

DEPLOY YOUR OWN FAQ WITH RASA CHATBOT- WHEN QUESTION AND ANSWERING IS NOT ENOUGH

A Contextual chatbot using open-source rasa

Introduction

We used to hear about the question and answering, when we ask something we have an answer. It's like we play a quiz game. But it's just the answer from the question. How about the question but related to each other? Imagine you go to the university to have the quiz test but the question 1 related to question 2, 3, 4,...etc.. That's the time you think that we can't get all the answer. But sometimes, the answer is in the question related to each other. That's how we have "context".

In this blogpost, I will introduce you a friend that can have a conversation, to answer and advise you everything you need. Even you are not the developer and not good in Machine Learning, deep learning..etc.. You still can make your own chatbot by using Rasa

Overall architecture

In the year 2009, a company called WeChat in China created a more advanced Chatbot. Since its launch, WeChat has conquered the hearts of many users who demonstrate an unwavering loyalty to it. It is a highly thriving social media platform.

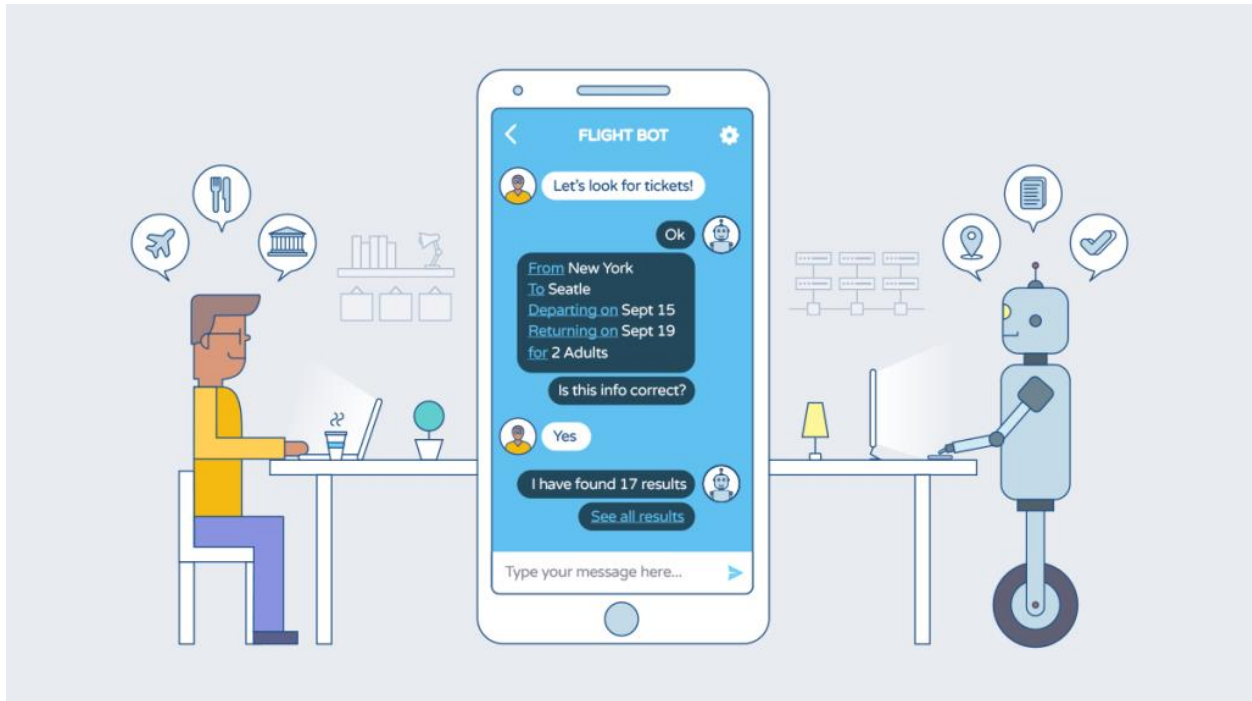
In this blogpost, we are going to understand what is Chatbot and how to build it using framework RASA. I will describe what it is and why we should use it

This article is structured as follows –

- What is chatbot?
- Why we need to build a contextual chatbot? Five steps to build own chatbot
- What is rasa?
- Implement chatbot using rasa

Let's take a lookback in your daily life.

You've probably interacted with a [chatbot](#) whether you know it or not. For example, you're at your computer researching a product, and a window pops up on your screen asking if you need help. Or perhaps you're on your way to a concert and you use your smartphone to request a ride via chat. Or you might have used voice commands to order a coffee from your neighborhood café and received a response telling you when your order will be ready and what it will cost. These are all examples of scenarios in which you could be encountering a chatbot.



At the most basic level, a chatbot is a computer program that simulates and processes human conversation (either written or spoken), allowing humans to interact with digital devices as if they were communicating with a real person. Chatbots can be as simple as rudimentary programs that answer a simple query with a single-line response, or as sophisticated as digital assistants that learn and evolve to deliver increasing levels of personalization as they gather and process information.

Why we need a contextual chatbot?

Simply, contextual chatbot is a bot that can help you answer/doing/ listen that based on the context. Imagine you want to order the pizza in 4 sentences

You: I want to order pizza

Bot: Which pizza do you want?

You: Pizza with mozzarella

Bot: Okay, I have order it

Sound okay, right? You have order it. But at that time, the restaurant call you and say that the pizza with mozzarella is sold out. You have to order again. Now you have to talk a bot again and order again? Simply you can call the restaunt to order it fast. But if this happen so many times? How we can solve it? That's the reason why we need a chatbot based on the context. It can make a conversation like this:

You: I want to order pizza

Bot: Which pizza do you want? Sincerely, I saw that you have order the pizza with mozzarella

You: Yes, I want it

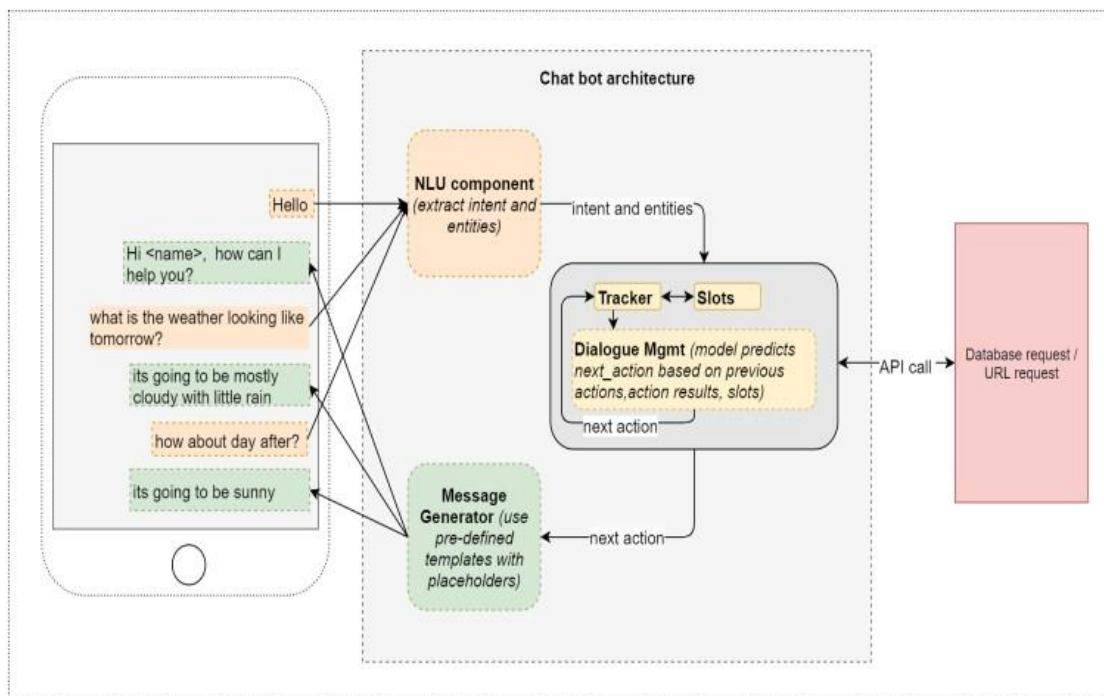
Bot: As the menu in our fanpage, it sold out. Can you order the different?

You: Oh, thank you. I think I will order normal pizza with bacon.

Do you see the different? The bot based on the context that we used to talk to it, and it also based on the menu that's available you can order. That's the reason why we need contextual chatbot.

Five steps to build own chatbot:

As usually, every chatbot and even human always based on the intents. And we can say that this is the core of building chatbot. And when the word in the question/ sentences is repeat all the time, we easily to have the entity of the user. To make a general look at chatbot, look at the following picture



Hình 4: Mô hình các thành phần xử lý trong Chatbot [1]

A basic chatbot can have so many things, but it must have at least 3 compulsory components:

- Natural Language Understanding(NLU): It make chatbot understanding what we teach for our chatbot by processing natural language. More over it will extract intents and entites from user's question
- Dialogue Management(DM): Predict the action of chatbot based on previous action, action result or from intents and entites.
- Natural Language Generator(NLG): It make messages generator from policy and action that identify from DM. From that it can make a answer for the question or can based on the pre-define template that we train for our bot.

Step 1: Building our intents and entites set

Step 2: Build a template-base generator to generate the answer

Step 3: Build a stories for our chatbot

Step 4: Training our NLU model

Step 5: Testing on API and check the result of chatbot doing

What is Rasa?

Rasa is an open source machine learning framework for building [AI assistants and chatbots](#). Mostly you don't need any programming language experience to work in Rasa.

Advantages of Rasa:

1. **Integration**- being open-source and with the ease of building chatbots, Rasa can be integrated into systems easily and automate stuff.
2. **Customization** — who said we cannot add our touch? Rasa is flexible and can be modified to fit needs.
3. **Interactive learning**- Rasa is trained to learn on its own. It doesn't matter if businesses don't know how to train the AI, it learns on its own as you talk to it. Talk about AI being so cool!

4. **Not your regular static machine**- Rasa is not a machine but a conversational AI bot which is the replica of the human interaction mechanism. You could expect the bot to be extremely interactive learning from the chats.

Let's have a walk-through of the implement chatbot

Setup

1. Install Anaconda
2. Create Anaconda environment with python version 3.7
3. Setup rasa and some module needed, recommend you should use rasa 1.10.3

```
pip install rasa==1.10.3
pip install -U tables==3.6.1
pip install -U ipython
python -m spacy download en
pip install nest_asyncio==1.3.3
```

4. Now you have to open the anaconda prompt and run

```
Rasa init
```

Now you can see that in folder you have create that have the file `config.yml`, `data/nlu.md`, `domain.yml`, `data/stories.md`. This is all the file you need to care about it.

5. Open file `data/nlu.md` and write the intent for our training data

this is an example for it

```
version: "3.1"

nlu:
- intent: web_application_development
  examples: |
    - công ty bạn có tạo ứng dụng web không?
    - công ty bạn có ứng dụng web không?

- intent: ask_web_dev
  examples: |
    - ứng dụng web mà công ty của bạn viết trong như thế nào?
    - web công ty bạn làm như thế nào?

- intent: ask_web_exp
  examples: |
    - công ty bạn có bao nhiêu năm kinh nghiệm trong việc tạo ứng dụng web?
    - công ty có kinh nghiệm gì trong việc tạo web?
```

6. Open file `domain.yml` and write the template, action for our output. It will has the structure like this. This is an example

Intents:

- Greeting
- Goodbye
- Thankyou

responses:

- utter_Greeting:
 - text: "hey"
- utter_Goodbye:
 - text: "Bye"
- utter_Thankyou:
 - text: "no problem"

actions:

- utter_Greeting
- utter_Goodbye
- utter_Thankyou

session_config:

session_expiration_time: 60.0

carry_over_slots_to_new_session: true

Note: Remember how many intents that will have how many actions and responses and it's up to you.

7. Config your config.yml

If you use the dataset that's all it English, just ignore it. If you use Vietnamese, change the language to vi and set the epochs:100

8. Now it is so important that you need to do, open the file data/stories.md

Overall, this section has pretty much the same logic as if-else

Anticipating conversation flows and building a script will help our bot handle it more smoothly and appear to be smarter. So, try to think deeply enough about the cases where the user can ask the bot, this is really necessary if the bot gives the expected answer.

Structure of it will be like this:

```
stories: Name of stories 1
steps:
- intent: Name of intent 1
- action: Name of action 1
```

```
stories: Name of the stories 2
steps:
- intent: Name of intent 2
- action: Name of action 2
```

...

Example:

```
stories:
- story: Greeting
  steps:
  - intent: Greeting
  - action: utter_Greeting
  - intent: ask_greeting
  - action: utter_ask_greeting
  - intent: ask_health
  - action: utter_ask_health
  - intent: reply_health
  - action: utter_reply_health
- story: ask_ques
  steps:
  - intent: Greeting
  - action: utter_Greeting
  - intent: ask_ques
  - action: utter_ask__ques
```

9. Now training our data by using command: **Rasa train**

After training, your model will be saved

```
Py 1.22)
quantile_values = np.quantile(
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Finished training component 'UnexpectEDIntentPolicy'.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'CountVectorsFeaturizer' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'CountVectorsFeaturizer' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'DIETClassifier' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'EntitySynonymMapper' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'LexicalSyntacticFeaturizer' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'RegexFeaturizer' from cache.
2022-04-08 15:20:28 INFO rasa.engine.training.hooks - Restored component 'ResponseSelector' from cache.
Your Rasa model is trained and saved at 'models\20220408-151939-frayed-neutron.tar.gz'.
```

10. Now you can test your chatbot by using the command: **Rasa shell**

If u want to connect facebook/zalo/or some different api. You can config the credential.

Rasa support facebook/slack/socketio and mattermost.

When you run the command:

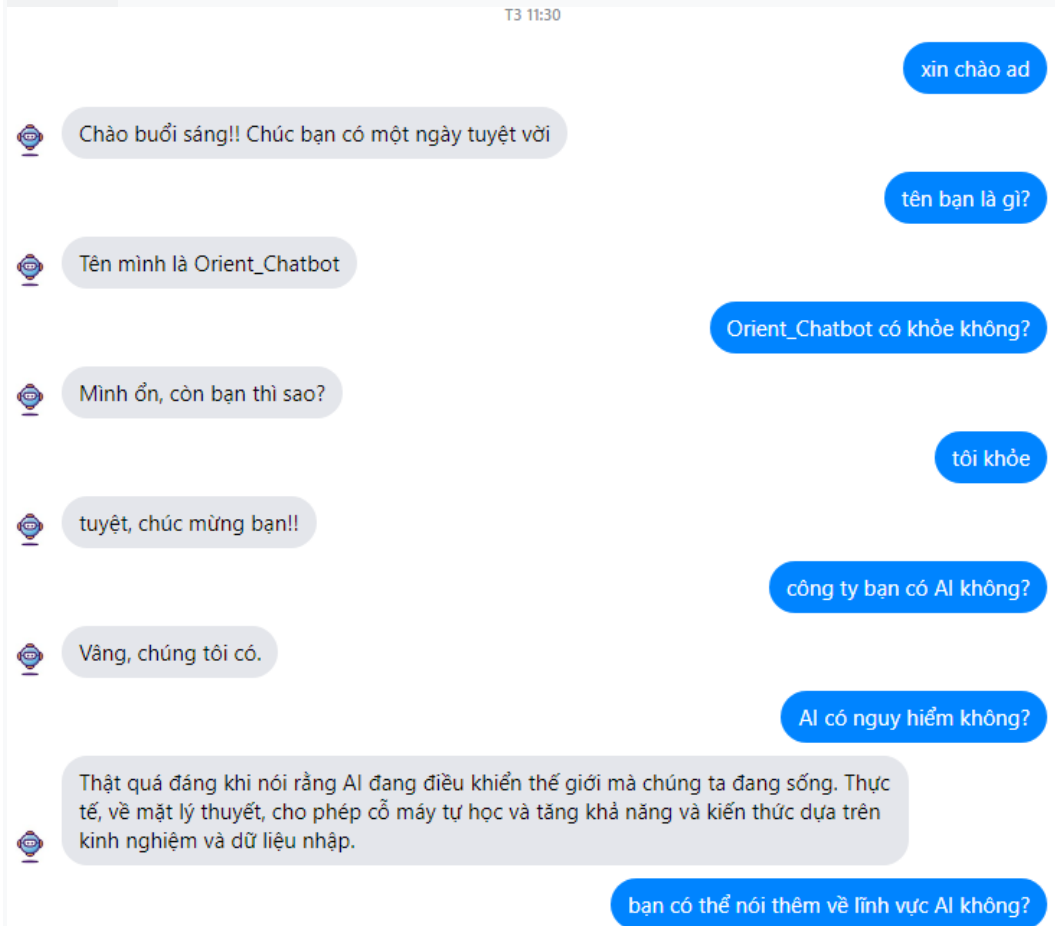
```
rasa run --endpoints endpoints.yml --credential credential.yml
```

After that, the server of rasa will open at: <http://localhost:5005>

You can use the ngrok to connect the api and deploy it in to any framework you want, especially facebook.

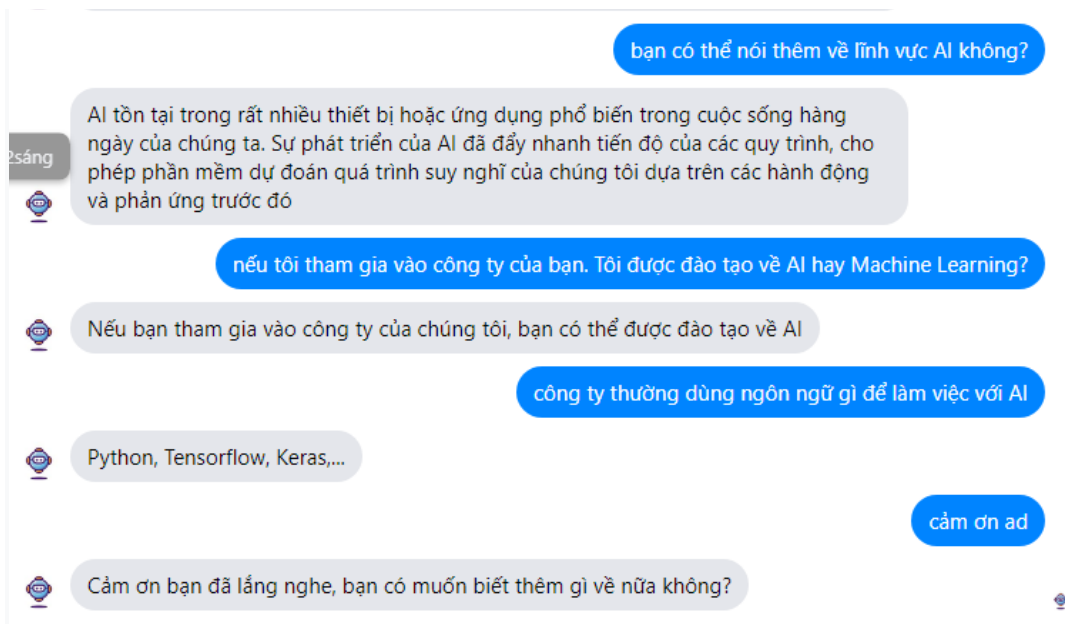
Result

T3 11:30



The screenshot shows a chat interface with a light gray background. On the left, the chatbot's messages are in light gray bubbles with a small robot icon. On the right, the user's messages are in blue bubbles. The conversation is as follows:

- User: xin chào ad
- Chatbot: Chào buổi sáng!! Chúc bạn có một ngày tuyệt vời
- User: tên bạn là gì?
- Chatbot: Tên mình là Orient_Chatbot
- User: Orient_Chatbot có khỏe không?
- Chatbot: Mình ổn, còn bạn thì sao?
- User: tôi khỏe
- Chatbot: tuyệt, chúc mừng bạn!!
- User: công ty bạn có AI không?
- Chatbot: Vâng, chúng tôi có.
- User: AI có nguy hiểm không?
- Chatbot: Thật quá đáng khi nói rằng AI đang điều khiển thế giới mà chúng ta đang sống. Thực tế, về mặt lý thuyết, cho phép cỗ máy tự học và tăng khả năng và kiến thức dựa trên kinh nghiệm và dữ liệu nhập.
- User: bạn có thể nói thêm về lĩnh vực AI không?



At the result, I created a user interface and asked the system several questions and advised.

It's CORRECT!!

Conclusion

In this blogpost, I have explained about chatbot concepts and rasa and why we need a contextual chatbot, also I have guide how to build a chatbot using framework rasa.

Although it can be good but sometimes it not good.

Chatbot has the ability to randomly answer the sentence patterns in the template make the dialogue more natural. Besides the bot has the ability navigating the user to the available answer template. However, navigation is also based on the randomness of the bot leads to the fact that the bot is not smart in handling it situations like this.

