies=[MemoizationPolicy(max_history=1),KerasPolicy(epochs=200,
            batch_size=20), fallback])
    training_data = agent.load_data(dialog_training_data_file)
    agent.train(
        training_data,
        augmentation_factor=50,
        validation_split=0.2)
    agent.persist(path_to_model)

# Train
#-----
train_dialog('data/stories.md', 'data/domain.yml')
```

- Như đã đề cập ở B1 thì ở đây để đơn giản hóa chúng ta sẽ config policies bằng mã code như trên.
- Chúng ta tiến hành train rasa core:

```
(envcb) /Desktop/chatbot$ python train_dialog.py
```

- Output sẽ có dạng:

```
Epoch 196/200
141/141 [=====] - 0s 378us/sample - loss: 0.0453 - acc: 1.0000
Epoch 197/200
141/141 [=====] - 0s 438us/sample - loss: 0.0650 - acc: 0.9929
Epoch 198/200
141/141 [=====] - 0s 434us/sample - loss: 0.0539 - acc: 0.9929
Epoch 199/200
141/141 [=====] - 0s 438us/sample - loss: 0.0808 - acc: 0.9787
Epoch 200/200
141/141 [=====] - 0s 374us/sample - loss: 0.0645 - acc: 0.9858
INFO:rasa_core.policies.keras_policy:Done fitting keras policy model
INFO:rasa_core.agent:Model directory models/dialogue exists and contains old model files. All files
will be overwritten.
INFO:rasa_core.agent:Persisted model to 'root:\home\svtt\chatbot\models\dialogue'
```

- Khi tiến hành training cho chatbot ở các bước trên thì dữ liệu training được lưu vào modles.

3.4.4 Cuối cùng chúng ta sẽ tạo các file run project.

B1) Run rasa core.

- Tạo file `test_dialog.py`:

```
(envcb) /Desktop/chatbot$ touch test_dialog.py
(envcb) /Desktop/chatbot$ sudo nano test_dialog.py
```

- Chúng ta code trong file này như sau:

```
# Imports
# -----
# rasa core
import logging
import rasa_core
from rasa_core import training
from rasa_core.actions import Action
from rasa_core.agent import Agent
from rasa_core.domain import Domain
from rasa_core.policies.keras_policy import KerasPolicy
from rasa_core.policies.memoization import MemoizationPolicy
from rasa_core.featurizers import (
    MaxHistoryTrackerFeaturizer,
    BinarySingleStateFeaturizer,
)

# from rasa_core.channels.console import ConsoleInputChannel
from rasa_core.interpreter import RegexInterpreter
from rasa_core.utils import EndpointConfig
from rasa_core.interpreter import RasaNLUInterpreter
import json
from rasa_core import utils, train, run

from rasa_core.training import interactive

from rasa_core.channels.facebook import FacebookInput
```



```
def run_dialogue(serve_forever=True):
    print("run_dialogue is called\n")
    interpreter = RasaNLUIInterpreter(
        "/models/nlu/default/chat"
    ) # Phân tích cú pháp tin nhắn văn bản. Trả lại giá trị default nếu phân tích cú pháp văn
    bản không thành công.
    action_endpoint = EndpointConfig(
        url="http://localhost:5055/webhook"
    ) # Cấu hình cho một điểm cuối HTTP bên ngoài.
    agent = Agent.load(
        "/models/dialogue", interpreter=interpreter, action_endpoint=action_endpoint
    ) # Tải một mô hình tồn tại từ đường dẫn.
    print("*****run_dialogue is called*****\n")
    print("\n\n\n\n\n\n\n")
    rasa_core.run.serve_application(agent, channel="cmdline")
    #input_channel = FacebookInput(
    #    fb_verify="MyAssistant", # you need tell facebook this token, to confirm your URL
    #    fb_secret="0a4cd3ad0fd1839b863f8515d31a7c89", # your app secret
    #)
    fb_access_token="EAAKkT0UIkREBAGzVIXGF88OD9qtC859kEekvLRojjhZCd0NPh
a9PHBrG00SBJ1hDWIlgXWsdFHzHuS2YIZBoxFOxYWi4DtE6mr6gFNqmOK1FpxfY
ydD5KGePWDd9tGAkLPZAzKd6iRYDxc0J3hvjVPxEgeAruDuIyI83j0SpsfyFCh5HpUgt
U0vZCS6ksZD", # token for the page you subscribed to
    #)

    #agent.handle_channels([input_channel], 5004, serve_forever=True)
    return agent

run_dialogue()
```

- Run rasa core:

```
(envcb) /Desktop/chatbot$ python test_dialog.py
```

- Output sẽ có dạng input nhập từ cửa sổ dòng lệnh cmdline:

```
127.0.0.1 - - [2020-09-22 21:09:04] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 154 0.445664
Your input -> xin chào
Xin chào, hãy cho tôi biết tên của bạn để tôi xưng hô nhé.
127.0.0.1 - - [2020-09-22 21:09:18] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 331 0.044355
Your input -> anh là Phát
Chào anh phát. Tecapro-Telecom có thể giúp gì được anh phát ạ?
127.0.0.1 - - [2020-09-22 21:09:43] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 317 0.048084
Your input -> bạn có thể làm được những gì
Tôi có thể tìm mọi thứ có trong kho cho bạn!
127.0.0.1 - - [2020-09-22 21:11:00] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 284 0.048387
Your input -> tạm biệt
Chào tạm biệt anh phát và hẹn gặp lại!
127.0.0.1 - - [2020-09-22 21:11:13] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 278 0.048143
Your input -> █
```

- Phần kết quả này mới chỉ đang sử dụng rasa nlu đã được training và lưu trong model để chatbot thực hiện phản hồi cho người dùng chưa sử dụng đến actions của rasa.

B2) Tăng độ chính xác khi bắt sự kiện (lookup rasa)

- Để tăng độ chính xác khi bắt intent với các utter chúng ta tạo thêm file regex_intent.txt và thêm các intent mà chúng ta muốn bắt khi xảy ra các event liên quan:

```
(envcb) /Desktop/chatbot$ cd data &&touch regex_intent.txt &&sudo  
nano regex_intent.txt
```

- Thêm các intent vào regex_intent.txt:

```
transistor  
c1815  
bc337  
2n3904  
điện trở  
r100K  
r10K  
r50k
```

- Tiếp đến thêm các dòng sau vào cuối file nlu.md.

```
## lookup:regex_intent.txt  
data/regex_intent.txt
```

- Để biết thêm chi tiết về lookup, chúng ta có thể tham khảo ở link sau:
<https://blog.rasa.com/improving-entity-extraction/>

B3) Tạo và Run project dưới dạng input là text hoặc voice.

- Trên cửa sổ terminal chúng ta thực hiện bật actions bằng câu lệnh:

```
(envcb) /Desktop/chatbot$ python -m rasa_core_sdk.endpoint --actions
```

- Mở thêm cửa sổ terminal mới và thực hiện run rasa core. (ở bước này chúng ta có thể hỏi và yêu cầu chatbot truy vấn dữ liệu trong database bằng cmdline được rồi).

```
(envcb) /Desktop/chatbot$ python test_dialog.py
```

+ Mục đích của 2 việc làm trên là để bật server rasa webhook khi rasa core được run trên cổng 5005, và rasa actions trên cổng 5055 để có thể thực hiện kết nối và truyền nhận dữ liệu với thư viện hỗ trợ voice của google.

- Cài đặt các thư viện hỗ trợ nhận dạng voice (Tiếng Việt)

```
(envcb) /Desktop/chatbot$ pip install SpeechRecognition
(envcb) /Desktop/chatbot$ pip install gtts
(envcb) /Desktop/chatbot$ pip install playsound
(envcb) /Desktop/chatbot$ pip install PyAudio
```

+ Khi cài đặt gói PyAudio chúng ta sẽ thường gặp lỗi sau:

```
Running setup.py clean for pyaudio
Failed to build pyaudio
Installing collected packages: pyaudio
Running setup.py install for pyaudio ... error
ERROR: Command errored out with exit status 1:
  command: /home/huyphat/Desktop/chatbot_v1_p/chatbot/envcb/bin/python3 -u -c 'import sys, setuptools, tokenize; sys.argv[0] = '"'/tmp/pip-install-hzle2yg4/pyaudio/setup.py'"'; __file__ = '"'/tmp/pip-install-hzle2yg4/pyaudio/setup.py'"'; f=getattr(tokenize, '"'/open'"', open)(__file__);code=f.read().replace('"'/\r\n'"', '"'/\n'"'); f.close();exec(compile(code, __file__, '"'/exec'"'))' install --record /tmp/pip-record-lz2hedvx/install-record.txt --single-version-externally-managed --compile --install-headers /home/huyphat/Desktop/chatbot_v1_p/chatbot/envcb/include/site/python3.6/pyaudio
```

+ Để sửa được lỗi này chúng ta làm theo các bước sau:

```
(envcb) /Desktop/chatbot$ sudo apt-get install portaudio19-dev
python-pyaudio
(envcb) /Desktop/chatbot$ pip install PyAudio
```

- Tạo file stt_chatbot_tts.py (mở thêm terminal mới rồi thực hiện).

```
(envcb) /Desktop/chatbot$ touch stt_chatbot_tts.py &&sudo nano
stt_chatbot_tts.py
```

- Thực hiện code trong file này như sau:

```
# import library
import speech_recognition as sr
import pyaudio
import requests as re
import os
from gtts import gTTS
import playsound

n = 1
while n == 1:
    ## xử lý chuyển giọng nói sang văn bản
    #print(sr.Microphone.list_microphone_names())

    # Initialize recognizer class (for recognizing the speech)
    r = sr.Recognizer()

    # Đọc tệp âm thanh dưới dạng source
    # nghe tệp âm thanh và lưu trữ trong biến audio_text

    # with sr.AudioFile("output.wav") as source:
    with sr.Microphone() as source:
        print("Bot: Mời bạn nói: ")
        audio_text = r.listen(source, timeout=20)
        # recognize_() phương thức sẽ đưa ra lỗi yêu cầu nếu không thể truy cập được API,
        do đó sử dụng xử lý ngoại lệ
        try:
            # sử dụng nhận dạng giọng nói của Google
            text = r.recognize_google(audio_text, language="vi-VI")
            print("You: " + text)
        except:
            print("Bot: Bạn hãy nói gì đi, năn nỉ bạn đó.")
            output = gTTS("Bạn hãy nói gì đi, năn nỉ bạn đó.", lang="vi", slow=False)
            output.save("output.mp3")
            playsound.playsound('output.mp3', True)
            os.remove("output.mp3")
            audio_text = r.listen(source, timeout=20)
            try:
                # sử dụng nhận dạng giọng nói của Google
                text = r.recognize_google(audio_text, language="vi-VI")
                print("You: "+text)
            except:
                print("Bot : Xin chào, hãy cho tôi biết tên của bạn để tôi dễ xưng hô nhé.")
                text = "xin chào"
        with open("data/question.txt", "w", encoding="utf-8") as reader:
            reader.write(text + "\n")
```

```
## xử lý request api webhook rasa
with open("data/question.txt", encoding="utf-8") as f: # context manager
    content = f.readlines()
    content = [x.strip() for x in content]

for item in content:
    data = {"message": item}

resp = re.post("http://localhost:5005/webhooks/rest/webhook", json=data)
answer = resp.json()[0]["text"]

with open("data/answer.txt", "w", encoding="utf-8") as reader:
    reader.write(str(answer) + "\n")

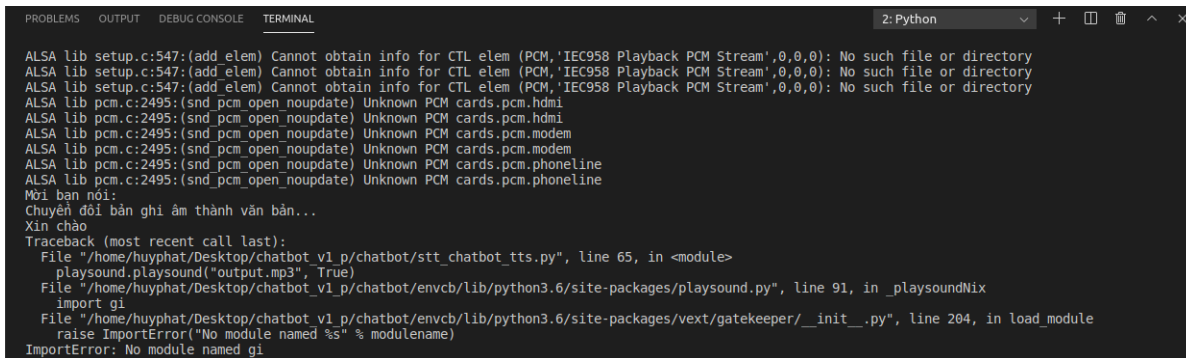
## xử lý chuyển văn bản sang giọng nói
with open("data/answer.txt", "r", encoding="utf-8") as file:
    output = gTTS(file.read(), lang="vi", slow=False)
output.save("output.mp3")
playsound.playsound("output.mp3", True)
os.remove("output.mp3")
```

- Thực hiện run stt_chatbot_tts.py (run voice chatbot).

```
(envcb) /Desktop/chatbot$ python stt_chatbot_tts.py
```

- Khi run stt_chatbot_tts.py thường thì chúng ta sẽ gặp lỗi sau :

« **error module 'gi'** »



```
ALSA lib setup.c:547:(add_elem) Cannot obtain info for CTL elem (PCM,'IEC958 Playback PCM Stream',0,0,0): No such file or directory
ALSA lib setup.c:547:(add_elem) Cannot obtain info for CTL elem (PCM,'IEC958 Playback PCM Stream',0,0,0): No such file or directory
ALSA lib setup.c:547:(add_elem) Cannot obtain info for CTL elem (PCM,'IEC958 Playback PCM Stream',0,0,0): No such file or directory
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.hdmi
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.modem
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
ALSA lib pcm.c:2495:(snd_pcm_open_noupdate) Unknown PCM cards.pcm.phoneline
Mời bạn nói:
Chuyển đổi bản ghi âm thành văn bản...
Xin chào
Traceback (most recent call last):
  File "/home/huyphat/Desktop/chatbot_v1_p/chatbot/stt_chatbot_tts.py", line 65, in <module>
    playsound.playsound("output.mp3", True)
  File "/home/huyphat/Desktop/chatbot_v1_p/chatbot/envcb/lib/python3.6/site-packages/playsound.py", line 91, in _playsoundNix
    import gi
  File "/home/huyphat/Desktop/chatbot_v1_p/chatbot/envcb/lib/python3.6/site-packages/vext/gatekeeper/_init_.py", line 204, in load_module
    raise ImportError("No module named %s" % modulename)
ImportError: No module named gi
```

- + Cách sửa lỗi trên :

```
(envcb) /Desktop/chatbot$ sudo apt-get install python3-gi
(envcb) /Desktop/chatbot$ pip install vext
(envcb) /Desktop/chatbot$ pip install vext.gi
```

- Sau các bước sửa lỗi, cuối cùng chúng ta sẽ chạy lại file stt_chatbot_tts.py :

```
(envcb) /Desktop/chatbot$ python stt_chatbot_tts.py
```

- Gắn tai phone và micro vào máy đồng thời máy phải kết nối internet để nói chuyện với chatbot . (Chúng ta đang mượn giọng nói chị assistant của google)
- Output sẽ có dạng :

Bot: Bạn hãy nói gì đi, năn nỉ bạn đó.

You: anh là phát

Bot: Chào anh phát. Tecapro-Telecom có thể giúp gì được anh phát ạ?

Bot: Mời bạn nói:

You: Bạn biết gì

Bot: Tôi có thể tìm mọi thứ có trong kho cho bạn!

Bot: Mời bạn nói:

You: c1815 giá bao nhiêu

Bot: c1815 có giá: 100

Bot: Mời bạn nói:

You: transistor

Bot: transistor: c1815

Bot: Mời bạn nói:

You: cảm ơn

Bot: Cảm ơn bạn!

Bot: Mời bạn nói:

You: tạm biệt

Bot: Chào tạm biệt anh phát và hẹn gặp lại!

Bot: Mời bạn nói:

Bot: Bạn hãy nói gì đi, năn nỉ bạn đó.

Bot : Xin chào, hãy cho tôi biết tên của bạn để tôi xưng hô nhé.

Bot: Xin chào, hãy cho tôi biết tên của bạn để tôi xưng hô nhé.

Bot: Mời bạn nói:

□

⇒ Như vậy là chúng ta đã thành công xây dựng một chatbot basic trong việc hỗ trợ tìm kiếm đồ trong kho.

3.4.5 Đưa chatbot-assistant lên docker-linux

- Các bước cài đặt docker trên ubuntu18.04

```
:~$ sudo apt update
:~$ sudo apt install apt-transport-https ca-certificates curl
software-properties-common
:~$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo apt-key add -
:~$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
:~$ sudo apt update
:~$ apt-cache policy docker-ce
:~$ sudo apt install docker-ce
:~$ sudo systemctl status docker
```

- Output phải tương tự như sau, cho thấy dịch vụ đang hoạt động và đang chạy:

```
huyphat@ubuntu:~$ sudo systemctl status docker
[sudo] password for huyphat:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Thu 2020-09-24 02:34:19 PDT; 4 days ago
     Docs: https://docs.docker.com
   Main PID: 50864 (dockerd)
    Tasks: 21
   CGroup: /system.slice/docker.service
           └─50864 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
             └─82381 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 5001 -container-ip 172.17.0.3 -container-port 5001

Sep 25 23:48:55 ubuntu dockerd[50864]: time="2020-09-25T23:48:55.160242923-07:00" level=error msg="attach failed with error: error attaching s
Sep 25 23:59:29 ubuntu dockerd[50864]: time="2020-09-25T23:59:29.840750936-07:00" level=info msg="Container 245248b0dd9c0925a5aa32a84f5e1191d8
Sep 25 23:59:30 ubuntu dockerd[50864]: time="2020-09-25T23:59:30.140248621-07:00" level=info msg="ignoring event" module=libcontainerd namespa
Sep 26 00:05:44 ubuntu dockerd[50864]: time="2020-09-26T00:05:44.818523824-07:00" level=info msg="Container fd0fd7ae61e53f0fa290458d6840eabe75
Sep 26 00:05:45 ubuntu dockerd[50864]: time="2020-09-26T00:05:45.133197073-07:00" level=error msg="attach failed with error: error attaching s
Sep 26 00:05:45 ubuntu dockerd[50864]: time="2020-09-26T00:05:45.133206596-07:00" level=info msg="ignoring event" module=libcontainerd namespa
Sep 26 00:06:42 ubuntu dockerd[50864]: time="2020-09-26T00:06:42.158848668-07:00" level=info msg="Container fd0fd7ae61e53f0fa290458d6840eabe75
Sep 26 00:06:42 ubuntu dockerd[50864]: time="2020-09-26T00:06:42.559553363-07:00" level=info msg="ignoring event" module=libcontainerd namespa
Sep 26 00:53:45 ubuntu dockerd[50864]: time="2020-09-26T00:53:45.788285541-07:00" level=info msg="Container fd0fd7ae61e53f0fa290458d6840eabe75
Sep 26 00:53:46 ubuntu dockerd[50864]: time="2020-09-26T00:53:46.022162039-07:00" level=info msg="ignoring event" module=libcontainerd namespa
lines 1-20/20 (END)
```

- Tiến hành dowload images ubuntu18.04 về docker
 - + Kiểm tra các gói images ubuntu18.04 hiện có trên docker hub.

```
:~$ sudo docker search ubuntu18.04
```

+ Output sẽ có dạng :

```
huyphat@ubuntu:~$ sudo docker search ubuntu18.04
```

NAME	DESCRIPTION	STARS
AUTOMATED waffleimage/ubuntu18.04		1
ecpe4s/ubuntu18.04-spack		1
devcafe/ubuntu18.04-gcc7.3.0-openmpi2.1.0-nkl2017.4.239 [OK]	Ubuntu 18.04 image with GCC 7.3.0, OpenMPI 2...	1
duruo850/ubuntu18.04-python3.6 [OK]	ubuntu18.04-python3.6	1
jjuanrivvera99/ubuntu18.04-apache2-php7.2-oracleclient12.2	A web development environment on linux with ...	1

- Download images ubuntu18.04

```
:~$ sudo docker pull duruo850/ubuntu18.04-python3.6
```

- Kiểm tra images đã tồn tại chưa :

```
:~$ sudo docker images
```

+ Output chúng ta thấy images ” duruo850/ubuntu18.04-python3.6” trong docker thì đã thành công:

```
huyphat@ubuntu:~$ sudo docker images
```

[sudo] password for huyphat:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
paschmann/rasa-ui	3.0.3	10e231b7ffe6	12 months ago	308MB
duruo850/ubuntu18.04-python3.6	latest	a7beef9d2712	23 months ago	914MB

```
huyphat@ubuntu:~$ █
```

- Tạo container với images duruo850/ubuntu18.04-python3.6:

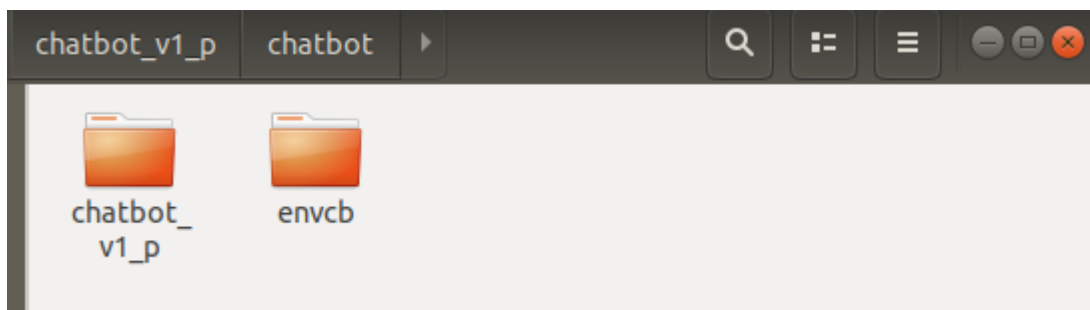
```
:~$ sudo docker run -it --name ubun duruo850/ubuntu18.04-  
python3.6:latest
```

+ Lệnh trên có nghĩa : tạo một container có tên ubun với images duruo850/ubuntu18.04-python3.6.

+ Sau khi chạy lệnh trên trong cửa sổ dòng lệnh sẽ nhảy vào user ‘root’ của ubuntu container ‘ubun’:

```
huyphat@ubuntu:~$ sudo docker run -it --name ubun duruo850/ubuntu18.04-python3.6:latest  
root@92af9d479725:/usr/local/bin# █
```


- Khi này chúng ta đang đứng ở quyền root container, và khi chạy lệnh trên docker sẽ cấp cho new container một ID mới, như ở đây ID của container là 92af9d479725.
- Cut thư mục môi trường ảo trong thư mục chatbot ra bên ngoài trước khi thực hiện thao tác copy.



- Mở terminal mới (ta tạm gọi là terminal máy host) ta thực hiện copy thư mục chatbot từ máy host vào container 'ubun' đang chạy :

```
huyphat@ubuntu:~/Desktop$ sudo docker cp chatbot_v1_p/ ubun:/home/chatbot_v1_p
huyphat@ubuntu:~/Desktop$
```

- Quay lại terminal container 'ubun' :
 - + Thực hiện kiểm tra thư mục chatbot đã tồn tại chưa :

```
root@92af9d479725:/home# ls
chatbot_v1_p
root@92af9d479725:/home#
```

+ Thực hiện cài đặt các gói thư viện cần thiết :

```
root@id_container:/home# apt update
root@id_container:/home# pip3 install --upgrade pip
root@id_container:/home# pip install --upgrade pip setuptools wheel
root@id_container:/home# pip install rasa_core rasa_nlu rasa_core_sdk
feedparser spacy sklearn_crfsuite
root@id_container:/home# python -m spacy download en
root@id_container:/home# pip install tornado
root@id_container:/home# pip install --pre requests
root@id_container:/home# pip install pyvi
root@id_container:/home# pip install bs4
```

+ Vào thư mục chatbot, train model và start server webhook rasa actions chatbot (lưu ý không được tắt terminal start actions server khi run chatbot):

```
root@id_container:/home# cd chatbot_v1_p/chatbot
root@id_container:/home/chatbot_v1_p/chatbot# python train_nlu.py
root@id_container:/home/chatbot_v1_p/chatbot# python train_dialog.py
root@id_container:/home/chatbot_v1_p/chatbot# python -m
rasa_core_sdk.endpoint --actions actions
```

+ Output :

```
root@c964d5e9abb9:/home/chatbot_v1_p/chatbot# python -m rasa_core_sdk.endpoint --actions actions
2020-09-30 07:21:13 INFO      __main__ - Starting action endpoint server...
Database created and Successfully Connected to SQLite
2020-09-30 07:21:14 INFO      rasa_core_sdk.executor - Registered function for 'action_save_cust_info'.
2020-09-30 07:21:14 INFO      rasa_core_sdk.executor - Registered function for 'action_save_mobile_no'.
2020-09-30 07:21:14 INFO      rasa_core_sdk.executor - Registered function for 'action_reset_slot'.
2020-09-30 07:21:14 INFO      rasa_core_sdk.executor - Registered function for 'action_custom_loai_linh_kien'.
2020-09-30 07:21:14 INFO      rasa_core_sdk.executor - Registered function for 'action_custom_linh_kien'.
2020-09-30 07:21:14 INFO      __main__ - Action endpoint is up and running. on ('0.0.0.0', 5055)
```

+ Run chatbot trên cmdline docker (new terminal) :

```
root@id_container:/home/chatbot_v1_p/chatbot# export
PYTHONIOENCODING=utf-8
root@id_container:/home/chatbot_v1_p/chatbot# python test_dialog.py
```

+ Output :

```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input -> bạn ơi
Xin chào, hãy cho tôi biết tên của bạn để tôi dễ xưng hô nhé.
127.0.0.1 - - [2020-09-30 07:24:22] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 331 0.499397
Your input -> anh là SỬ
Chào anh SỬ. Tecapro-Telecom có thể giúp gì được anh sử ạ?
127.0.0.1 - - [2020-09-30 07:24:56] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 313 0.049418
Your input -> bạn tên gì
Tôi tên Bot họ Chat, luôn vui vẻ và nhiệt tình, bạn cần tôi giúp gì không?
127.0.0.1 - - [2020-09-30 07:25:23] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 344 0.057986
Your input -> bạn có thể làm được những gì
Tôi có thể tìm mọi thứ có trong kho cho bạn!
127.0.0.1 - - [2020-09-30 07:25:44] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 284 0.059319
Your input -> c1815 giá bao nhiêu
c1815 có giá: 100
127.0.0.1 - - [2020-09-30 07:26:11] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 227 0.090520
Your input -> bc337 giá bao nhiêu
bc337 có giá: 150
127.0.0.1 - - [2020-09-30 07:26:27] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 227 0.089059
Your input -> 2n3904 giá bao nhiêu
2n3904 có giá: 200
127.0.0.1 - - [2020-09-30 07:26:41] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 228 0.065277
Your input -> điện trở trong kho có những loại nào
Hiện tại trong kho không có loại linh kiện này.
127.0.0.1 - - [2020-09-30 07:27:02] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 282 0.065145
Your input -> thanks
Cảm ơn bạn!
127.0.0.1 - - [2020-09-30 07:27:46] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 226 0.065319
Your input -> tạm biệt
Kính chào tạm biệt và chúc quý khách một ngày tốt lành!
127.0.0.1 - - [2020-09-30 07:28:10] "POST /webhooks/rest/webhook?stream=true&token= HTTP/1.1" 200 315 0.079059
Your input -> █
```

+ Đóng gói container ra images docker.

```
~$ sudo docker commit ubun ubuntu/chatbot
```

Chú thích :

ubun : tên container hiện tại muốn commit

ubuntu/chatbot : tên images mới

+ Output :

```
huyphat@ubuntu:~$ docker commit ubun ubuntu/chatbot
sha256:a4c9cbf63b43fb42230c0ebc31ad9d952959e4666527c112587fa893cf71dc1b
huyphat@ubuntu:~$ █
```

+ Kiểm tra images mới đã tồn tại chưa :

```
huyphat@ubuntu:~$ sudo docker images
[sudo] password for huyphat:
REPOSITORY          TAG          IMAGE ID          CREATED          SIZE
ubuntu/chatbot      latest      a4c9cbf63b43      2 minutes ago   2.12GB
paschmann/rasa-ui   3.0.3       10e231b7ffe6      12 months ago   308MB
duruo850/ubuntu18.04-python3.6 latest      a7beef9d2712      23 months ago   914MB
huyphat@ubuntu:~$ █
```

⇒ Như vậy chúng ta đã thấy images ubuntu/chatbot đã được tạo mới từ container chatbot 'ubun'. Bây giờ chúng ta có thể đẩy images này lên 'docker hub' hoặc nén ra file.tar để chuyển giao công nghệ.

+ Save image thành file .tar theo cú pháp

Cú pháp : `docker save {image_name} > {/host_path/new_image.tar}`

```
:~$ sudo docker save ubuntu/chatbot > /home/huyphat/u_chatbot.tar
```

+ Output :

```
huyphat@ubuntu:~$ sudo docker save ubuntu/chatbot > /home/huyphat/u_chatbot.tar
[sudo] password for huyphat:
huyphat@ubuntu:~$ ls
Desktop  Documents  examples.desktop  npm-debug.log  Public  snap  u_chatbot.tar
DockerCbRs  Downloads  Music  Pictures  rtlwifi_new  Templates  Videos
huyphat@ubuntu:~$
```

⇒ Như vậy là chúng ta đã có gói images chứa chatbot-assistent, chúng ta có thể đem images này đi cài trên nhiều máy khác nhau mà không sợ xung đột phần mềm hay lỗi phiên bản.

- Mục đích của việc sử dụng docker là để chuyển giao công nghệ nhanh hiệu quả, và để nâng cấp các version mới thuận tiện dễ dàng hơn.

+ Để có thể sử dụng images của chúng ta mới tạo trên một computer có cài đặt docker bất kỳ ta run theo cú pháp sau :

Cú pháp : `sudo docker load < file.tar`

→ Khi run lệnh trên chúng ta sẽ có image docker, được thực thi từ file.tar

- Chúng ta có thể tham khảo thêm về docker và các cú pháp lệnh ở linh dưới :

+ Document docker :

<https://docker-ghicheck.readthedocs.io/en/latest/lenh-docker/>

<https://docs.docker.com/engine/reference/commandline/load/>

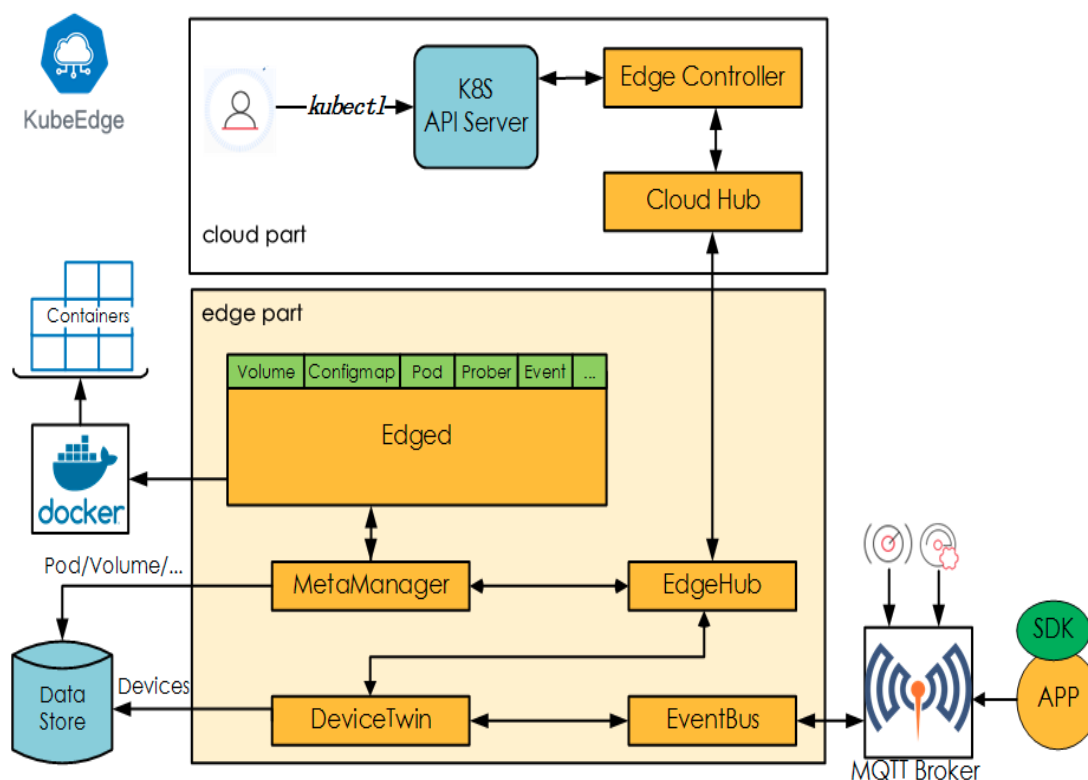
<https://docs.docker.com/engine/reference/commandline/save/>

4. Tìm hiểu về KubeEdge- mô hình Edge Computing.

Edge computing nói đơn giản thì mô hình công nghệ này các Data center sẽ nằm ở vị trí trung tâm, các thiết bị Getway (như Switch, router,..) sẽ nằm ngoài rìa. Các IOT khi có Data, sẽ gửi đến các thiết bị nằm ở rìa, sau đó các Processor(chip) trong mạng lưới sẽ phân tích và xử lý Data này.

Cloud Computing có điểm yếu dễ thấy là tốc độ sẽ không đảm bảo khi phải truyền dữ liệu đi xa. Vì vậy Edge Computing đã ra đời để giải quyết vấn đề này, bằng cách lưu trữ và xử lý thông tin quan trọng ngay tại một trung tâm dữ liệu nhỏ trước khi nó được gửi tới trung tâm dữ liệu chính.

- *KubeEdge là nền tảng điện toán cận biên Kubernetes đầu tiên với cả hai thành phần Edge và Cloud mã nguồn mở. Với các giao diện dựa trên Kubernetes nhất quán và có thể mở rộng, KubeEdge cho phép điều phối và quản lý các cụm biên tương tự như cách Kubernetes quản lý trên cloud.*
- *KubeEdge cung cấp một nền tảng điện toán biên được đóng gói, vốn có khả năng mở rộng. Vì nó được mô-đun hóa và tối ưu hóa, nó rất nhẹ (66 MB foot print và ~ 30 MB bộ nhớ hoạt động) và có thể được triển khai trên các thiết bị cấu hình thấp. Tương tự, node biên có thể có kiến trúc phần cứng khác nhau và với các cấu hình phần cứng khác nhau. Đối với việc kết nối thiết bị, nó có thể hỗ trợ nhiều giao thức và nó sử dụng giao tiếp dựa trên chuẩn MQTT. Điều này giúp mở rộng các cụm biên với các node và thiết bị mới một cách hiệu quả.*
- *Nguyên lý kiến trúc cốt lõi cho KubeEdge là xây dựng các giao diện phù hợp với Kubernetes, có thể là ở trên cloud hoặc cận biên.*



Edged: Quản lý các ứng dụng được đóng gói tại Edge.

EdgeHub: Mô-đun giao diện truyền thông tại Edge. Nó là một máy khách web socket chịu trách nhiệm tương tác với Cloud Service để tính toán biên.

CloudHub: Mô-đun giao diện truyền thông tại Cloud. Một máy chủ websocket chịu trách nhiệm theo dõi các thay đổi ở phía đám mây, lưu trữ và gửi tin nhắn đến EdgeHub.

EdgeContoder: Quản lý các node Edge. Nó là một bộ điều khiển Kubernetes mở rộng, quản lý các node biên và siêu dữ liệu pods (pods metadata) để dữ liệu có thể được nhắm mục tiêu đến một node biên cụ thể.

EventBus: Xử lý truyền thông cận biên bên trong bằng MQTT. Nó là một máy khách MQTT để tương tác với các máy chủ MQTT (mosquitto), cung cấp khả năng publish và subscribe cho các thành phần khác.

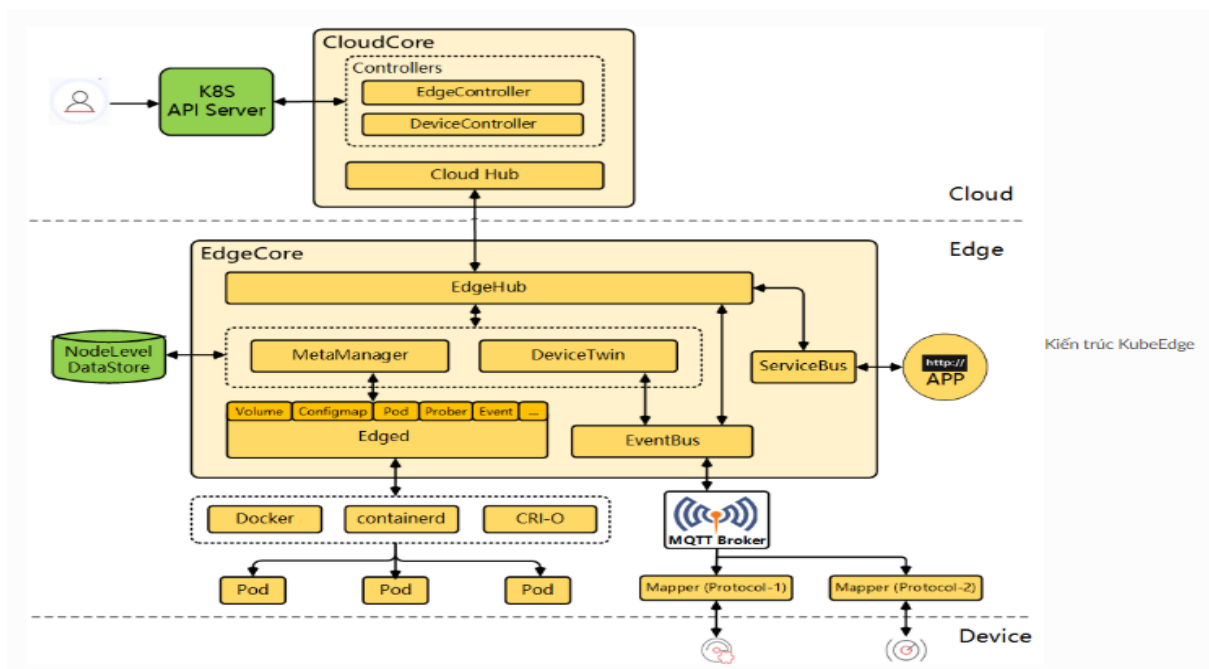
DeviceTwin: Nó là *software mirror* cho các thiết bị xử lý siêu dữ liệu của thiết bị. Mô-đun này giúp xử lý trạng thái thiết bị và đồng bộ hóa tương tự với đám mây. Nó cũng cung cấp các giao diện truy vấn cho các ứng dụng, vì nó giao tiếp với một cơ sở dữ liệu *lightweight* (SQLite).

MetaManager: Nó quản lý siêu dữ liệu tại node biên. Đây là bộ xử lý bản tin (*message*) giữa *edged* và *edgehub*. Nó cũng chịu trách nhiệm lưu trữ / truy xuất siêu dữ liệu đến / từ cơ sở dữ liệu *lightweight* (SQLite).

➤ Bắt đầu với KubeEdge

KubeEdge là một hệ thống mã nguồn mở mở rộng khả năng điều phối ứng dụng và quản lý thiết bị trong vùng chứa riêng cho các máy chủ tại Edge. Nó được xây dựng dựa trên Kubernetes và cung cấp hỗ trợ cơ sở hạ tầng cốt lõi cho mạng, triển khai ứng dụng và đồng bộ hóa siêu dữ liệu giữa đám mây và biên. Nó cũng hỗ trợ MQTT và cho phép các nhà phát triển tạo logic tùy chỉnh và cho phép giao tiếp với thiết bị hạn chế tài nguyên tại Edge. KubeEdge bao gồm một phần đám mây và một phần cạnh. Cả hai phần cạnh và đám mây hiện đều có nguồn mở.

- Kiến trúc KubeEdge gồm 3 phần: Cloud, Edge, và Device.



- Để làm việc với kubeEdge chúng ta cần phải thực hiện theo đúng các quy trình sau :

B1) Yêu cầu cài đặt.

+ Phần Cloud : Kubernetes

Link : <https://kubernetes.io/docs/setup/>

- + Phần Edge : Container runtimes, KubeEdge hỗ trợ: Docker, Containerd, Cri-o, Virtlet, MQTT Server(Optional)

Link: <https://www.docker.com/>
<https://github.com/containerd/containerd>
<https://cri-o.io/>
<https://docs.virtlet.cloud/>
<https://mosquitto.org/>

B2) Xây dựng kubeEdge từ source.

- + Chúng ta tiến hành Build KubeEdge theo Document ở link sau:

Link: <https://docs.kubeedge.io/en/latest/setup/build.html>

B3) Deploying KubeEdge

- + Deploying locally thực hiện theo link:

<https://docs.kubeedge.io/en/latest/setup/local.html>

- + Deploying using k8s thực hiện theo link:

<https://docs.kubeedge.io/en/latest/setup/k8s.html>

CHƯƠNG 3 : KẾT QUẢ ĐẠT ĐƯỢC

I. TRÌNH BÀY VÀ ĐÁNH GIÁ CÁC KẾT QUẢ THỰC TẬP

Qua đợt thực tập này em cảm thấy bản thân mình học hỏi thêm được nhiều kiến thức về ngôn ngữ lập trình Python và quy trình xây dựng chatbot-assistant sử dụng Machine Learning, có cách nhìn mới về mô hình KubeEdge (Edge-Computing) hoạt động như thế nào, và qua đó em có cái nhìn khái quát, bao quát hơn để có thể hiểu và làm được nhiều ứng dụng hơn từ nó. Sau hai tháng rưỡi mà em đã học hỏi và tìm tòi về nó, và cũng như được làm việc và được cọ xát với môi trường thực tế, làm việc chuyên nghiệp của công ty tecapro, và được giao cho đề án xây dựng hệ thống chatbot-assistant hỗ trợ nhân viên trong việc tìm kiếm đồ trong kho, em cảm thấy kiến thức có mình có được nhiều hơn và có nhiều kinh nghiệm thực tế hữu ích hơn để ứng dụng và làm được các sản phẩm giúp ích cho đời sống thực tế. Từ đó, em còn được làm quen với phong cách, tác phong làm việc, môi trường làm việc chuyên nghiệp, giao tiếp với các anh chị đang làm việc trong ngành, biết thêm về yêu cầu công việc. Học hỏi được nhiều hơn từ kinh nghiệm từ các anh chị đi trước và hiểu rõ hơn bản thân của mình hơn để chuẩn bị việc bổ sung kiến thức cho công việc sau này.

II. ƯU ĐIỂM

Em đã có thể tự xây dựng một chatbot-assistant với đầy đủ tính năng cơ bản trong việc hỗ trợ con người, có thể thuận tiện trong việc nâng cấp các version mới và chuyên giao công nghệ. Chatbot-assistant đang là lựa chọn hàng đầu của các doanh nghiệp lớn nhỏ, ví dụ như trong ngành dịch vụ khách hàng, hay cụ thể hơn trong việc hỗ trợ quyết định của người mua, việc sử dụng chatbot đã thay đổi mạnh mẽ trải nghiệm của khách hàng. Mục đích của việc sử dụng chatbot trong kinh doanh là hỗ trợ doanh nghiệp kết giao được với không chỉ khách hàng hiện tại, mà còn cả những khách hàng tiềm năng. Ngoài ra Chatbot-assistant còn được ứng dụng nhiều trong các hệ thống IOT, marketing, hay dự báo thời tiết, logistics, phân luồng đơn hàng, quảng đường,

Và qua đó em nhận thấy rằng nó có thể thay thế con người làm việc ở một mức độ nào đó để cho công việc được thực hiện một cách nhanh chóng và gọn lẹ hơn, đồng thời việc xử lý cũng nhanh hơn rất nhiều để có thể tiết kiệm thời gian thực hiện công việc một cách tối ưu và hiệu quả nhất có thể. Version chatbot rasa hiện tại có thể được nâng cấp lên các version cao hơn của rasa, đảm bảo có thể cập nhật các tính năng mới mà RASA hỗ trợ trong tương lai.

Có cách hiểu về quy trình các bước hoạt động của mô hình KubeEdge (Edge Computing).

III. NHƯỢC ĐIỂM

Phiên bản Chatbot – assistant này chưa được hoàn thiện, chưa tối ưu hóa mã code, phần cơ sở dữ liệu chưa đầy đủ nên phần xử lý backend cũng chưa được hoàn thiện. Sản phẩm này là sản phẩm demo và chưa thể sử dụng trong thực tế.

Chưa thể đi sâu phân tích thuật toán mà Rasa framework sử dụng để triển khai những modle có tính ứng dụng và độ chính xác cao hơn. Chatbot chưa có khả năng tự học, để làm được điều này cần xây dựng riêng cho chatbot modle để thực hiện nhiệm vụ này, rasa hay bất kỳ framework open source nào cũng không hỗ trợ tính năng này

Chưa thể cài đặt cũng như xây dựng mô hình KubeEdge.

C. KẾT LUẬN

Qua thời gian 2 tháng rưỡi thực tập và việc tiếp xúc với thực tế cùng với sự giúp đỡ của Thầy Cô và sự chỉ dẫn tận tình của các anh chị ở công ty TECAPRO Telecom, chúng em đã tiếp thu được nhiều kiến thức thực tế, đặc biệt là yêu cầu công việc trong tương lai, tác phong làm việc chuyên nghiệp giao tiếp với các anh chị đang làm việc trong ngành, biết thêm về yêu cầu công việc. Học hỏi được nhiều hơn từ kinh nghiệm từ các anh chị đi trước và hiểu rõ hơn bản thân của mình hơn để chuẩn bị cho việc bổ sung kiến thức cho công việc sau này. Qua đây chúng em xin chân thành cảm ơn Ban giám hiệu, các thầy cô trong khoa Công Nghệ - Điện Tử, bộ môn máy tính, Trường Đại Học Công Nghiệp TP.Hồ Chí Minh và đặc biệt là các anh chị trong công ty TECAPRO Telecom đã tạo điều kiện, môi trường làm việc và tận tình giúp đỡ chỉ bảo cho em. Em xin chân thành cảm ơn !

TÀI LIỆU THAM KHẢO

[1] Rasa core 0.14.5

<https://pypi.org/project/rasa-core/>

[2] Build a Conversational Chatbot with Rasa Stack and Python— Rasa NLU

<https://medium.com/@itsromiljain/build-a-conversational-chatbot-with-rasa-stack-and-python-rasa-nlu-b79dfbe59491>

[3] Deploy your chatbot on Slack

<https://medium.com/@itsromiljain/3-deploy-your-chatbot-on-slack-1a0e8390f66b>

[4] Hiểu Rasa qua quy trình xây dựng một chatbot giúp bạn trả lời câu hỏi: "Hôm nay ăn gì?"

<https://viblo.asia/p/hieu-rasa-qua-quy-trinh-xay-dung-mot-chatbot-giup-ban-tra-loi-cau-hoi-hom-nay-an-gi-WAyK8P4pKxX>

[5] Cách kết nối Chatwork với Rasa, và 5 phút mặc niệm latency trên trời.

<https://viblo.asia/p/cach-ket-noi-chatwork-voi-rasa-va-5-phut-mac-niem-latency-tren-troi-924IJb0IPM>

[6] Rasa docs Mattermost

<https://rasa.com/docs/rasa/connectors/mattermost>

[7] Rasa Actions

<https://legacy-docs.rasa.com/docs/core/run-code-in-custom-actions/#custom-actions/>

[8] Sử dụng Rasa Custom Actions xử lý cuộc hội thoại cho chatbot

<https://viblo.asia/p/su-dung-rasa-custom-actions-xu-ly-cuoc-hoi-thoai-cho-chatbot-bJzKmOywl9N>

[9] Database SQLite cơ bản

<https://hoclaptrinh.vn/tutorial/hoc-sqlite-co-ban-va-nang-cao/tao-database-trong-sqlite>

[10] Truy vấn dữ liệu trong SQL

<http://giasutinhoc.vn/database/co-so-du-lieu/truy-van-du-lieu-trong-sql-bai-5-2/>

[11] Tập tành rasa chatbot

<https://viblo.asia/p/tap-tanh-lam-rasa-chatbot-gAm5y8Nwldb>

[12] Rasa core - khi nào cần thiết ?

<https://viblo.asia/p/rasa-core-khi-nao-thi-can-thiet-bWrZn7Qmlxw>

[13] SQLite Python

https://www.tutorialspoint.com/sqlite/sqlite_python.htm

[14] DB-API 2.0 interface for SQLite databases

<https://docs.python.org/2/library/sqlite3.html>

[15] Xây dựng chatbot Messenger cập nhật thời khóa biểu cho sinh viên (Phần 4) - Xử lý logic chatbot khi được hỏi các thông tin về học phần

<https://viblo.asia/p/xay-dung-chatbot-messenger-cap-nhat-thoi-khoa-bieu-cho-sinh-vien-phan-4-xu-li-logic-chatbot-khi-duoc-hoi-cac-thong-tin-ve-hoc-phan-Qbq5Q0jJID8>

[16] Getting Started with MySQL in Python

https://www.datacamp.com/community/tutorials/mysql-python?utm_source=adwords_ppc&utm_campaignid=1455363063&utm_adgroupid=65083631748&utm_device=c&utm_keyword=&utm_matchtype=b&utm_network=g&utm_adposition=&utm_creative=278443377095&utm_targetid=aud-392016246653:dsa-429603003980&utm_loc_interest_ms=&utm_loc_physical_ms=1028581&gclid=EAiaIQobChMIpaCX4P_V6wIVYtVMAh0BmA9SEAAAYASAAEgJUzvD_BwE

[17] Document RASA CORE 0.14.5

<https://legacy-docs.rasa.com/docs/core/0.14.5/>

[18] Unable to connect to localhost port 5005 in WSL (rasa chatbot)

https://www.reddit.com/r/bashonubuntuonwindows/comments/bdbn0b/unable_to_connect_to_localhost_port_5005_in_wsl/

[19] Python Requests post() Method

https://www.w3schools.com/python/ref_requests_post.asp

[20] Xây dựng trợ lý ảo tiếng việt đơn giản bằng python

<https://quantraai.com/xay-dung-tro-ly-ao-tieng-viet-bang-ngon-ngu-python-co-ban/>

[21] What would cause the “gi” module to be missing from Python?

<https://askubuntu.com/questions/80448/what-would-cause-the-gi-module-to-be-missing-from-python>

[22] Convert MP3 to WAV

<https://pythonbasics.org/convert-mp3-to-wav/>

[23] PYTHON PROGRAMMING

https://pythonprogramming.altervista.org/from-text-to-mp3-with-gtts-reading-external-file/?doing_wp_cron=1601534386.0061759948730468750000

[24] docker load

<https://docs.docker.com/engine/reference/commandline/load/>

[25] docker save

<https://docs.docker.com/engine/reference/commandline/save/>

[26] Tìm hiểu về Docker - Phần 4 - Tạo image từ Dockerfile

<https://blog.cloud365.vn/container/tim-hieu-docker-phan-4/>

[27] Tập hợp các lệnh sử dụng trong Docker

<https://docker-ghicheck.readthedocs.io/en/latest/lenh-docker/>

[28] Làm việc với Docker Data Volumes

<https://viblo.asia/p/lam-viec-voi-docker-data-volumes-EoDGQOaNkbV>

[29] Chia sẻ dữ liệu giữa Docker Host và Container

<https://xuanthulab.net/chia-se-du-lieu-giua-docker-host-va-container.html>

[30] Kubernetes Documentation

<https://kubernetes.io/docs/setup/>

[31] KubeEdge, a Kubernetes Native Edge Computing Framework

<https://kubernetes.io/blog/2019/03/19/kubeedge-k8s-based-edge-intro/>