Internship Report – Career Bot

Overview of Rasa: -

Rasa is an open source machine learning framework for building contextual AI assistants and chatbots.

Rasa has two main modules:

- **NLU –** NLU is for understanding user messages
- **Core –** Core is for holding conversations and deciding what to do next

Setting up and configuring Rasa: -

Rasa Installation:

The following command will install Rasa NLU as well as spacy and its language model for the English

```
pip install rasa
python -m spacy download en_core_web_md
```

Creating Model with Rasa: -

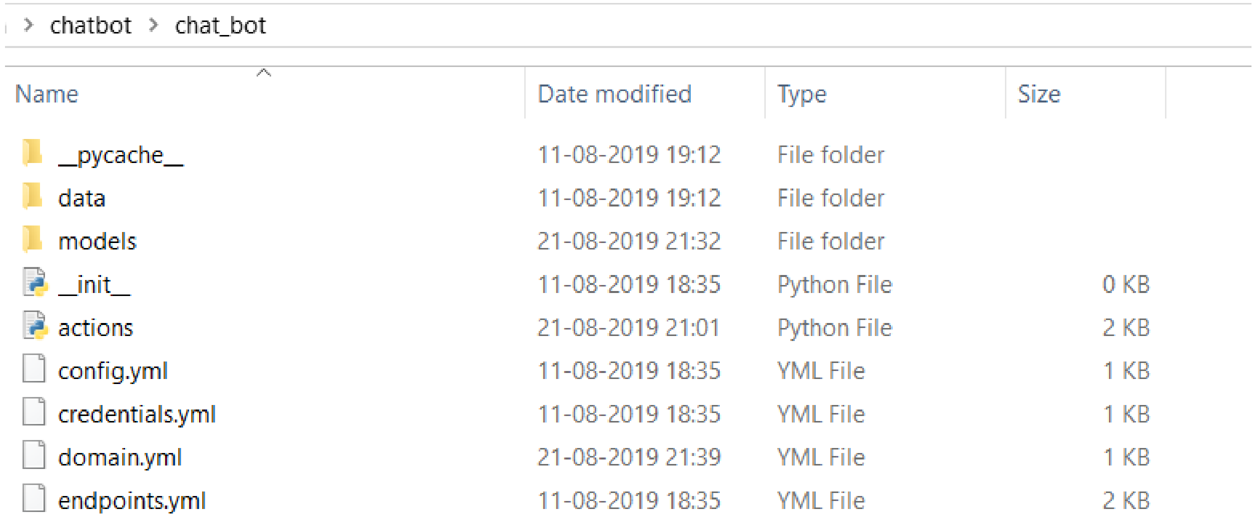
Creating a Rasa Project –

To create a Rasa Project run the command as follows –

```
rasa init
```

After running the above command, the following files will be created.

1) init.py → an empty file that helps python find your actions
2) actions.py → code for your custom actions
3) config.yml → configuration of your NLU and Core models
4) credentials.yml → details for connecting to other services
5) data/nlu.md → our NLU training data
6) data/stories.md → our model stories
7) domain.yml → our assistant’s domain
8) endpoints.yml → details for connecting to channels like fb messenger
9) models/<timestamp>.tar.gz → our initial model



› chatbot › chat_bot

Name	Date modified	Type	Size
__pycache__	11-08-2019 19:12	File folder	
data	11-08-2019 19:12	File folder	
models	21-08-2019 21:32	File folder	
__init__	11-08-2019 18:35	Python File	0 KB
actions	21-08-2019 21:01	Python File	2 KB
config.yml	11-08-2019 18:35	YML File	1 KB
credentials.yml	11-08-2019 18:35	YML File	1 KB
domain.yml	21-08-2019 21:39	YML File	1 KB
endpoints.yml	11-08-2019 18:35	YML File	2 KB

<u>Configuring the data for the Model</u> –

We can provide the training data for the model in markdown file or in JSON file.

Markdown file: Markdown is the easiest Rasa NLU format for humans to read and write.

JSON file: A JSON file is a file that stores simple data structures and objects. It contains data in a standard data interchange format which is lightweight, text-based, and human-readable.

Creating Model with Rasa: -

<u>Training Data</u> –

NLU.MD –

i. The first piece of a Rasa assistant is an NLU model.
ii. NLU stands for Natural Language Understanding, which means turning user messages into structured data.
iii. To do this with Rasa, we provide training examples that show how Rasa should understand user messages, and then train a model.

```
## intent:goodbye
- bye
- goodbye
- see you around
- see you later

##intent:job
-i want [c++](skill) job
-i want [java](skill) job
```

```
-i want [ML](skill) job
-i want [data science](skill) job
-i want [software development](job) job
- i want [management](job) job
## intent: location
- in [bangalore](location)
- in [chennai](location)
- in [hydrabad](location)
- in [delhi](location)
- in [kolkata](location)
- in [mumbai](location)
- [bangalore](location)
- [chennai](location)
- [hydrabad](location)
- [delhi](location)
- [kolkata](location)
- [mumbai](location)

## intent: amount
- [10000](amount)
- [30000](amount)
- [40000](amount)
- [50000](amount)
- [60000](amount)
- [100000](amount)
- [25000](amount)
- [20000](amount)
- [35000](amount)


## intent:job_search
- part time [job](job)
- full time [job](job)
- how do I get a [job](job)?
- I know [python](skill), find me a [job](job)
- [job](job) for [computer engineering](skill)
- [job](job) for student
- [job](job) for housewife
- [job](job) after B.sc
- I'm interested in [ML](skill), find me a [job](job)
- I'm interested in [data science](skill), find me a [job](job)
- help me to find a [job](job)
- search a [job](job) for [100000](amount) salary
- search a [job](job) for [10000](amount) salary
- search a [job](job) for [200000](amount) salary
- search a [job](job) for [20000](amount) salary
- search a [job](job) for [300000](amount) salary
- search a [job](job) for [30000](amount) salary
- find a job in [10000](amount) salary
- job in [10000](amount) salary
- [job](job) in the field of [chemical](skill)
- [job](job) for [electrical engineer](skill)
- I know [c](skill), find me a [job](job)
- I know [java](skill), find me a [job](job)
- I know [c++](skill), find me a [job](job)
- [job](job) for [CA](skill) students
- [job](job) for [BBA](skill) students
- [job](job) for [MBA](skill) students
- find [job](job) in [Bangalore](location)
- find [job](job) in [Hyderabad](location)
- work from home [job](job)
- find [bank](skill) [job](job)
- My skills are [Python](skill), [R](skill), [MATLAB](skill) which is the best [job](job) should I go for?
- I want to find a [job](job) ,can you find it for me?
- I have upper hand on [Java](skill) and other technical skills
- I want to find a [job](job) for [Data Science](skill) and [Machine Learning](skill) Skills
- [job](job) in [ML](skill),[AI](skill)
- [job](job) for [commerce](skill) students
- [job](job) for [arts](skill) student
- [job](job) for architect
- [job](job) for teacher
- [job](job) for driver
- [job](job) for assistant
- [job](job) for PhD holders
- [job](job) in multinational companies
- find [job](job) in [delhi](location)
- find [job](job) in [Chennai](location)
- any [job](job) available?
- [job](job) in [management](skill)
- [job](job) in [cyber security](skill)
- any software development [job](job)
- [job](job) available
- find a [job](job) for me?
- i need a [job](job)?
-

## intent:skill
- I know to learn [web development](skill)
- I know to learn [c](skill) programming
- I know to learn [html](skill)
- I know to learn [angular js](skill)
- I know  [c++](skill)
- I know to learn [Python](skill)
- I know to learn [Android](skill) development
- I know to study [java](skill)
- I know to study [deep learning](skill)
- I know to learn [machine leaning](skill)
- I know to learn [big data](skill)
- I know to learn [data structures](skill)
- I know to learn [ethical hacking](skill)
- I know to learn [data science](skill)
- I know to learn [sql](skill)
- I know to learn [data minng](skill)
- I know to learn [data analytics](skill)
- I know to learn [php](skill)
- I know to learn [NLP](skill)
- I know to learn [bootstrap](skill)
- I know to learn [CSS](skill)
- I know to learn [jquery](skill)
- I know to learn microsoft [excel](skill)
- I know to learn microsoft [powerpoint](skill)
- I know to learn microsoft [word](skill)
- I know to learn database [management](skill)
- I know to learn [verilo](skill)
- i know [java](skill)
- i know [python](skill),[java](skill),[c](skill)
- i know [c](skill),[c++](skill)
- i know [c++](skill),[java](skill)
- i know [java](skill),[python](skill)
- [java](skill)
- [python](skill)
- [c](skill)
- [c++](skill)
- [ruby](skill)
- [R](skill)
```

DOMAIN.YML –

This file configures the NLU and Core components that our model will use.

```
intents:
  - greet
  - goodbye
  - job_search
  - skill
  - location
  - amount
  - jobs

actions:
- utter_greet
- utter_goodbye
- utter_job
- utter_skill
- utter_location
- utter_amount
- utter_who

entites:
  -skill
  -location
  -job
  -amount

Slots:
  amount:
    type: text
  job_search:
    type: text
  skill:
    type: text
  location:
    type: text

templates:
  utter_who:
  - text: "I am chatbot!!"
  utter_greet:
  - text: "Hello!,how can I help you?"

  - text: "Hey! how can i help you?"

  utter_job:
  - text: "What kind of job you looking for?"

  utter_skill:
    - text: "which kind of skill you have?"

  utter_location:
    - text: "where you want to do job?"
  utter_amount:
    - text: "how much selary you expected?"

  utter_goodbye:
  - text: "Thank you,we will inform you soon"
  - text: "bye bye"
```

stories.md –

i. Our Core model learns from real conversational data in the form of training “stories”.
ii. A story is a real conversation between a user and an assistant.
iii. Lines with intents and entities reflect the user’s input and action names show what the assistant should do in response.

```
## job search

* greet
  -utter_greet

* job_search
  -utter_job

* job
  -utter_skill

* skill
  -utter_location

* location
 -utter_amount

* amount
  -utter_goodbye

## greet

* greet
  -utter_greet


## bye

* goodbye
  -utter_goodbye

## job_2
```

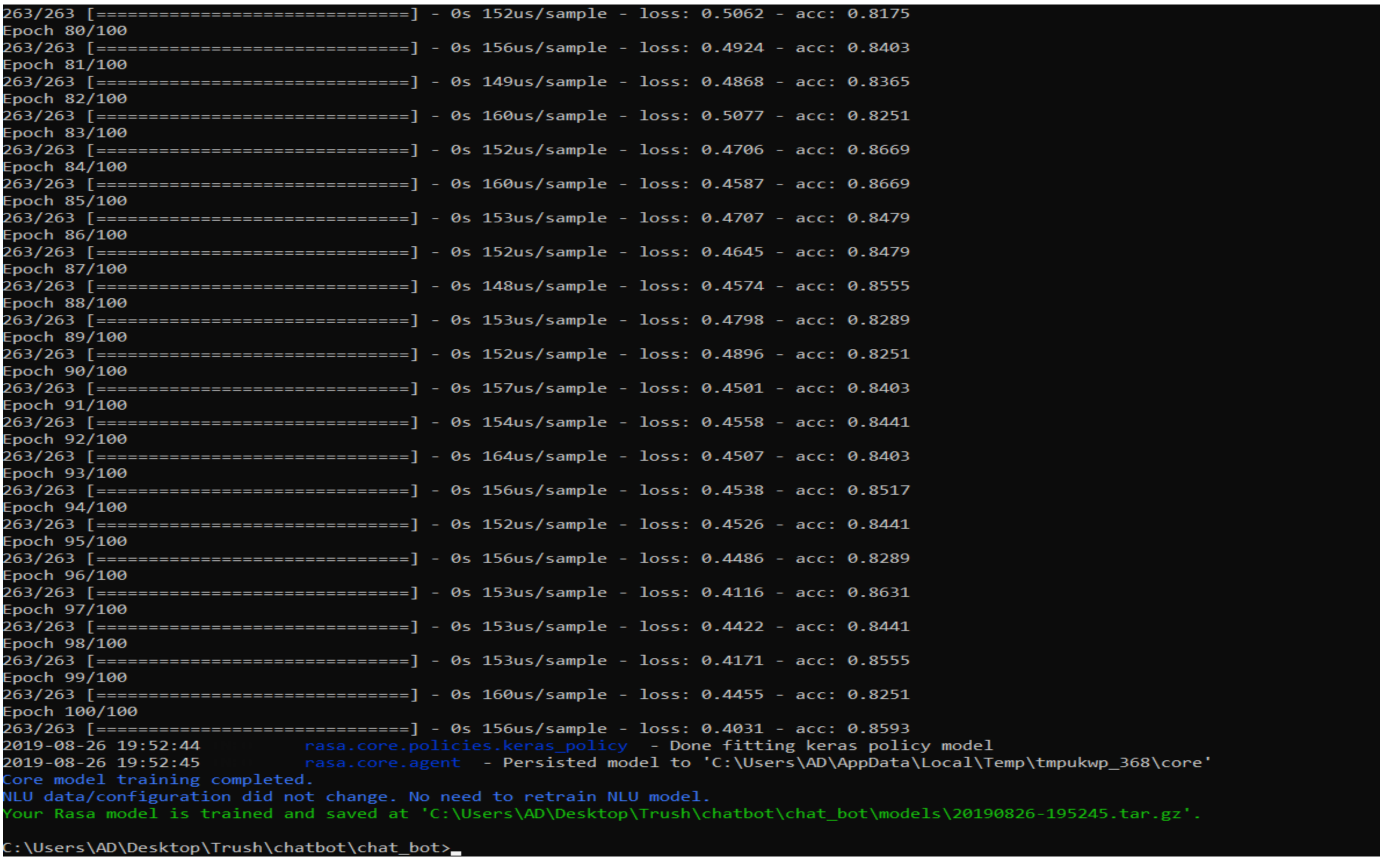
```
* job
  -utter_skill

* skill
  -utter_location

* location
 -utter_amount

* amount
  -utter_goodbye

## job_search_2

* job_search
  -utter_location
*
location
 -utter_amount

* amount
  -utter_goodbye

## job_search_3

* job
  -utter_skill

* skill
  -utter_amount

* amount
  -utter_goodbye

## job_search_4

  * job_search{"amount": "30000"}
  -utter_skill

* skill
  -utter_location

* location
  -utter_goodbye
```

```
## smart_job search

* greet
  -utter_greet

* job_search
  -utter_job

* job{"job": "software development"}
  -utter_skill

* skill{"skill": "python"}
  -utter_location

* location{"location": "chennai"}
 -utter_amount

* amount{"amount": "10000-300000"}
  -utter_goodbye
```

Training the Model: -

The following command will call the Rasa Core and NLU train functions and store the trained model into the models directory.

```
rasa train
```



```
263/263 [==============================] - 0s 152us/sample - loss: 0.5062 - acc: 0.8175
Epoch 80/100
263/263 [==============================] - 0s 156us/sample - loss: 0.4924 - acc: 0.8403
Epoch 81/100
263/263 [==============================] - 0s 149us/sample - loss: 0.4868 - acc: 0.8365
Epoch 82/100
263/263 [==============================] - 0s 160us/sample - loss: 0.5077 - acc: 0.8251
Epoch 83/100
263/263 [==============================] - 0s 152us/sample - loss: 0.4706 - acc: 0.8669
Epoch 84/100
263/263 [==============================] - 0s 160us/sample - loss: 0.4587 - acc: 0.8669
Epoch 85/100
263/263 [==============================] - 0s 153us/sample - loss: 0.4707 - acc: 0.8479
Epoch 86/100
263/263 [==============================] - 0s 152us/sample - loss: 0.4645 - acc: 0.8479
Epoch 87/100
263/263 [==============================] - 0s 148us/sample - loss: 0.4574 - acc: 0.8555
Epoch 88/100
263/263 [==============================] - 0s 153us/sample - loss: 0.4798 - acc: 0.8289
Epoch 89/100
263/263 [==============================] - 0s 152us/sample - loss: 0.4896 - acc: 0.8251
Epoch 90/100
263/263 [==============================] - 0s 157us/sample - loss: 0.4501 - acc: 0.8403
Epoch 91/100
263/263 [==============================] - 0s 154us/sample - loss: 0.4558 - acc: 0.8441
Epoch 92/100
263/263 [==============================] - 0s 164us/sample - loss: 0.4507 - acc: 0.8403
Epoch 93/100
263/263 [==============================] - 0s 156us/sample - loss: 0.4538 - acc: 0.8517
Epoch 94/100
263/263 [==============================] - 0s 152us/sample - loss: 0.4526 - acc: 0.8441
Epoch 95/100
263/263 [==============================] - 0s 156us/sample - loss: 0.4486 - acc: 0.8289
Epoch 96/100
263/263 [==============================] - 0s 153us/sample - loss: 0.4116 - acc: 0.8631
Epoch 97/100
263/263 [==============================] - 0s 153us/sample - loss: 0.4422 - acc: 0.8441
Epoch 98/100
263/263 [==============================] - 0s 153us/sample - loss: 0.4171 - acc: 0.8555
Epoch 99/100
263/263 [==============================] - 0s 160us/sample - loss: 0.4455 - acc: 0.8251
Epoch 100/100
263/263 [==============================] - 0s 156us/sample - loss: 0.4031 - acc: 0.8593
2019-08-26 19:52:44          rasa.core.policies.keras_policy  - Done fitting keras policy model
2019-08-26 19:52:45          rasa.core.agent  - Persisted model to 'C:\Users\AD\AppData\Local\Temp\tmpukwp_368\core'
Core model training completed.
NLU data/configuration did not change. No need to retrain NLU model.
Your Rasa model is trained and saved at 'C:\Users\AD\Desktop\Trush\chatbot\chat_bot\models\20190826-195245.tar.gz'.

C:\Users\AD\Desktop\Trush\chatbot\chat_bot>_
```

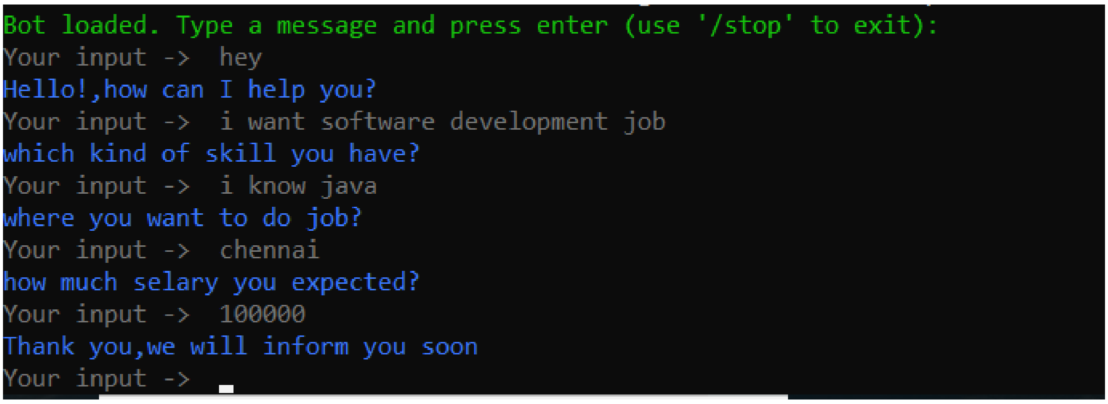
Rasa Architechture: -

- As soon as Rasa model receives a message from the end user, it tries to predict & extract the “intents” and “entities” present in the message. This part is handled by Rasa NLU.
- Once the user intent is identified, the Rasa stack performs an action called action/utter.
- Rasa then tries to predict what it should do next. This decision is taken considering multiple factors and is handled by Rasa Core.

Talking to our Bot: -

The next step is start talking to our assistant run the command as follows

rasa shell



```
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hey
Hello!,how can I help you?
Your input ->  i want software development job
which kind of skill you have?
Your input ->  i know java
where you want to do job?
Your input ->  chennai
how much selary you expected?
Your input ->  100000
Thank you,we will inform you soon
Your input ->  _
```

Scopes of Career bot: -

- It can be at service 24/7.
- When deployed in any application, can handle multiple users simultaneously.
- If user having any queries related to the careers are clarified by this Career bot.

Outcomes of the Career bot: -

- The bot can respond to the queries such as jobs availability, job details, skills enhancement, Exams etc.
- The bot is able to respond to the queries of the users career related.
- More speed and accuracy