Developing a Chatbot in Python

- Mrityunjaya Singh, Scientist-E
NIC-MoHUA Division
New Delhi

Chatbots

- ***Chatbots*** *are computer program that simulates human conversation through voice commands or text chats or both in natural language, understand the user's intent and send responses based on the application's business rules and data.*
- *They can decipher verbal or written questions and provide responses with appropriate information or direction.*
- *It can be considered as an enhanced channel of user interaction which would move from Interactive Voice Response to Intelligent Assistant Response.*

Types of Chatbots

There are two kinds of chatbots: Scripted bots and Artificial Intelligence (AI) bots.

- *Scripted bot is like a rule-based guided conversation and performs like a decision tree where each action by the user prompts the bot to take action or respond. It engages users with questions about preferences and serves up content and offers relevant to their responses.*
- *AI bots are built on Machine Learning (ML) and Natural Language Processing (NLP) capabilities. They are based on the human capability of learning and absorbing information owing to which they are more efficient and can process much faster than humans and may come up with more subtle results. The more the consumer engages with the bot over time, the smarter the bot becomes about consumer preferences and is able to serve personalized content and offers.*

Types of Chatbot Frameworks

There are different types of Chatbot Frameworks such as –

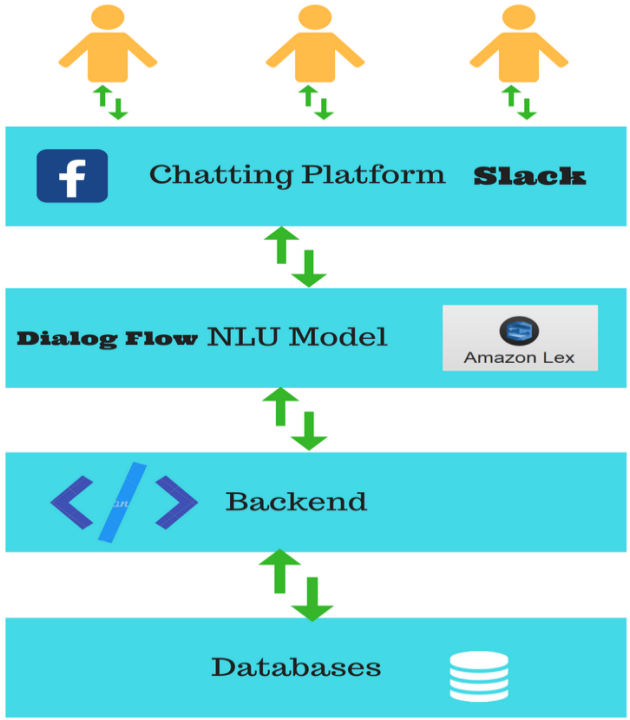
1. Dialogflow 2. Alexa 3. Luis 4. Wit 5. Rasa

Of these frameworks Rasa is an Open Source framework for developing text-and voice-based chatbots and assistants. It comprises of mainly two things, Rasa NLU, and Rasa Core. The benefit of Rasa Stack is data privacy and zero running cost. In other frameworks, for corpus data to be processed through their NLP engine, user chats need to be moved through their servers. And also being open source no cost of using the platform in Rasa, though developers cost will be higher as here you need experienced developers to develop and maintain the chatbot.

One can choose dialogflow or alexa if one need a quick solution. But for highly scalable, data sensitive assistant Rasa Stack is better.

AI Chatbot Architecture

As in the AI bot architecture diagram say for Chatting Platform Slack the chat starts from chatting platform as input request and end at chat platform with an auto response. In between the message passes into so many stages. Here NLU models are chat bot APIs which extract the information from user request. The Web-hook back end is simple web application which provides the restful API. NLU model extract the information in the form of entity which works as parameter in these restful APIs .



AI bot Architecture

AI Chatbot Architecture...

It produces some response that you get as robot response. The only magic comes with the ***Natural Language Processing (NLP)*** *applies in these NLUs. Lets understand it with an example byte – "Order 3 roses for me on coming Monday" . Here NLU should extract three information –*

- ***Product -****Rose*
- ***User –*** *Current Login*
- ***Product Quantity -****3*
- ***Time –*** *Date corresponding to Monday*

If your system is intelligent enough to extract these parameter, You can easily develop the backend for it. Chat bot can save so much time for consumer to understand the product user interface. These NLU API also provide voice interface to user. Once you speak any thing to bot, it first converts speech to text. Once it gets the text, NLP comes into picture.

Chatbots processing Languages!

- *Chatbot can look like a normal app. There is an application layer, a corpus database and APIs to call external services. In a case of the chatbot, UI is replaced with chat interface.*
- *Challenge to a bot is how can it understand the intent of the user. The bots are first trained with the actual data. Developers use logs of conversations already available to analyze what users are trying to ask and what does that mean. With a combination of Machine Learning models and tools built, developers match questions that user asks and answers with the best suitable answer. For example: If a user is asking "Where is my payment receipt?" and "I have not received a payment receipt", mean the same thing.*
- *Developers strength is in training the models so that the chatbot is able to connect both of those questions to correct intent and as an output produces the correct answer. If there is no extensive data available, different APIs data can be used to train the chatbot.*

How Chatbot Actually Works?

The chatbots work by adopting 3 classification methods:

1. Pattern Matching

- *Bots use pattern matching to classify the text and produce a suitable response for the users. A standard structure of these patterns is "Artificial Intelligence Markup Language" (AIML). A simple pattern matching example is:*

```
<aiml version = "1.0.1" encoding = "UTF-8"?>
  <category>
    <pattern> WHO IS ABRAHAM LINCOLN </pattern>
    <template>Abraham Lincoln was the US President during American civil war.</template>
  </category>

  <category>
    <pattern>DO YOU KNOW WHO * IS</pattern>
    <template>
      <srai>WHO IS <star/></srai>
    </template>
  </category>
</aiml>
```

Pattern Matching…

The machine then gives an output:

***Human**: Do you know who Abraham Lincoln is?*

***Robot**: Abraham Lincoln was the US President during American civil war.*

Chatbot knows the answer only because his or her name is in the associated pattern. Similarly, chatbots respond to anything relating it to the associated patterns. But it can not go beyond the associated pattern. To take it to an advanced level algorithms can help.

How Chatbot Actually Works…

2. Algorithms

For each kind of question, a unique pattern must be available in the database to provide a suitable response. With lots of combination on patterns, it creates a hierarchical structure. We use algorithms to reduce the classifiers and generate the more manageable structure. Computer scientists call it a "Reductionist" approach- in order to give a simplified solution, it reduces the problem.

Multinomial Naive Bayes is the classic algorithm for text classification and NLP. For an instance, let's assume a set of sentences are given which are belonging to a particular class. With new input sentence, each word is counted for its occurrence and is accounted for its commonality and each class is assigned a score. The highest scored class is the most likely to be associated with the input sentence.

Algorithms…

For example Sample Training set

class: greeting

“How you doing?”

“good morning”

“hi there”

Few sample Input sentence classification:

input: “Hello good morning”

term: “hello” (no matches)

Term: “good” (class: greeting)

term: “morning” (class: greeting)

classification: greeting (score=2)

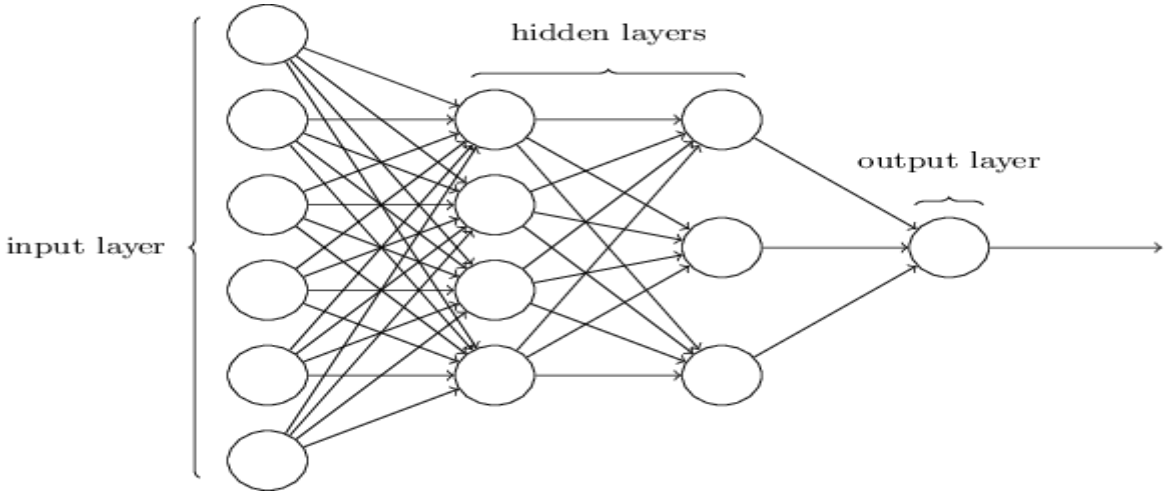
Algorithms…

With the help of equation, word matches are found for given some sample sentences for each class. Classification score identifies the class with the highest term matches but it also has some limitations. The score signifies which intent is most likely to the sentence but does not guarantee it is the perfect match. Highest score only provides the relativity base.

How Chatbot Actually Works…

3. Artificial Neural Networks

Neural Networks are a way of calculating the output from the input using weighted connections which are calculated from repeated iterations while training the data. Each step through the training data amends the weights resulting in the output with accuracy.



Artificial Neural Networks…

As discussed earlier here also, each sentence is broken down into different words and each word then is used as input for the neural networks. The weighted connections are then calculated by different iterations through the training data thousands of times. Each time improving the weights to making it accurate. The trained data of neural network is a comparable algorithm more and less code. When there is a comparably small sample, where the training sentences have 200 different words and 20 classes, then that would be a matrix of 200×20. But this matrix size increases by n times more gradually and computational effort increases. In this kind of situation, processing speed should be considerably high.

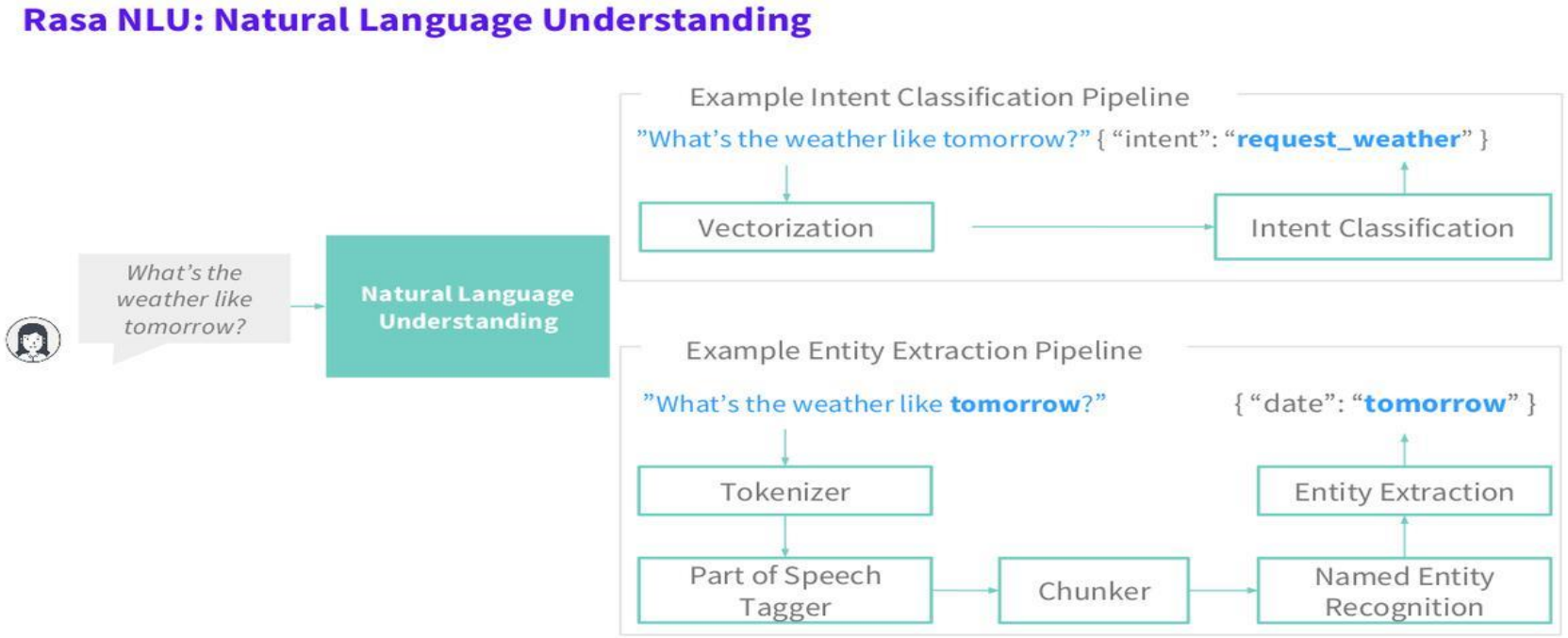
There are multiple variations in neural networks, algorithms as well as patterns matching code. Complexity may also increase in some of the variations. But the fundamental remains the same, and the important work is that of classification.

Natural Language Understanding (NLU)

It has 3 specific concepts like:

- ***Entities**: Entity basically represents a concept in your Chatbot. It might be a payment system in Ecommerce Chatbot, NGO in NGO Darpan Portal etc.*
- ***Intents**: It is basically the action chatbot should perform when the user say something. For instance, intent can trigger same thing if user types "I want to order a red pair of shoes", "Do you have red shoes? I want to order them" or "Show me some red pair of shoes", all of these user's text show trigger single command giving users options for Red pair of shoes.*
- ***Context**: When a NLU algorithm analyzes a sentence, it does not have the history of the user conversation. It means that if it receives the answer to a question it has just asked, it will not remember the question. For differentiating the phases during the chat conversation, it's state should be stored. It can either be flags like "Ordering Pizza" or parameters like "Restaurant: 'Dominos'". With context, you can easily relate intents with no need to know what was the previous question.*

Natural Language Understanding (NLU)



Natural Language Processing (NLP)

Natural Language Processing (NLP) Chatbot takes some combination of steps to convert the user's text or speech into structured data that is used to select the related answer. Some of the Natural Language Processing steps are:

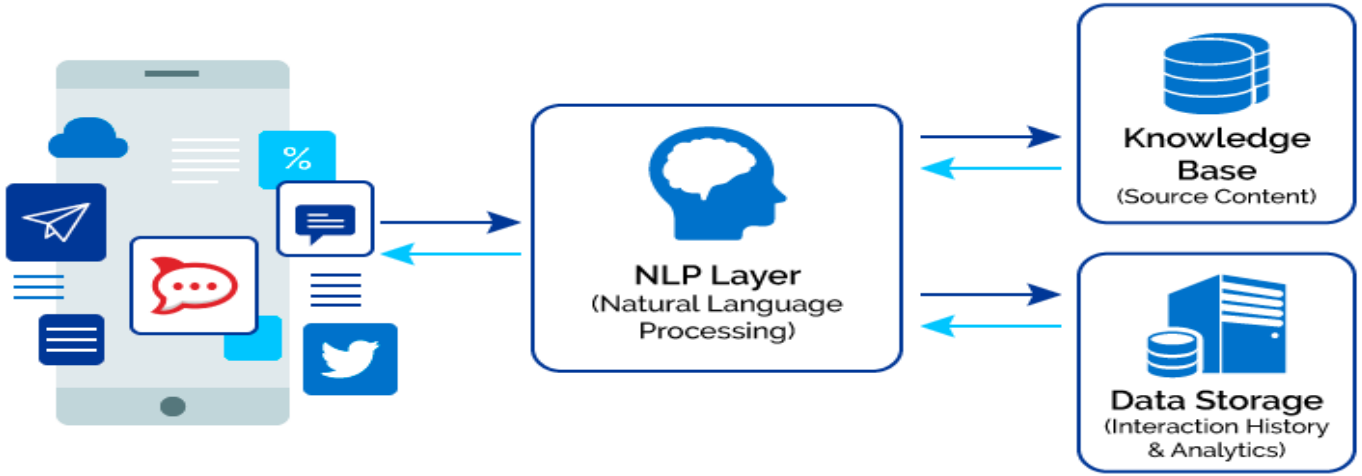
- ***Tokenization:*** *The NLP divides a string of words into pieces or tokens that are linguistically symbolic or are differently useful for the application.*
- ***Named Entity Recognition****: The chatbot program model looks for categories of words, like the name of the product, the user's name or address, whatever data is required.*
- ***Normalization****: The Chatbot program model processes the text in an effort to find common spelling mistakes or typographical errors that might effect what the user intents to convey. This gives more human like effect of the Chatbot to the users.*
- ***Dependency Parsing****: The Chatbot looks for the objects and subjects- verbs, nouns and common phrases in the user's text to find dependent and related phrases that users might be trying to convey.*
- ***Sentiment Analysis****: Tries to learn if the user is having a good experience or if after some point the chat should be forwarded to the human.*

Natural Language Processing (NLP)…

Like most of the Applications, the Chatbot is also connected to the Database. The knowledge base or the database of information is used to feed the chatbot with the information needed to give a suitable response to the user. Data of user's activities and whether or not your chatbot was able to match their questions, is captured in the data store. NLP translates human language into information with a combination of patterns and text that can be mapped in the real time to find applicable responses.

There are NLP services and applications programming interfaces (APIs) that are used to build the chatbots and make it possible for all type of organisations, small, medium and large scale to use chatbot services. The main point here is that Smart Bots have the potential to help increase your base by improving the support services and as a result boosts the efficiency as well as profits for the organisation.

Natural Language Processing (NLP)…



Chatbot with Rasa Stack and Python

RASA stack is an open-source AI tool *and being an open source framework, it is easy to customize. Clients usually do not want to share their data and majority of the tools available are cloud-based and provide software as a service. With RASA,* ***One can build, deploy or host Rasa internally in your server or environment with complete control on it.***

Rasa *comes up with 2 components —*

- ***Rasa NLU —*** *a library for natural language understanding (NLU) which does the classification of intent and extract the entity from the user input and helps bot to understand what the user is saying.*
- ***Rasa Core —*** *a chatbot framework with machine learning-based dialogue management which takes the structured input from the NLU and predicts the next best action using a probabilistic model like LSTM neural network.*

NLU and Core are independent and one can use NLU without Core, and vice versa. Though Rasa recommends using both.

Rasa NLU

Rasa Natural Language Understanding (NLU) comprises of Intent Classification & Entity extraction and create a structured output which can be fed into Rasa Core.

We need to teach our bot to understand our messages first. For that, we have to train the NLU model with inputs in a simple text format and extract structured data. We will achieve this by defining the intents and providing a few ways users might express them.

To make this work, we need to define some files. Lets first understand these files.

- ***NLU training file**: It contains some training data in terms of user inputs along with the mapping of intents and entities present in each of them. The more varying examples you provide, better your bot's NLU capabilities become.*

Rasa NLU…

- ***Stories file**: This file contains sample interactions the user and bot will have. Rasa (Core) creates a probable model of interaction from each story.*
- ***Domain file**: This file lists all the intents, entities, actions, templates and some more information. The templates are nothing but the sample bot reply which can be used as actions.*

Objective

Let us build the Bot (say Newsie) which will help the user to get the news from around the world and also in the specific categories like sports, business, entertainment, technologies and more.

Rasa NLU…

Installation & Setup

Lets first do the environment setup and for that, we need to install python. We can use say Conda (miniconda) to set up the virtual environment for Python.

First, install the Conda (miniconda) as per the OS. After installation check the Conda version.

$ conda --version or conda –V

In case there is a need to upgrade, run bthe following command

$ conda update conda

Once Conda is installed we should create the virtual environment and proceed with further installation of Rasa NLU packages.

Rasa NLU…

Once the package installation is done, Let's create the project structure. We have named the project as newsie and so this is our base project directory. For training/data files, we create a data directory under newsie and create the training file ***nlu.md*** *in that.*

$ mkdir newsie

$ cd newsie

$ mkdir data

$ cd data

$ touch nlu.md

The training data for Rasa NLU is structured into different parts: common examples, synonyms, regex features and lookup tables. While common examples is the only part that is mandatory, including others will help the NLU model learn the domain with fewer examples and also help it be more confident of its predictions.

Rasa NLU…

intent:greet

- hey

- hello

- Hi

intent:fine_ask

- I am good, how are you doing?

- I'm fine, how are you?

- I'm good, how are you?

intent:fine_normal

- I am doing great

- I'm fine

- I'm good

regex:zipcode

[0-9]{5}

Rasa NLU…

intent:news

- Share some latest news around the [world](category)?
- Share some latest news in [sports](category)?
- What is going on in [technology](category)?

lookup:currencies <!-- lookup table list -->

- USD
- Euro

intent:thanks

- Thanks
- Thank you so much

intent:bye

- No, I am good as of now. Bye
- Bye, have a good day

Rasa NLU…

‘## intent:’ defines the name of the intent and [word](entity) defines the entity e.g. — [word](category). We have defined multiple examples for Bot to understand it better.

Now we have to define the pipeline through which this data will flow and intent classification & entity extraction can be done for the bot. ‘spacy_sklearn’ is used as pipeline, created a file nlu_config.yml in the base project directory (i.e. newsie) and added the pipeline in that.

language: "en"

pipeline: "spacy_sklearn“

Now as our NLU data and pipeline are ready, it is time to train the bot. We can do so either by running the script in terminal or we can create a python file and run it.

Rasa NLU…

If using the script in terminal (don't forget to activate the virtual env where all the packages were installed. In our case it is botenv)

$ conda activate botenv

$ cd newsie

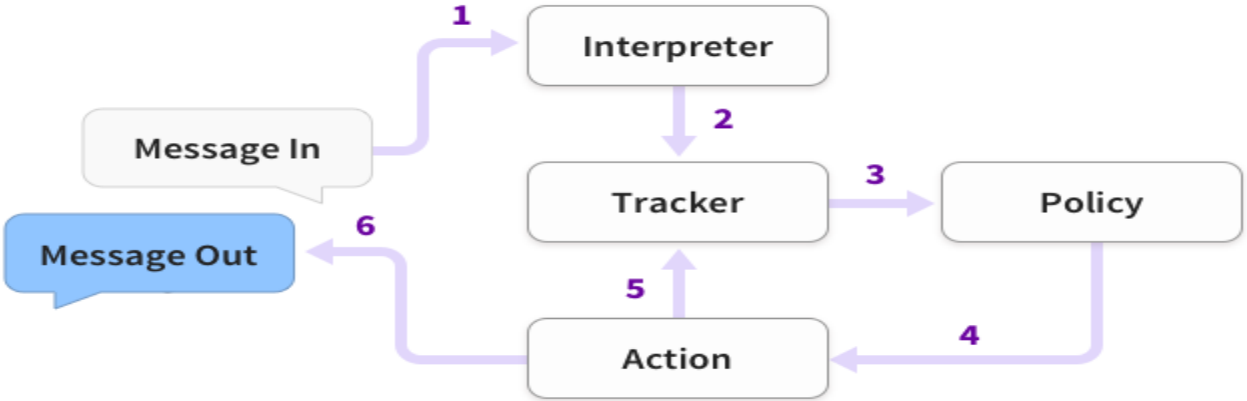
$ python3 -m rasa_nlu.train -c nlu_config.yml --data data/nlu.md -o models --fixed_model_name nlu --project current –verbose

This will train the NLU model and save it at 'models/current/nlu'. The same path is passed to NLU Interpreter to parse some sample intents from the user to see if NLU is able to classify the intent and extract the entities correctly.

Rasa Core

Finally, the training of Bot is considered successful where it is able to understand the natural language but we still need to build the dialogues so that bot can respond to the messages.

After training the bot we need to build a dialogues management for bot to respond to the messages.



Rasa Core High Level Architecture

Rasa Core — Dialog Management

Earlier we talked about the stories which are nothing but the sample interaction between the user and the chatbot. The user's inputs are expressed as intents with corresponding entities, and chatbot responses are expressed as actions.

For dialog training, Rasa has 4 main components —

1. Domain(domain.yml)

The domain consists of five key parts consisting of intents, entities, slots, actions, and templates. We already discussed the first two in the previous part, let's understand the rest.

- ***slots**: slots are basically bot's memory. They act as a key-value store which can be used to store information the user provided (e.g their home city) as well as information gathered about the outside world (e.g. the result of a database query).*

Rasa Core — Dialog Management…

- ***actions****: are nothing but bots response to user input. There are 3 kinds of actions in Rasa Core:* ***default actions, utter actions & custom actions***
- ***templates****: templates are messages the bot will send back to the user.*

Rasa Core — Dialog Management…

2. Stories (stories.md)

Once we have defined our domain now let's create a sample user-bot interaction as below.

User	*intent:greet*	Hi
Bot	utter_greet	Hi, How are you doing?
User	*intent:fine_ask*	I am good, How are you doing?
Bot	utter_reply	I am doing great. Tell me How can I help you today?
User	*intent:news*	Share some news from around the World
Bot	utter_ofc action_get_news	Sure, Here you go. Below are Top 5 news of the World
User	*intent:thanks*	Thanks
Bot	utter_anything_else	No worries, Tell me If I can help you with anything else
User	*intent:bye*	No, I am good as of now. Bye
Bot	utter_bye	Bbye and have a nice day

Rasa Core — Dialog Management…

User	*intent:greet*	Hi
Bot	**utter_greet**	Hi, How are you doing?
User	*intent:fine_normal*	I am doing good. Thanks for asking
Bot	**utter_help**	Great. Tell me what can I do for you today?
User	*intent:news*	Tell me some news about Sports
Bot	**utter_ofc** **action_get_news**	Sure, Here you go. Below are Top 5 news of the Sports
User	*intent:thanks*	Thanks
Bot	**utter_anything_else**	No worries, Tell me If I can help you with anything else
User	*intent:bye*	No, I am good as of now. Bye
Bot	**utter_bye**	Bbye and have a nice day

Lets put it down in a file. Let's name this file as stories.md and kept it in the data directory where we kept the nlu.md earlier.

$ cd newsie

$ cd data

$ touch stories.md

Rasa Core — Dialog Management…

3. Policies (policy.yml)

The rasa core policies decide which action to take at every step in the conversation. There are different policies to choose from, and one can include multiple policies in a single rasa core Agent but at every turn, the policy which predicts the next action with the highest confidence will be used.

We have configured a basic policy(policy.yml) for our bot as shown below which has FallbackPolicy as well. The fallback policy comes in to picture when 'nlu_threshold' & 'core_threshold' meets the levels defined in the policy which means that bot is not able to understand the user message and it responds with 'utter_default'.

Rasa Core — Dialog Management…

4. Custom Actions (actions.py)

We have already defined the utter action by adding an utterance template to the domain file but if we need to run any code we want then we would need custom actions. Custom actions can do anything that you can imagine like turning on the lights, adding an event to a calendar, check a user's bank balance, etc.

Rasa Core calls an endpoint specified by us when a custom action is predicted. This endpoint should be a web server that reacts to this call, runs the code and optionally returns information to modify the dialogue state.

To specify, our action server we use the endpoints.yml and pass it to the scripts using --endpoints endpoints.yml

```
1.  action_endpoint:
2.      url: "http://localhost:5055/webhook"
```

Rasa Core — Dialog Management…

Now run below command to run the action. Keep it running as when we will run the core it would need action to be up and running (run in separate terminal).

$ conda activate botenv

$ cd newsie

$ python3 -m rasa_core_sdk.endpoint --actions actions

Now we have created all the required files/components needed for dialog management. Its time to train the dialog and run the Rasa Core to start the bot for the conversation.

First, we will train the dialogue (Make sure to activate the virtual environment (botenv) we created in the previous part)

Rasa Core — Dialog Management…

$ conda activate botenv

$ cd newsie

$ python3 -m rasa_core.train -d domain.yml -s data/stories.md -o models/dialogue -c policy.yml

then run the core

$ python3 -m rasa_core.run -d models/dialogue -u models/current/nlu --endpoints endpoints.yml

If using python then we can do both in one python file. We have created a python file dialogue_model.py and one can simply execute this file to train and run the dialogue.

$ conda activate botenv

$ cd newsie

$ python3 dialogue_model.py

Rasa Core — Dialog Management…

When python file gets executed, you will see the message that "Your bot is ready to talk! Type your message or send 'stop'". Some actual conversation with the bot looks like as below. Make sure to check the logs (nlu_model.log & dialogue_model.log) to get more idea on how bot classifies the intents and extracts the entities and then make a call to custom actions which eventually call a 3rd party APIs.

****