

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



**BÁO CÁO THỰC TẬP THỰC TẾ
Mã môn: CT455**

**Đề tài:
XÂY DỰNG CHATBOT CHO DỊCH VỤ CÔNG SỐC TRẮNG
DỰA TRÊN NỀN TẢNG RASA CHATBOT**

Đơn vị thực tập: Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC)

**Sinh viên: Đặng Hiếu Nghĩa
MSSV: B1509936
Khóa 41**

Cần Thơ, 4/2020

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



BÁO CÁO THỰC TẬP THỰC TẾ

Đề tài:

**XÂY DỰNG CHATBOT CHO DỊCH VỤ CÔNG SÓC TRĂNG
DỰA TRÊN NỀN TẢNG RASA CHATBOT**

Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC)

Người hướng dẫn: Ts. Trần Hoàng Việt

Giảng viên hướng dẫn: Ts. Lưu Tiến Đạo

Sinh viên thực hiện: Đặng Hiếu Nghĩa – B1509936

Cần Thơ, 4/2020

[illegible]

LỜI CẢM ƠN

Lời đầu tiên em xin bày tỏ lòng cảm ơn chân thành đến Trung tâm Công nghệ Phần mềm Đại học Cần Thơ đã tạo điều kiện cho em được thực tập và học hỏi trong suốt khoảng thời gian vừa qua. Đặc biệt, em xin gửi lời cảm ơn chân thành đến **Ts. Trần Hoàng Việt** đã lựa chọn và tận tình hướng dẫn cho em thực hiện đề tài này. Với sự quan tâm, dạy dỗ và chỉ bảo chân tình chu đáo của quý thầy, đã giúp em có được những kiến thức vô cùng quý giá, giúp em hiểu được giá trị của việc nghiên cứu và tìm hiểu những kiến thức mới cũng như giúp em rèn luyện được một tác phong công nghiệp trong học tập và làm việc.

Với kiến thức còn hạn chế, cũng như thiếu kinh nghiệm trong làm việc nên báo cáo này không thể tránh được những thiếu sót. Em rất mong nhận được sự chỉ bảo, đóng góp ý kiến của quý thầy cô để em có điều kiện bổ sung và rút kinh nghiệm để có thể áp dụng vào cho công việc sau này.

Em xin chân thành cảm ơn!

Cần Thơ, ngày tháng năm 2020

Người viết

Đặng Hiếu Nghĩa

[illegible]

MỤC LỤC

DANH MỤC HÌNH	8
DANH MỤC BẢNG	9
CHƯƠNG I: TÌM HIỂU CƠ QUAN THỰC TẬP.....	1
1. Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC).....	1
1.1 Tổng quan	1
1.2 Cơ cấu tổ chức	2
1.3 Địa chỉ liên hệ	3
1.4 Thành tựu đạt được	3
CHƯƠNG II: NỘI DUNG VÀ PHƯƠNG PHÁP THỰC HIỆN.....	4
1. Nội dung công việc thực tập.....	4
2. Phương pháp, thời gian thực hiện	4
CHƯƠNG III: NỘI DUNG THỰC TẬP	5
1. Tìm hiểu về chatbot và Rasa	5
2. Cài đặt Rasa.....	7
2.1 Yêu cầu của hệ thống	7
2.2 Cài đặt	7
3. Cấu trúc của Rasa	8
3.1 Các thành phần cơ bản của Rasa	8
3.2 Các khái niệm cơ bản của Rasa	10
4. Xây dựng Chatbot cho dịch vụ công Sóc Trăng	11
4.1 Cấu hình pipeline xử lý ngôn ngữ tiếng Việt.....	11
4.2 Tạo bộ dữ liệu (database).....	12
4.3 Xây dựng chức năng thống kê số lượng hồ sơ của một đơn vị.....	13
4.4 Cải tiến chatbot	18
4.5 Đưa chatbot vào website	20

CHƯƠNG 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN	22
1. Kết quả và kinh nghiệm học tập qua thời gian thực tập thực tế.....	22
2. Những kiến thức lý thuyết được củng cố	22
3. Những điểm hạn chế.....	22
4. Hướng phát triển.....	22

DANH MỤC HÌNH

Hình 1: Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC)	1
Hình 2: Cơ cấu tổ chức	2
Hình 3: Sơ đồ các bước xây dựng chatbot	6

DANH MỤC BẢNG

Bảng 1: Bảng công việc thực hiện.....	4
Bảng 2: Bảng mô tả dữ liệu dùng làm cơ sở dữ liệu	13

CHƯƠNG I: TÌM HIỂU CƠ QUAN THỰC TẬP

1. Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC)

1.1 Tổng quan



Hình 1: Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC)

Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (Tiếng Anh: Cantho University Software Center - CUSC) được thành lập ngày 29/03/2001 theo quyết định số 1574/QĐ-BGD&ĐT- TCCB. CUSC là trung tâm Công nghệ Phần mềm đầu tiên trong khu vực Đồng bằng sông Cửu Long (ĐBSCL) ra đời nhằm đáp ứng nhu cầu đào tạo lập trình viên chuyên nghiệp, phát triển phần mềm và ứng dụng công nghệ thông tin vào mọi mặt của đời sống xã hội.

Với phương châm hợp tác cùng phát triển, một mặt CUSC sản xuất những phần mềm theo nhu cầu thị trường; mặt khác, hỗ trợ các địa phương xây dựng đội ngũ làm phần mềm, đề xuất các giải pháp ứng dụng CNTT vào nhiều lĩnh vực khác nhau. CUSC luôn nỗ lực hết mình, tận tâm, đem lại tri thức và năng lực phục vụ, là bạn đồng hành tốt nhất trong việc phát triển ứng dụng CNTT vào việc quản lý, kinh doanh và đào tạo nhân lực, đáp ứng mọi nhu cầu cần thiết của con người và xã hội trong nền kinh tế thị trường và hội nhập quốc tế.

CUSC không ngừng xúc tiến mối quan hệ hợp tác và phát triển thị trường nước ngoài, góp phần xây dựng hình ảnh thương hiệu chung cho công nghiệp phần mềm Việt Nam trên thị trường quốc tế. Ngoài ra, CUSC mong muốn trở thành đối tác tin cậy, là một tổ chức phát triển trên cơ sở tri thức và công nghệ, góp phần đưa nền kinh tế, giáo dục tri thức trong khu vực và cả nước tiến nhanh, tiến mạnh và đem lại

cho mỗi thành viên của mình điều kiện phát triển tốt nhất, đầy đủ về vật chất và tinh thần.

Các hoạt động của CUSC tuân thủ quy trình chất lượng: ISO 9001:2015, hướng đến chuẩn CMNI nhằm đảm bảo chất lượng đạt chuẩn quốc gia và mang tầm vóc quốc tế.

Ngay từ những năm cuối của thế kỷ 20, Ban giám hiệu Trường Đại học Cần Thơ đã nhận thức cần phát triển một trung tâm công nghệ phần mềm nhằm đáp ứng các nhu cầu ngày càng tăng về phần mềm ứng dụng và về đào tạo lập trình viên chuyên nghiệp cho vùng ĐBSCL. Năm 2000, Trường Đại học Cần Thơ đã đầu tư mạnh mẽ, cải tạo nhà A1-Khu 3, mua sắm trang thiết bị, chuẩn bị cơ sở vật chất cho việc thành lập CUSC.

Tháng 3/2001, Bộ GD&ĐT ra quyết định thành lập CUSC. Cũng vào thời gian này, Ban giám hiệu trường Đại học Cần Thơ cử Giám đốc và điều động cán bộ cho trung tâm. Ngay từ ngày đầu thành lập, Ban giám đốc Trung tâm đã cho tiến hành một số phần mềm ứng dụng, tích cực tìm kiếm đối tác trong và ngoài nước cho đào tạo lập trình viên chuyên nghiệp.

Ban giám đốc đã xem xét các công ty đào tạo lập trình viên chuyên nghiệp ở các nước phát triển như Mỹ, Anh, Singapore,... và cuối cùng đã quyết định chọn một trong các công ty của Ấn Độ làm đối tác để đào tạo lập trình viên chuyên nghiệp. Hiệu trưởng trường Đại học Cần Thơ, Trưởng khoa Công nghệ Thông tin và Giám đốc Trung tâm CNPM đã đích thân sang Ấn Độ tham quan các công ty chuyên đào tạo lập trình viên trước khi quyết định chọn Công ty Aptech.

1.2 Cơ cấu tổ chức



Hình 2: Cơ cấu tổ chức

1.3 Địa chỉ liên hệ

Trung tâm Công nghệ Phần mềm Đại học Cần Thơ (CUSC).

Địa chỉ: 01 Lý Tự Trọng, Quận Ninh Kiều, TP. Cần Thơ.

Điện thoại: (84) 292 373 1072 - Fax: (84) 292 373 1071

Email: cusc@ctu.edu.vn - Website: <http://www.cusc.vn>

1.4 Thành tựu đạt được

- Năm 2001, trường Đại học Cần Thơ làm lễ ký kết hợp tác với Aptech thành lập Trung tâm Đào tạo Lập trình viên Quốc tế Mekong Delta – Aptech.
- Năm 2002, Trung Tâm Mekong Delta – Aptech mở lớp đào tạo kỹ thuật viên quốc tế với 96 sinh viên cho khóa đầu tiên.
- Năm 2005, Chương trình Quản trị mạng ACNA (Aptech Certified Network Administrator) được chính thức triển khai.
- Năm 2008, CUSC hợp tác với Prometric thành lập trung tâm khảo thí chứng chỉ CNTT quốc tế đầu tiên tại ĐBSCL.
- Năm 2009, CUSC xây dựng chương trình đào tạo quản trị mạng mang thương hiệu riêng của CUSC là Quản trị mạng I10 và Chuyên gia QTM Cao cấp.
- Năm 2011, được sự chấp nhận của Bộ GD&ĐT và trường Đại Học Cần Thơ, CUSC nâng tầm đào tạo lên hệ Cao Đẳng CNTT chính quy 2 chuyên ngành kỹ thuật phần mềm và Công nghệ đa phương tiện.
- Năm 2012, CUSC đã vinh dự nhận giải thưởng “Best Academic Center” của tập đoàn Aptech toàn cầu.
- Năm 2015, CUSC vinh dự nhận danh hiệu “Đơn vị đào tạo xuất sắc” và “Đơn vị đạt chất lượng đào tạo xuất sắc nhất” do Aptech Ấn Độ trao tặng năm 2015.
- Năm 2017, CUSC tiếp tục được vinh danh giải thưởng Sao Khuê 2017 do Hiệp hội Phần mềm và Dịch vụ Công nghệ thông tin Việt Nam (VINASA) tổ chức.

CHƯƠNG II: NỘI DUNG VÀ PHƯƠNG PHÁP THỰC HIỆN

1. Nội dung công việc thực tập

- Tìm hiểu và cài đặt Rasa chatbot.
- Đọc, tìm hiểu và sử dụng mã nguồn mở Rasa chatbot và xử lý ngôn ngữ tự nhiên.
- Tạo bộ tokenization tiếng Việt (custom tokenization) để cấu hình pipeline xử lý ngôn ngữ tự nhiên.
- Xây dựng cơ sở dữ liệu để hỗ trợ cho thực hiện các chức năng của chatbot.
- Xây dựng chức năng thống kê số lượng hồ sơ theo từng đơn vị và thời gian cụ thể.
- Cải tiến cho chatbot thông minh hơn bằng việc gợi ý cho người dùng chọn bằng các buttons.
- Nhúng chatbot vào website.
- Viết báo cáo.

2. Phương pháp, thời gian thực hiện

Tuần	Nội dung công việc được giao	Thời gian làm việc
Từ ngày 24/02 đến ngày 28/02	Tìm hiểu và cài đặt mã nguồn mở Rasa chatbot	6 buổi
Từ ngày 02/03 đến ngày 06/03	Đọc và tìm hiểu về mã nguồn mở Rasa chatbot và xử lý ngôn ngữ tự nhiên.	6 buổi
Từ ngày 09/03 đến ngày 13/03	Tạo bộ tokenization tiếng Việt để cấu hình pipeline xử lý ngôn ngữ tự nhiên.	6 buổi
Từ ngày 16/03 đến ngày 20/03	Xây dựng bộ cơ sở dữ liệu (database) của các đơn vị	6 buổi
Từ ngày 23/03 đến ngày 27/03	Xây dựng chức năng thống kê số lượng hồ sơ theo từng đơn vị và thời gian cụ thể.	6 buổi
Từ ngày 30/03 đến ngày 03/04	Cải tiến cho chatbot thông minh hơn bằng việc gợi ý cho người dùng chọn bằng các buttons.	6 buổi
Từ ngày 06/04 đến ngày 10/04	Đưa chatbot vào website.	6 buổi
Từ ngày 13/04 đến ngày 17/04	Viết báo cáo thực tập	6 buổi

Bảng 1: Bảng công việc thực hiện

CHƯƠNG III: NỘI DUNG THỰC TẬP

1. Tìm hiểu về chatbot và Rasa

1.1 Tìm hiểu về chatbot

Hệ thống đối thoại người máy hay còn gọi với thuật ngữ là chatbot. Chatbot là một chương trình máy tính tiến hành cuộc trò chuyện thông qua nhắn tin nhanh, nó có thể tự động trả lời những câu hỏi hoặc xử lý tình huống. Phạm vi và sự phức tạp của ChatBot được xác định bởi thuật toán của người tạo nên chúng. Chatbot thường được ứng dụng trong nhiều lĩnh vực như thương mại điện tử, dịch vụ khách hàng, y tế, tài chính ngân hàng, các dịch vụ giải trí...

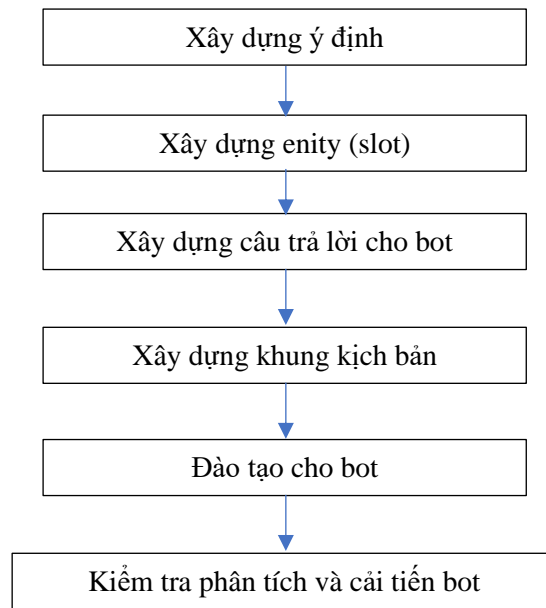
Chatbot có thể được chia thành 2 loại:

- Hệ thống hướng mục tiêu trên một miền ứng dụng (Task-Oriented)
- Hệ thống không có định hướng mục tiêu (chit-chat)

Chatbot có 3 thành phần chính là hiểu **ngôn ngữ tự nhiên** (NLU), **quản lý hội thoại** (DM), **thành phần sinh ngôn ngữ** (NLG). Các thành phần nhận dạng giọng nói Speech Recognition (text to speech hay speech to text) là các thành phần tăng cường. Mỗi thành phần trong chatbot đều có vai trò riêng:

- NLU: 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)
- DM: Quản lý hội thoại 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 dự định sẵn (history) đã đào tạo cho bot. Thành phần này cũng đảm nhiệm việc lấy dữ liệu từ hệ thống khác qua các API gọi trong action
- NLG: 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. NGL có thể được sinh ra câu trả lời dựa vào tập mẫu câu trả lời (pre-defined template) đã đào tạo cho bot.

Việc xây dựng chatbot sẽ thực hiện qua một số bước như sau:



Hình 3: Sơ đồ các bước xây dựng chatbot

1.2 Tìm hiểu về Rasa

Mã nguồn mở Rasa là là khung trò chuyện tự động, thông minh để xây dựng các trợ lý ảo theo ngữ cảnh câu chuyện.

Rasa cung cấp cho ta 2 phương pháp chính **xây dựng dữ liệu** training cho bot:

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

Có thể chọn Rasa để xây dựng một chatbot là vì Rasa *đễ tiếp cận*, hoạt động khá tốt và mạnh mẽ đặc biệt trong vấn đề xác định ý định người dùng và đối tượng được nhắc đến trong câu. Rasa là một mã nguồn mở giúp chúng ta dễ dàng xây dựng chatbot theo ý muốn và yêu cầu.

Rasa là một nền tảng chatbot bao gồm: Natural Language Unit (NLU), The Rasa Core Dialogue Engine và Rasa X.

2. Cài đặt Rasa

2.1 Yêu cầu của hệ thống

- Hệ điều hành là Ubuntu 18.04
- Python với phiên bản Python 3.6.7
- Môi trường ảo Virtualenv

2.2 Cài đặt

*** Đầu tiên ta cần cài đặt môi trường ảo Virtualenv:**

- Sử dụng lệnh `pip3 install virtualenv`
- Tiếp theo chúng ta di chuyển đến thư mục cần tạo dự án:

`cd my_project_folder`

- Sau đó ta tạo môi trường ảo cho dự án:

`virtualenv my_project_env`

- Để kích hoạt môi trường ảo:

`source my_project_env/bin/activate`

- Khi cần thoát môi trường ảo:

`deactivate`

*** Tiếp đến là phần cài đặt Rasa :**

- Đầu tiên cần kích hoạt môi trường ảo cho dự án của chúng ta

`source my_project_env/bin/activate`

- Kế tiếp cài đặt Rasa bằng lệnh:

`pip3 install rasa`

- Sau khi cài đặt xong, kế tiếp là khởi tạo dự án chatbot mới, ta sử dụng lệnh:

`rasa init`

Lúc này máy sẽ hỏi là bạn muốn lưu trong thư mục nào, nhấn Enter để lưu trong thư mục hiện tại hoặc có thể chọn một thư mục khác

Sau khi Rasa đã khởi tạo các file cơ bản, Rasa sẽ hỏi bạn có muốn huấn luyện mô hình hay không, mô hình huấn luyện này sẽ là tiếng Anh do chúng ta chưa chuẩn bị dữ liệu gì cho nó, nhấn Y nếu bạn muốn huấn luyện thử hoặc nhấn N nếu bạn muốn bỏ qua.

Như vậy là đã hoàn thành bước cài đặt Rasa cơ bản đã hoàn thành.

3. Cấu trúc của Rasa

3.1 Các thành phần cơ bản của Rasa

3.1.1 Cấu trúc thư mục

- data/nlu.md
- data/stories.md
- config.yml
- domain.yml
- actions.py
- endpoints.yml
- credentials.yml

3.1.2 File config.yml

Đây là nơi dùng để cấu hình cho NLU (Nature Language Understanding), nơi cho phép lựa chọn ngôn ngữ để xử lý cũng như các model cần thiết. Tùy theo các bộ Tokenizers mà chúng ta có thể chọn ngôn ngữ xử lý cho phù hợp.

* Hiện tại Rasa có cung cấp một số bộ Tokenizers cơ bản như:

- ***WhitespaceTokenizer***: Hỗ trợ Tokenizers cho các ngôn ngữ latin (tách dựa vào ký tự khoảng trắng) trong đó có ngôn ngữ tiếng Việt, tuy nhiên cách tách từ này chỉ tách được các từ đơn do đó khi xử lý với tiếng Việt sẽ cho kết quả không như mong đợi.

- ***JiebaTokenizer***: Tokenizers sử dụng Jieba cho ngôn ngữ tiếng Trung Quốc.

- ***ConveRTTokenizer***: Tokenizers sử dụng ConveRT model, dùng cho ngôn ngữ English

- Và còn nhiều bộ Tokenizers khác

- Ngoài ra, Rasa cho phép chúng ta có thể tự xây dựng một bộ Tokenizers riêng để có thể xử lý tốt hơn với những ngôn ngữ mà chúng ta chọn.

* Bên cạnh Tokenizers, chúng ta cũng có các lựa chọn khác:

- ***Text Featurizers***: dùng để lấy đặc trưng văn bản, rasa hỗ trợ một số bộ Text Featurizers cơ bản: MitieFeaturizers, SpacyFeaturizers, ConveRTFFeaturizers, RegexFeaturizers,...

- **EntityExtractors**: bộ tách các thực thể (entity Extractors), một số lựa chọn như sau: MitieEntityExtractors, SpacyEntityExtractors, CRFEntityExtractors, DucklingHTTPExtractor,...

- **Intent Classifiers**: bộ phân loại các ý đồ người dùng (intent), có các lựa chọn sau: MitieIntentClassifier, SklearnIntentClassifier, EmbeddingIntentClassifier, KeywordIntentClassifier,...

- Ngoài ra có thể sử dụng DIETClassifier để kết hợp EntityExtractors và Intent Classifiers.

Ngoài ra trong file **config.yml** này còn cho phép cấu hình cho Rasa Core, một số Policy cần thiết khi sử dụng như:

- **MemoizationPolicy**: quyết định tin nhắn 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 tin nhắn tiếp theo

- **MappingPolicy**: quyết định tin nhắn dựa vào dữ liệu đã ánh xạ

- **FallbackPolicy**: trong trường hợp, việc tính xác suất đầu ra không thể vượt được ngưỡng mà FallbackPolicy đề ra, tin nhắn trả ra sẽ là một utter_fallback (do người cài đặt định ra) ví dụ như: “Xin lỗi! Tôi chưa hiểu được yêu cầu của quý khách”.

- **FormPolicy**: cần khai báo khi bạn sử dụng Form để lấy dữ liệu từ người dùng.

3.1.3 File nlu.md

File nlu.md này chịu trách nhiệm về phần dữ liệu (data) cung cấp cho quá trình huấn luyện mô hình.

Ở đây chứa các câu mà người dùng có thể hỏi, các câu này được gom nhóm lại theo từng ý đồ của người dùng (intent). Với ý đồ là chào (greet) ta có ví dụ như sau:

intent: greet

- Chào bạn
- Hello
- Em ơi, cho anh hỏi

3.1.4 File domain.yml

File này dùng để khai báo cho rasa core biết được các intents, entities đã được xây dựng trong nlu.md, cũng như các phản hồi của chatbot (responses) ứng với từng ý đồ của người dùng và các hành động (actions) mà chatbot sẽ thực hiện khi bắt được ý đồ của người dùng. Qua đó hỗ trợ cho quá trình huấn luyện mô hình chatbot.

3.1.5 File stories.md

File này chứa các câu truyện, ngữ cảnh có thể xảy ra khi người dùng hỏi và chatbot trả lời, các câu truyện càng đa dạng thì chatbot có độ hiểu quả càng cao.

3.1.6 File actions.py

File này sẽ chứa các actions, form mà người cài đặt xây dựng lên để giải quyết các ý đồ của người dùng .

3.2 Các khái niệm cơ bản của Rasa

3.2.1 Rasa NLU

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.

3.2.2 Rasa core

Rasa core là 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.

3.2.3 Rasa X

Rasa X là một tool giúp chúng ta xây dựng, cải thiện và triển khai model chatbot vừa tạo.

3.2.4 Intent

RASA cần biết người dùng muốn gì, vì vậy cần nhận ra ý định của họ.

Ví dụ: Người dùng nói rằng: *"**Thông kê hồ sơ Sở Công thương**"* thì intent ở đây là **thong_ke**.

3.2.5 Entity

Entity là 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 sở công thương (don_vi)

3.2.6 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.

3.2.7 Action

Hành động 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.

3.2.8 Rasa form action

Khi chatbot cần thu thập nhiều thông tin từ người dùng để tiến hành một tác vụ như: thống kê hồ sơ của đơn vị nào đó, vào một thời gian cụ thể... Đó là lúc cần đến Form Action để thực hiện Slot Filling.

Form Action sẽ thực hiện hỏi lần lượt người dùng để thu thập đủ các thông tin intent cần thiết. Trong quá trình hỏi, Form cũng sẽ tiến hành xác nhận giá trị của các intent để đảm bảo giá trị thu được là hợp lí.

3.2.9 Rasa images and buttons

Rasa images and buttons là một công cụ của Rasa giúp đỡ việc xây dựng chatbot hiệu quả và dễ sử dụng hơn. Giúp tạo ra các nút, hình ảnh để người dùng có thể click chọn yêu cầu mà không cần nhập văn bản cụ thể. Qua đó việc thực hiện các yêu cầu mà người dùng đưa ra sẽ chính xác hơn.

4. Xây dựng Chatbot cho dịch vụ công Sóc Trăng

4.1 Cấu hình pipeline xử lý ngôn ngữ tiếng Việt

Để có thể xử lý tốt cho ngôn ngữ tiếng Việt, chúng ta cần một bộ Tokenizer dành riêng cho tiếng việt. Hiện tại Rasa chưa có bộ Tokenizer hỗ trợ riêng cho tiếng việt nên chúng ta phải tự đi xây dựng nó.

Điểm đặc biệt của ngôn ngữ tiếng việt là có các từ ghép, chẳng hạn như “sở y tế” nếu xử lý như ngôn ngữ latin khác thì cụm từ trên sẽ được xử lý thành “sở”, “y”, “tế”, điều đó dẫn đến việc xử lý với tiếng việt sẽ không đạt được hiệu quả cao vì nó không xử lý được các từ ghép.

Để giải quyết vấn đề trên, ta sẽ sử dụng thư viện pyvi để hỗ trợ tách từ tiếng việt. Thư viện này được phát triển bởi Tran Viet Trung vào năm 2016, một số tính năng cơ bản được pyvi hỗ trợ như: Tokenize, POS tag, Remove accents, Add

accents. khi sử dụng pyvi thì cụm từ “sở y tế” sẽ được tách thành “sở”, “y tế”, hay với một ví dụ khác là “Tra cứu địa chỉ Sở Công Thương” sẽ được pyvi tách thành “Tra cứu”, “địa chỉ”, “Sở”, “Công Thương”

Các bước thực hiện:

- Đầu tiên truy cập vào:

`my_project_env/lib/python3.6/site-packages/rasa/nlu/tokenizers`

- Ở đây tạo một file **vietTokenizer.py** với nội dung được mô tả ở phần phụ lục
- Ở đây ta sẽ sử dụng hàm `ViTokenizer.spacy_tokenize` của pyvi để tách từ tiếng việt

Sau khi đã xây dựng xong Tokenizer, bây giờ ta cần khai báo nó với rasa.

- Đầu tiên truy cập vào:

`my_project_env/lib/python3.6/site-packages/rasa`

- Thêm dòng khai báo sau vào trong file **registry.py**

```
from rasa.nlu.tokenizers.vietTokenizer import vietTokenizer
```

- Và thêm vietTokenizer vào trong `component_classes`

```
component_classes = [  
    # utils  
    SpacyNLP,  
    MitieNLP,  
    HFTransformersNLP,  
    # tokenizers  
    vietTokenizer,  
    MitieTokenizer,  
    SpacyTokenizer,
```

Như vậy là ta đã tạo xong vietTokenizer, bây giờ đã có thể khai báo vietTokenizer và sử dụng.

4.2 Tạo bộ dữ liệu (database)

Trước khi xây dựng các chức năng cho chatbot, ta cần xây dựng database để sử dụng cho các chức năng như thống kê hồ sơ hay tra cứu địa chỉ của một đơn vị nào đó.

Với dữ liệu đầu vào được lưu trong file data_full.xlsx, dữ liệu đầu vào trong có cấu trúc như sau:

STT	COMPANYID	ĐỊA CHỈ WEBSITE	TÊN ĐƠN VỊ	TỪ ĐỒNG NGHĨA	ĐỊA CHỈ
1	69	motcua.sotttt.soctrang.gov.vn	Sở Thông tin và truyền thông	Sở tttt, sở tt và tt, sở thông tin & truyền thông	Địa chỉ: 56 Lê Duẩn, Phường 3, TP Sóc Trăng, tỉnh Sóc Trăng Điện thoại: 0299 3621 090, Fax: 0299 3621 171, Email: sotttt@soctrang.gov.vn .
2

Bảng 2: Bảng mô tả dữ liệu dùng làm cơ sở dữ liệu

Ta sẽ sử dụng chuyển dữ liệu này thành dạng json và lưu trữ chúng với tên data_full.json. Khi chuyển về json dữ liệu trên có dạng:

```
{
  "1": {"id": "69",
    "đơn vị": "sở thông tin và truyền thông",
    "từ đồng nghĩa": ["sở tttt", "so thông tin & truyền thông", "sở tt và tt", "sở thông tin & truyền thông "],
    "domain": "motcua.sotttt.soctrang.gov.vn",
    "địa chỉ": "Địa chỉ: 56 Lê Duẩn, Phường 3, TP Sóc Trăng, tỉnh Sóc Trăng\nĐiện thoại: 0299 3621 090, Fax: 0299 3621 171, Email: sotttt@soctrang.gov.vn."
  },
  "2": {...}
}
```

Khi cần sử dụng tới database này ta chỉ cần đọc file data_full.json và truy vấn đến dữ liệu cần sử dụng.

4.3 Xây dựng chức năng thống kê số lượng hồ sơ của một đơn vị

Chức năng này được xây dựng với mục đích hỗ trợ người dùng trong việc thống kê số lượng hồ sơ của một đơn vị nào đó một cách tiện lợi và nhanh chóng.

Tương tự như xây dựng chức năng tra cứu hồ sơ theo mã hồ sơ, đầu tiên ta cần làm cho chatbot hiểu được ý định của người dùng là muốn thống kê số lượng hồ sơ của một đơn vị nào đó.

Nhưng khác biệt với chức năng tra cứu mã hồ sơ, ở chức năng này thì cần số lượng thông tin nhiều hơn, Chẳng hạn như là đơn vị cần thống kê là đơn vị nào, thống kê vào thời gian nào.

Ví dụ khi người dùng nhắn tin: “Hãy thống kê hồ sơ Sở Công Thương vào tháng 3 năm 2020 giúp tôi”, trong câu trên thì ý định của người dùng là thống kê số lượng hồ sơ (intent là `thong_ke`) và các thực thể cần quan tâm là đơn vị cần thống kê là Sở Công Thương (`don_vi`) và thống kê vào tháng 3 (`thang_tk`) năm 2020 (`nam_tk`).

Việc bây giờ là làm sao cho chatbot hiểu được ý đồ người dùng và phát hiện được các thực thể, để thực hiện được điều này ta phải xây dựng chúng trong file **nlu.md**, cụ thể như sau:

```
## intent:thong_ke
- Thống kê hồ sơ toàn [Tỉnh Sóc Trăng](don_vi) giúp tôi
- thống kê hồ sơ [tỉnh sóc trăng](don_vi:Tỉnh Sóc Trăng) dùm tôi
- Thống kê [ban dân tộc](don_vi)
- Thống kê [Sở Tài Nguyên Và Môi Trường](don_vi)
- Thống kê [sở tài nguyên và môi trường](don_vi:Sở Tài Nguyên Và Môi Trường)
- Thống kê hồ sơ [Sở Y Tế](don_vi)
- Thống kê hồ sơ [sở yt](don_vi:Sở Y Tế) giúp tôi
- Kết quả thống kê [huyện cù lao dung](don_vi)
- Kết quả thống kê [xã hồ đặc kiện](don_vi) như thế nào?
- Kết quả thống kê [Thành Phố Sóc Trăng](don_vi) ra sao?
- Thống kê hồ sơ [thị trấn cù lao dung](don_vi)
- Kết quả thống kê [tp sóc trăng](don_vi:Thành Phố Sóc Trăng)
- Thống kê hồ sơ [phường 1](don_vi)
- Thống kê hồ sơ [thanh tra tỉnh](don_vi)
- Thống kê hồ sơ [Công ty hư cầu](don_vi)
- Kết quả thống kê [sở văn hóa thể thao và du lịch](don_vi)
- Kết quả thống kê [sở giáo dục và đào tạo](don_vi)
- Thống kê hồ sơ toàn [Tỉnh Sóc Trăng](don_vi) vào tháng [01](thang_tk) năm [2021](nam_tk) giúp tôi
- thống kê hồ sơ [tỉnh sóc trăng](don_vi:Tỉnh Sóc Trăng) vào tháng [01](thang_tk) năm [2021](nam_tk)
- thống kê [tỉnh st](don_vi:Tỉnh Sóc Trăng) vào tháng [01](thang_tk) năm [2021](nam_tk)
- Kết quả thống kê [sở giáo dục và đào tạo](don_vi) vào tháng [03](thang_tk) năm [2018](nam_tk)
```

Tuy nhiên có một số trường hợp người dùng chỉ hỏi: “Hãy thống kê hồ sơ Sở Công Thương” hay “Hãy thống kê hồ sơ Sở Công Thương vào năm 2020”, thậm chí có trường hợp là “Hãy thống kê hồ sơ”, nhìn chung trong các trường hợp đó thông tin người dùng cung cấp không đủ để có thể tiến hành thống kê. Do đó Rasa có một công cụ để hỗ trợ cho người cài đặt giải quyết được vấn đề trên, công cụ đó là Rasa Form. Để sử dụng được Form này ta cần khai báo trước trong file `config.yml`

```
policies:
  - name: FormPolicy
```

Để sử dụng form ta cần xây dựng thêm một intent để form lấy thông tin, cụ thể như sau:

```
## intent:inform
- vào tháng [02](thang_tk)
- vào năm [2061](nam_tk)
- tháng [03](thang_tk)
- năm [2041](nam_tk)
- tháng [1](thang_tk)
- tháng [3](thang_tk)
- tháng [2](thang_tk)
- tháng [12](thang_tk)
- tháng [6](thang_tk)
- [6](thang_tk)
- [7](thang_tk)
- năm [2012](nam_tk)
- tháng [01](thang_tk) năm [2021](nam_tk)
```

Tiếp theo, ta cần khai báo các intents, entites, actions và form vào trong file domain.yml

```
intents:
- thông_ke
- inform
forms:
- thôngke_form
entities:
- thang_tk
- nam_tk
- don_vi
slots:
thang_tk:
  type: unfeaturized
nam_tk:
  type: unfeaturized
don_vi:
  type: unfeaturized
requested_slot:
  type: unfeaturized
responses:
utter_loi:
- text: "Không thể kết nối với server! Mong quý khách thông cảm!"
- text: "Lỗi kết nối server!"
- text: "Server đang bảo trì!"
utter_ask_don_vi:
- text: "Bạn muốn thống kê đơn vị nào?"
utter_ask_thang_tk:
- text: "Bạn muốn thống kê {don_vi} vào tháng mấy?"
utter_ask_nam_tk:
- text: "Bạn muốn thống kê {don_vi} vào năm mấy?"
utter_saidv:
- text: "Vui lòng kiểm tra lại đơn vị bạn vừa nhập!"
utter_saithang:
- text: "Vui lòng kiểm tra lại tháng bạn vừa nhập!"
utter_sainam:
- text: "Vui lòng kiểm tra lại năm bạn vừa nhập!"
utter_thong_ke:
- text: "Thống kê {don_vi} vào tháng {thang_tk} năm {nam_tk} có kết quả như sau:"
actions:
- action_thong_ke_full
```


Ngoài ra ta thấy trong file này còn có slots, slots này rasa dùng để chứa các thực thể mà rasa xác định được từ trong tin nhắn của người dùng. Ứng với các thực thể thang_tk, nam_tk, don_vi sẽ có các slot tương tự, và đặc biệt sẽ có slot khác có tên là requested_slot, slot này dùng để xác định thực thể mà form đang thiếu và đưa ra yêu cầu để người dùng cung cấp thông tin cho form.

* **Về phần forms**, dùng để khai báo form thông kê_form được xây dựng trong file actions.py.

- Form này được xây dựng với các slots cần được lấp đầy là “don_vi”, “thang_tk”, “nam_tk”
- Trong đó:
 - o entity “don_vi” sẽ được xác định từ intents là “thong_ke”
 - o entity “thang_tk” sẽ được xác định từ intents là “thong_ke” và “inform”
 - o entity “nam_tk” sẽ được xác định từ intents là “thong_ke” và “inform”
- Trong form này ta sẽ xây dựng các hàm để kiểm tra xem thông tin người dùng nhập vào có đầy đủ hết các slots cần được lấp đầy hay không, nếu chưa thì form sẽ báo lại cho chatbot và thông báo cho người dùng là còn thiếu thông tin gì và yêu cầu họ nhập bổ sung cho đến khi lấp đầy hết các entity. Ngoài ra, ta còn có thể xây dựng các hàm kiểm tra xem liệu các entity được xác định có đúng với yêu cầu hay không. Trong trường hợp này ta sẽ xây dựng lần lượt các hàm kiểm tra entity “don_vi”, “thang_tk” và “nam_tk”
 - o Với entity “don_vi”: khi đã xác định được đơn vị ta sẽ tiến hành kiểm tra xem liệu đơn vị đó có tồn tại trong cơ sở dữ liệu hay chưa, nếu chưa có thì báo lỗi với người dùng và yêu cầu họ nhập lại.
 - o Với entity “thang_tk”: ta sẽ kiểm tra xem tháng này có hợp lệ hay không, nếu không thì báo lỗi và yêu cầu người dùng nhập lại.
 - o Với entity “thang_tk”: ta sẽ kiểm tra xem tháng này có hợp lệ hay không, nếu không thì báo lỗi và yêu cầu người dùng nhập lại.

* **Về phần responses**, đây là phần phản hồi của chatbot với người dùng. Ta sẽ xét lần lượt các phản hồi này:

- utter_loi: dùng để phản hồi với người dùng khi không thể lấy thông tin từ API
- utter_ask_don_vi: dùng để hỏi người dùng về thông tin đơn vị
- utter_ask_thang_tk: dùng để hỏi người dùng về thông tin tháng cần thống kê

- utter_ask_nam_tk: dùng để hỏi người dùng về thông tin năm cần thống kê
- utter_saidv: dùng để phản hồi cho người dùng khi đơn vị họ nhập không có trong database
- utter_saithang: dùng để phản hồi cho người dùng khi họ nhập sai tháng
- utter_sainam: dùng để phản hồi cho người dùng khi họ nhập sai năm
- utter_thong_ke: dùng để phản hồi kết quả thống kê cho người dùng

* **Về phần actions**, đây là nơi sẽ thực hiện nhiệm vụ kết nối với API để lấy kết quả thống kê về và trả kết quả cho người dùng, action_thong_ke_full sẽ được khai báo trong file actions.py

Với API thống kê số lượng hồ sơ của một đơn vị nào đó, URL có dạng như sau:

http://dichvucong.soctrang.gov.vn/api/hoso/tktxlhs?domain=domain&thang=thang_tk&nam=nam_tk

Trong đó tham số truyền vào:

- domain: tên domain muốn thống kê
- thang_tk: tháng cần thống kê
- nam_tk: năm cần thống kê

API sẽ trả về kết quả dạng từ điển (dictionary), có dạng:

```
{
    "hs_xl_dunghan_thang": 46789
    "title_daxyly_thang": 0.9961909598781107,
    "hs_xl_dunghan_nam": 143438,
    "hs_tronghan_nam": 16015,
    "hs_xylytrehan_thang": 179,
    "title_daxuly_nam": 0.9982095793763577,
    "hs_tronghan_thang": 287,
    "hs_moitiiepnhan_thang": 47256,
    "hs_moitiiepnhan_nam": 159739,
    "hs_xulytrehan_nam": 282
}
```

Với API như vậy thì action_thong_ke_full sẽ thực hiện như sau:

Khi form đã thu thập đủ các slots “don_vi”, “thang_tk” và “nam_tk” Khi đó chatbot sẽ gọi đến actions_thong_ke_full với đầu vào là “don_vi”, “thang_tk” và “nam_tk”.

Ở đây, action_thong_ke_full sẽ có nhiệm vụ là từ “don_vi” đầu vào và tìm trong cơ sở dữ liệu xem với “don_vi” đó sẽ có domain là gì và lấy giá trị domain đó kết hợp “thang_tk” và “nam_tk” để truy cập đến địa chỉ URL ở trên, tiếp đến ta chỉ việc parser URL ở trên với từ khóa là mã hồ sơ là sẽ thu được kết quả, trong python đã có thư viện bs4 hỗ trợ việc parser này. Sau khi đã lấy được dữ liệu ta chỉ cần dùng hàm dispatcher.utter_message() do rasa hỗ trợ đã trả tin nhắn kết quả về cho người dùng.

Tiếp theo là phải xây dựng câu chuyện (stories) cho chatbot, phải có câu chuyện này thì chatbot mới hiểu được các ngữ cảnh để có thể xử lý được tốt và chính xác theo ý đồ của người dùng. Do vậy xây dựng càng nhiều câu chuyện có thể xảy ra thì chatbot sẽ xử lý càng tốt. Các câu chuyện này được xây dựng trong file stories.md. Với chức năng tra cứu hồ sơ theo mã hồ sơ sẽ có câu chuyện như sau:

Như vậy ta đã xây dựng xong phần dữ liệu và cấu hình, cuối cùng là chỉ cần huấn

```
##thong_ke
* thong_ke
  - thongke_form
  - form{"name": "thongke_form"}
  - form{"name": null}
  - action_thong_ke_full
  - slot{"requested_slot": null}
```

luyện mô hình là xây dựng xong chatbot với chức năng tra cứu hồ sơ theo mã hồ sơ.

4.4 Cải tiến chatbot

Đôi khi người dùng chỉ nhập vào tên của một đơn vị vào thì vấn đề được đặt ra ở đây là chatbot sẽ không hiểu được ý định của người dùng là gì: “người dùng muốn thống kê hồ sơ của đơn vị đó” hay “người dùng muốn tra cứu thông tin, địa chỉ của đơn vị đó”. Do đó chatbot sẽ không thể trả lời được yêu cầu của người dùng.

Để giải quyết được vấn đề đó Rasa có hỗ trợ các Rasa Button. Khi người dùng chỉ nhập vào tên của đơn vị thì chatbot sẽ hiện ra các nút cho người dùng chọn. Để xây dựng các button cho chatbot cần thực hiện các bước như sau.

Đầu tiên cần phải cho chatbot hiểu được văn bản mà người dùng vừa nhập vào là tên của một đơn vị. Để thực hiện được vấn đề này chúng ta cần khai báo trong file nlu.md với intents mới.

```
## intent:chi_co_dv
- [Tỉnh Sóc Trăng](don_vi)
- [sở văn hóa thể thao và du lịch](don_vi)
- [huyện cù lao dung](don_vi)
- đơn vị [sở giáo dục và đào tạo](don_vi)
- [tỉnh st](don_vi:Tỉnh Sóc Trăng)
- [xã hồ đặc kiện](don_vi)
- [thị trấn cù lao dung](don_vi)
- đơn vị [sở công thương](don_vi)
- [ban dân tộc](don_vi)
- [Tỉnh Sóc Trăng](don_vi)
- [tỉnh st](don_vi:Tỉnh Sóc Trăng)
- đơn vị [thị trấn cù lao dung](don_vi)
- [sở giao thông vận tải](don_vi)
- [sở thông tin và truyền thông](don_vi)
```

Tiếp theo, ta cần khai báo intent vừa tạo và entities vào file domain.yml

```
intents:
  - chi_co_dv

entities:
  - don_vi

slots:
  don_vi:
    type: unfeaturized

responses:|
utter_chi_co_dv:
  - text: "Bạn cần thông tin gì ở {don_vi}?"
    buttons:
      - title: "Thống kê số lượng hồ sơ"
        payload: "Thống kê hồ sơ"
      - title: "Tìm thông tin địa chỉ"
        payload: 'Địa chỉ'

actions:
  - utter_chi_co_dv

session_config:
  session_expiration_time: 1
  carry_over_slots_to_new_session: true
```

Trong file domain.yml cần khai báo thêm phần responses (phản hồi của chatbot với ý đồ của người dùng). Phần phản hồi của chatbot sẽ không còn là văn bản mà sẽ là các nút để người dùng lựa chọn. Với hình trên nếu người dùng nhập vào tên một đơn vị chatbot sẽ hiện ra 2 buttons đó là “Thống kê số lượng hồ sơ” và “Tìm thông tin địa chỉ” của đơn vị mà người dùng nhập. Khi người dùng ấn chọn “Tìm thông tin địa chỉ” thì chatbot sẽ phản hồi thông tin địa chỉ của đơn vị người

dùng nhập vào. Nếu ấn vào nút “hồng kê số lượng hồ sơ” thì chatbot sẽ gọi đến chức năng thống kê số lượng hồ sơ.

Tiếp theo là ta cần xây dựng một câu chuyện cho chatbot. Câu chuyện sẽ được xây dựng trong file stories.md.

```
##chi co don vi
* chi_co_dv
- utter_chi_co_dv
```

Như vậy ta đã xây dựng xong phần dữ liệu và cấu hình, cuối cùng là chỉ cần huấn luyện mô hình là xây dựng xong chatbot với các nút chọn được cải tiến thông minh hơn.

4.5 Đưa chatbot vào website

Khi đã xây dựng thành công chatbot, việc quan trọng tiếp theo là phải đưa nó vào sử dụng ở thực tế như thế nào. Hiện này có rất nhiều cách để thực hiện việc này chẳng hạn như kết nối rasa chatbot với facebook messenger, zalo chat, slack và website. Nhưng về mức độ thông dụng và khả năng độc lập riêng với các mạng xã hội thì đưa rasa chatbot vào website là một lựa chọn tối ưu.

Rasa chatbot khi được xây dựng nên mặc định sẽ chạy với server là localhost. Do đó ta cần phải đưa cái server rasa chatbot này ra ngoài internet, ở đây ta sẽ chọn phần mềm Ngrok để thực hiện việc này.

Các bước thực hiện việc đưa chatbot vào website như sau:

- Đầu tiên ta sẽ khai socketio vào file **credentials.yml** cụ thể như sau:

```
rasa:
  url: "http://localhost:5002/api"

socketio:
  user_message_evt: user_uttered
  bot_message_evt: bot_uttered
  session_persistence: flase
```

- Sau đó ta khởi chạy server của chatbot tại localhost và khởi chạy actions của chatbot

- Kế tiếp khởi chạy ngrok tại port 5005, đợi một lúc ta sẽ thấy:

```
Session Status      online
Session Expires    7 hours, 59 minutes
Version            2.3.35
Region             United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding          http://cd30965a.ngrok.io -> http://localhost:5005
Forwarding          https://cd30965a.ngrok.io -> http://localhost:5005

Connections
t1l      opn      rt1      rt5      p50      p90
0         0         0.00    0.00    0.00    0.00
```

2 dòng Forwarding này là địa chỉ mà ngrok đưa server rasa chatbot ra ngoài internet, ta chỉ việc lấy nó và sử dụng.

- Sau khi đã đưa được server ra ngoài, bây giờ ta sẽ đưa giao diện chatbot lên trên website. Rasa có hỗ trợ cho người cài đặt một đoạn script để người cài đặt có thể dễ dàng đưa chatbot của mình lên website của họ. Cụ thể đoạn script sẽ được mô tả ở phần phụ lục.

- Người cài đặt chỉ cần đưa đoạn script này vào trong phần body của trang web của họ và thay đổi socketURL thành địa chỉ Forwarding mà ngrok đã cung cấp ở bước trên.

- Như vậy là đã hoàn thành việc kết nối chatbot vào website, bây giờ đã có thể đưa hệ thống vào sử dụng.

CHƯƠNG 4: KẾT QUẢ VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả và kinh nghiệm học tập qua thời gian thực tập thực tế

Sau khoản thời gian thực tập thực tế em đã xây dựng được một chatbot hỗ trợ người dùng trong việc thống kê số lượng hồ sơ của một đơn vị ở tỉnh Sóc Trăng, hay địa chỉ của từng đơn vị cũng như tra cứu hồ sơ theo mã hồ sơ của các nhân hay tổ chức nào đó ở tỉnh Sóc Trăng.

Ngoài ra, còn giúp em làm quen với môi trường làm việc ở công ty, qua đó giúp em rèn luyện được tinh thần tự giác, tác phong công nghiệp.

Bên cạnh đó, sau kỳ thực tập thực tế này còn giúp em rèn luyện cũng như trao dồi được khả năng ngoại ngữ của mình.

2. Những kiến thức lý thuyết được củng cố

Sau khi thực tập kết thúc, bản thân em đã cũng cố lại được khả năng lập trình với ngôn ngữ python, ngoài ra em còn cũng cố lại được các kiến thức về nguyên lý máy học cũng như trí tuệ nhân tạo.

3. Những điểm hạn chế

Mặc dù đã hoàn thành được sản phẩm, tuy nhiên vẫn còn hạn chế ở các chức năng của chatbot như:

- Chưa xây dựng được cho chatbot trả lời được những câu hỏi thường gặp
- Chưa xây dựng được chức năng tra cứu thủ tục
- Chưa xây dựng được cho chatbot có khả năng tự học những câu hỏi mới

4. Hướng phát triển

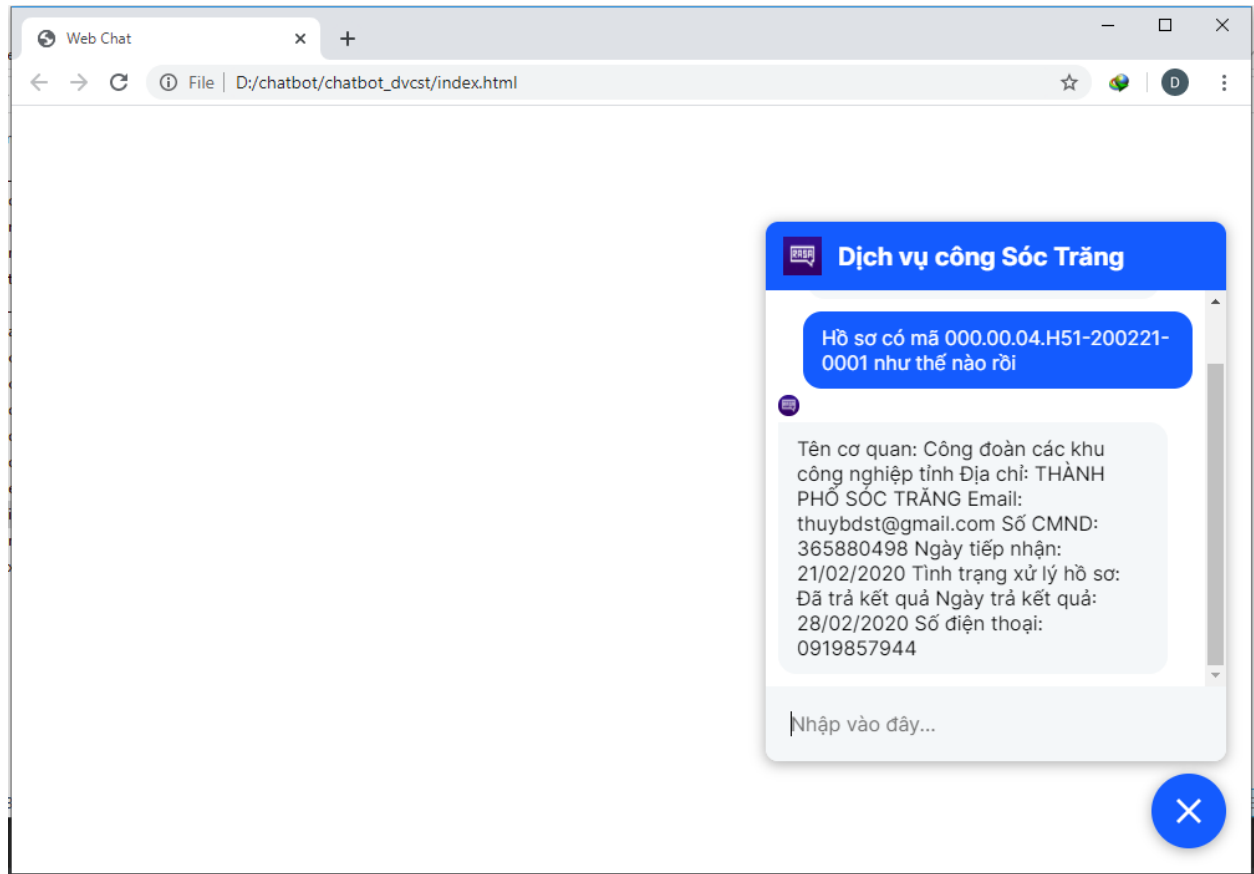
Trong thời gian tới, em sẽ cố gắng xây dựng thêm chức năng tra cứu thủ tục và chức năng trả lời được những câu hỏi thường gặp, bên cạnh đó tìm cách xây dựng cho chatbot có khả năng tự học được những câu hỏi mới hay những ngữ cảnh mà chatbot chưa được xây dựng từ trước.

Ngoài ra, em sẽ xây dựng chatbot với rasa cho nhiều lĩnh vực mới như trong giáo dục, y tế hay kinh doanh.

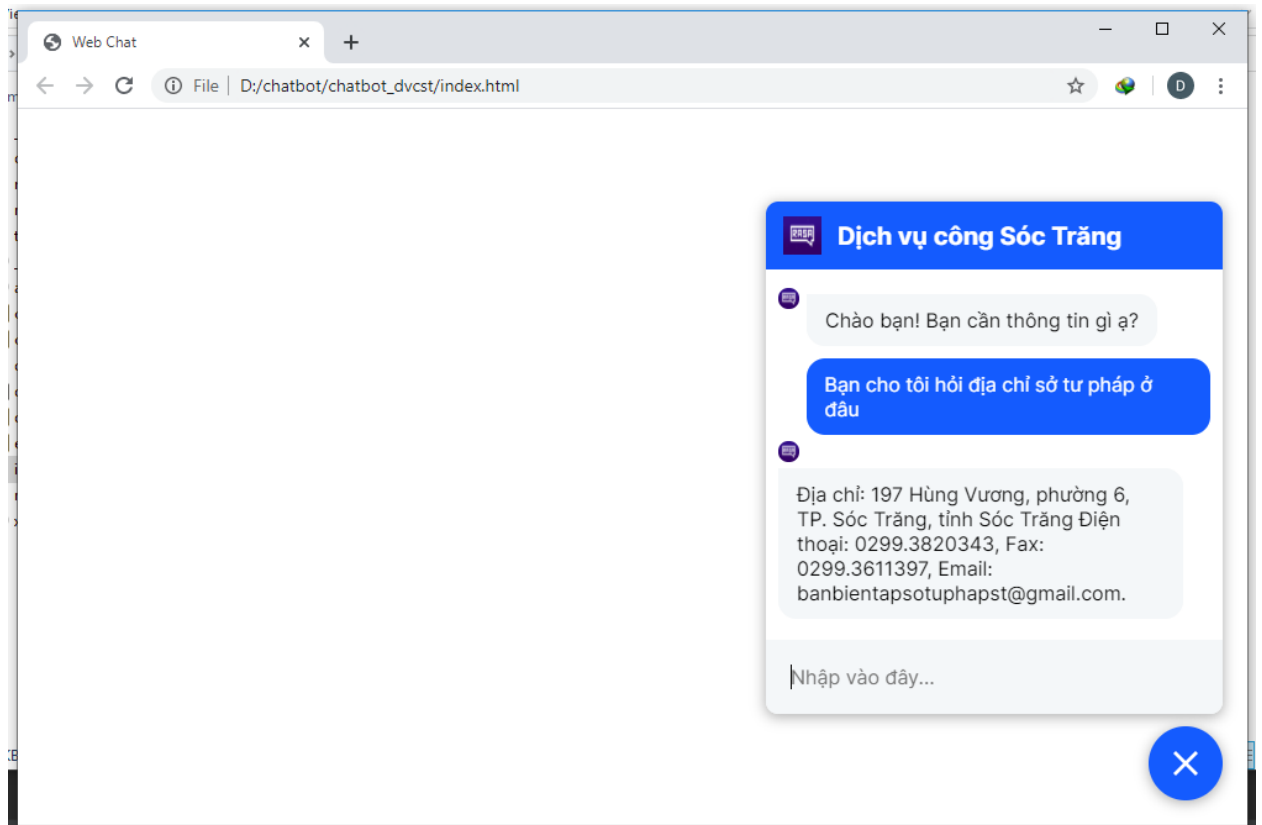
PHỤ LỤC

1. Giao diện Chatbot

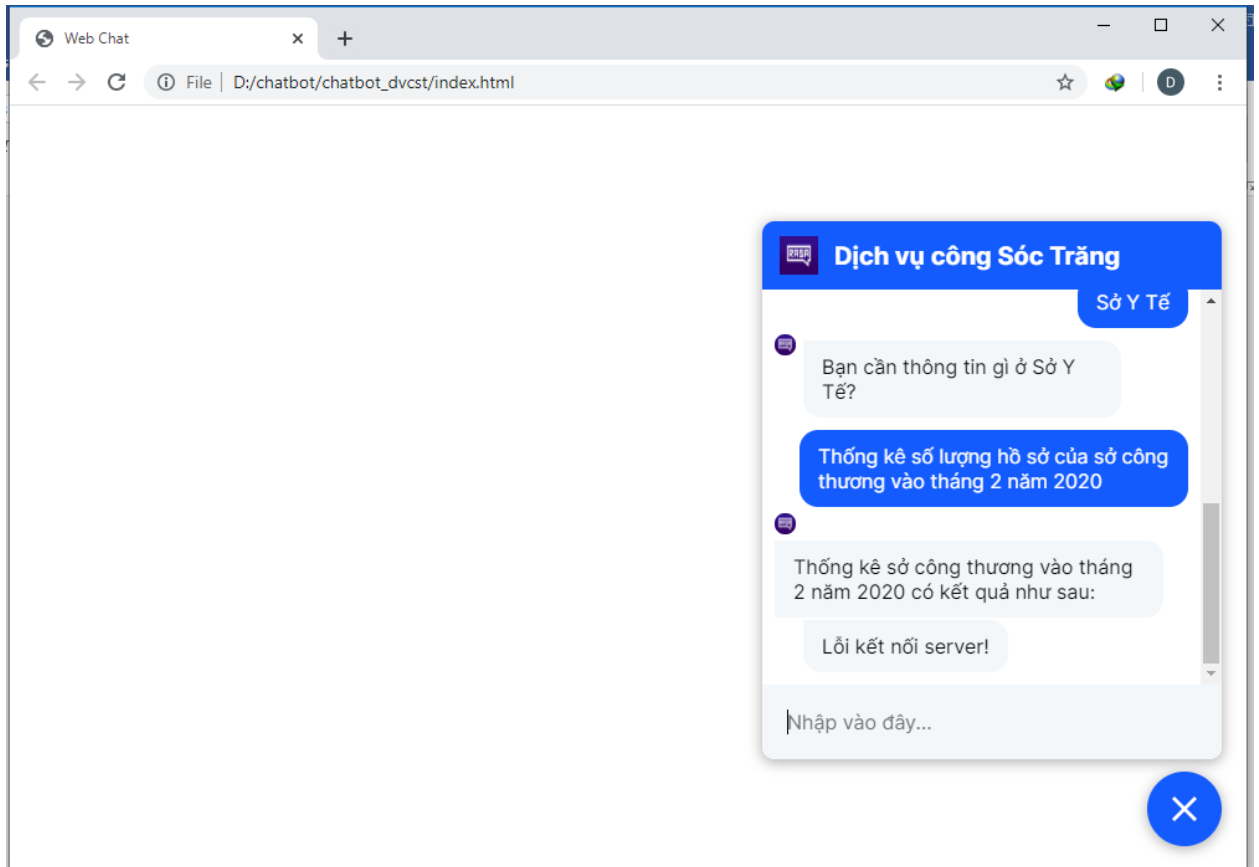
➤ Chức năng tra cứu hồ sơ theo mã hồ sơ



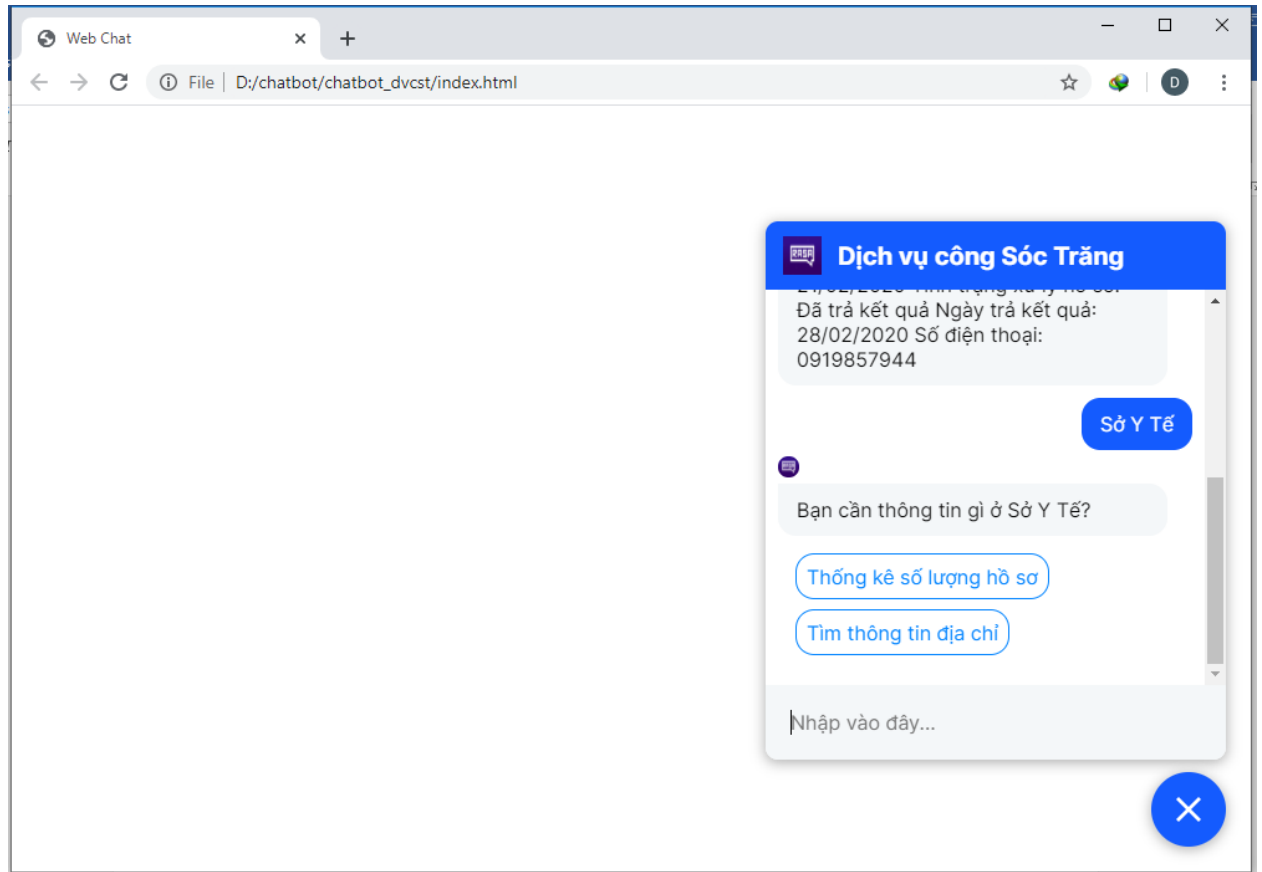
➤ Chức năng tra cứu địa chỉ của đơn vị



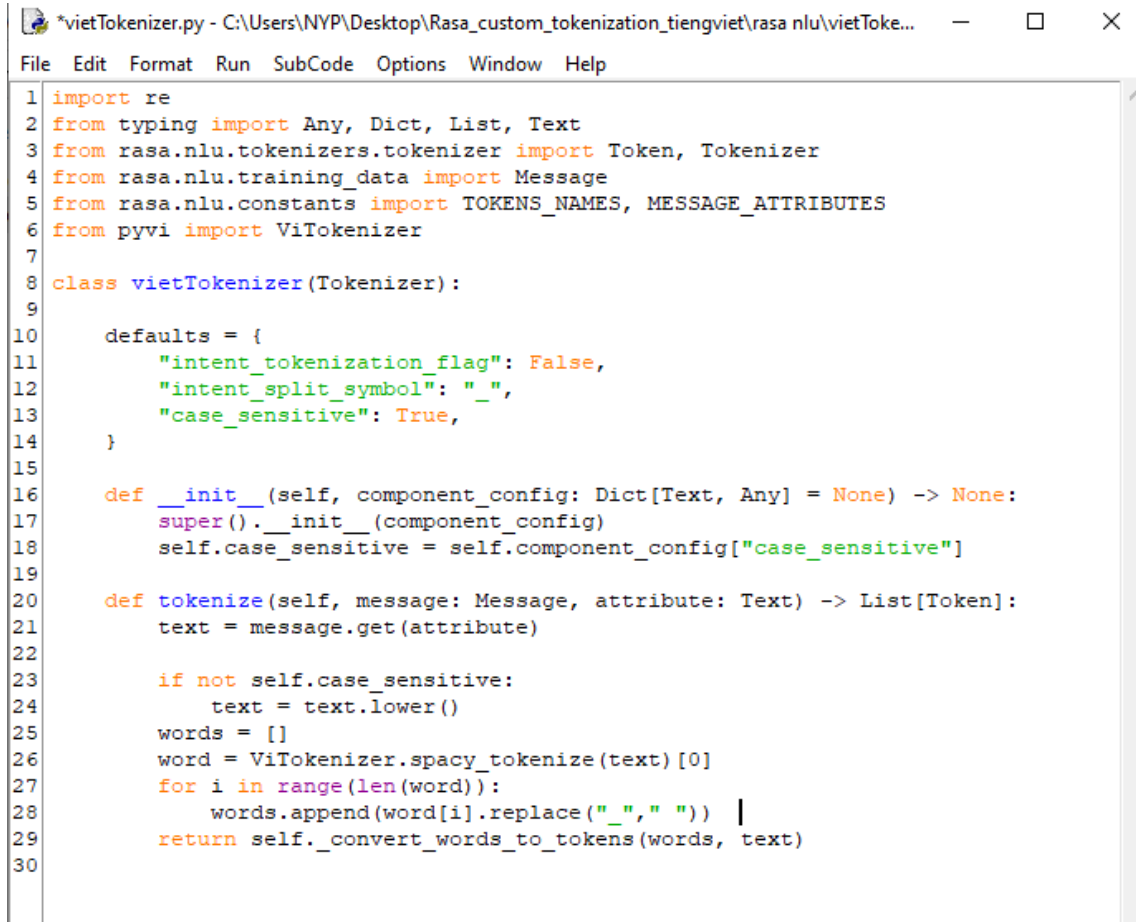
➤ Chức năng thống kê số lượng hồ sơ của từng đơn vị



➤ Chatbot có thể gợi ý cho người dùng chọn



2. Nội dung file vietTokenizer.py



```
1 import re
2 from typing import Any, Dict, List, Text
3 from rasa.nlu.tokenizers.tokenizer import Token, Tokenizer
4 from rasa.nlu.training_data import Message
5 from rasa.nlu.constants import TOKENS_NAMES, MESSAGE_ATTRIBUTES
6 from pyvi import ViTokenizer
7
8 class vietTokenizer(Tokenizer):
9
10     defaults = {
11         "intent_tokenization_flag": False,
12         "intent_split_symbol": "_",
13         "case_sensitive": True,
14     }
15
16     def __init__(self, component_config: Dict[Text, Any] = None) -> None:
17         super().__init__(component_config)
18         self.case_sensitive = self.component_config["case_sensitive"]
19
20     def tokenize(self, message: Message, attribute: Text) -> List[Token]:
21         text = message.get(attribute)
22
23         if not self.case_sensitive:
24             text = text.lower()
25         words = []
26         word = ViTokenizer.spacy_tokenize(text)[0]
27         for i in range(len(word)):
28             words.append(word[i].replace("_", " "))
29         return self._convert_words_to_tokens(words, text)
30
```

3. Nội dung script để thêm chatbot vào website

```
<div id="webchat"/>
<script src="https://storage.googleapis.com/mrbot-cdn/webchat-latest.js"></script>
<script>
  WebChat.default.init({
    selector: "#webchat",
    initPayload: "/intro_registered_accounts",
    socketUrl: "địa chỉ ngrok",
    tooltipPayload: "/tooltip_registered_accounts",
    tooltipDelay: 40000,
    socketPath: "/socket.io/",
    customData: {
      language: 'vi'
    },
    title: "Dịch vụ công Sốc Trắng",
    inputTextFieldHint: "Nhập vào đây...",
    connectingText: "Vui lòng chờ trong giây lát...",
    profileAvatar: "rasa.png",
    hideWhenNotConnected: false,
    defaultHighlightAnimation: `@keyframes default-botfront-blinker-animation {
      from {
        outline-style: none;
        outline-color: red;
      }
      to {
        outline-style: solid;
        outline-color: red;
      }
    }`,
    onSocketEvent: {
      'bot_uttered': () => console.log('bot uttered'),
    },
    docViewer: false,
    params: {
      images: {
        dims: {
          width: 300,
          height: 200
        }
      },
    },
    storage: "session"
  })
</script>
```