ChatBot

Senior Mentor

Pratibha Moogi

Mentor

Amanjot Singh / Taranjeet

Team Members

Parneet Kaur
Kamya Arora
Jaskaran Singh

Introduction to Chatbot

Chatbot is a computer application used to conduct chat conversations with machines in natural language. It is widely used in MNCs that earlier had a large number of human workforce to handle customer support. It is the most advanced and popular application of Human Machine interaction.

Our goal is to make a Data Science Bot that can answer the questions related to the field of Data Science as per the requirements of the user.

The chatbot made by us consists of topics like Machine learning, Data Analysis, Mathematics and Data Visualization. Each topic is further divided into a sub topic, thus giving a great insight of the topic.

System Requirements

Hardware Requirements

System Architecture

- Windows 64 bitX86, 32 bitX86
- Linux 64-bitX86, 32-bitX86
- MAC OS 64-bitX86
- Disk Space - Minimum 8GB

Software Requirements

Windows 8,8.1,10 MAC OS 10.10+ Linux

Python 3.6: Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Rasa - Rasa Open Source is a machine learning framework to automate text- and voice-based assistants. Understand messages, hold conversations, and connect to messaging channels and APIs.

Basic Approaches followed

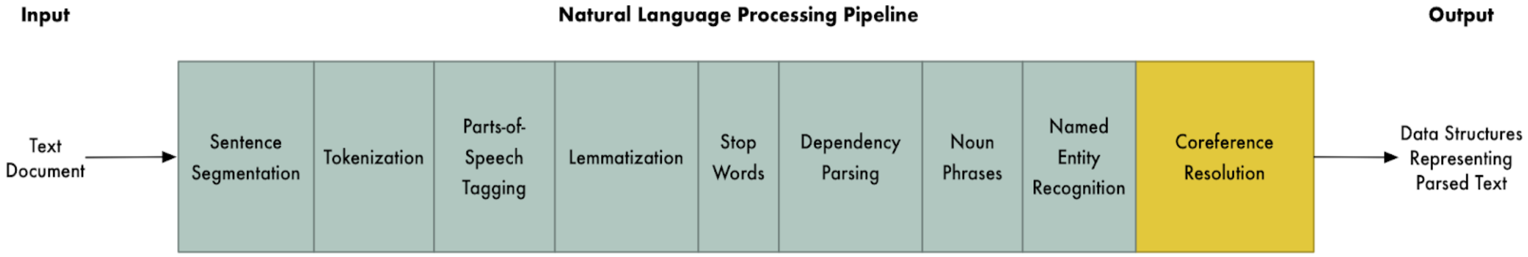
1. **Rule Based Approach**- In a Rule-based approach, a bot answers questions based on some rules on which it is trained on. The rules defined can be very simple to very complex. Thus, a rule based chatbot can solve the problems they are familiar with. The bots can handle simple queries but fail to manage complex ones.

2. **AI Based Approach**- Chatbots can use Artificial Intelligence and machine learning to generate their answers. They analyze users' responses and moods to offer better feedback. These are definitely more efficient than rule-based bots. The main advantage of AI chatbots is the ability to analyze the collected data.

Technologies used in Chatbot

The main technology working in the chatbot is **Natural Language Processing (NLP)**. It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as *translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation*.

Natural Language Processing Pipeline



1. **Sentence Segmentation**- Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

2. **Word Tokenization**- Word Tokenizer is used to break the sentence into separate words or tokens.

3. **Stemming**- Stemming is used to normalize words into its base form or root form. For example, celebrates, celebrated and celebrating, all these words originated with a single root word "celebrate." The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

4. **Lemmatization**- Lemmatization is quite similar to Stemming. It is used to group different inflected forms of the word, called Lemma. The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

5. **Identifying Stop Words**- In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. Stop words might be filtered out before doing any statistical analysis.

6. **Dependency Parsing**- Dependency Parsing is used to find out how all the words in the sentence are related to each other.

7. **POS tags**- POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective. It indicates how a word functions with its meaning as well as grammatically within the sentences. A word has one or more parts of speech based on the context in which it is used.

8. **Named Entity Recognition (NER)**- Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

9. **Chunking-** Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

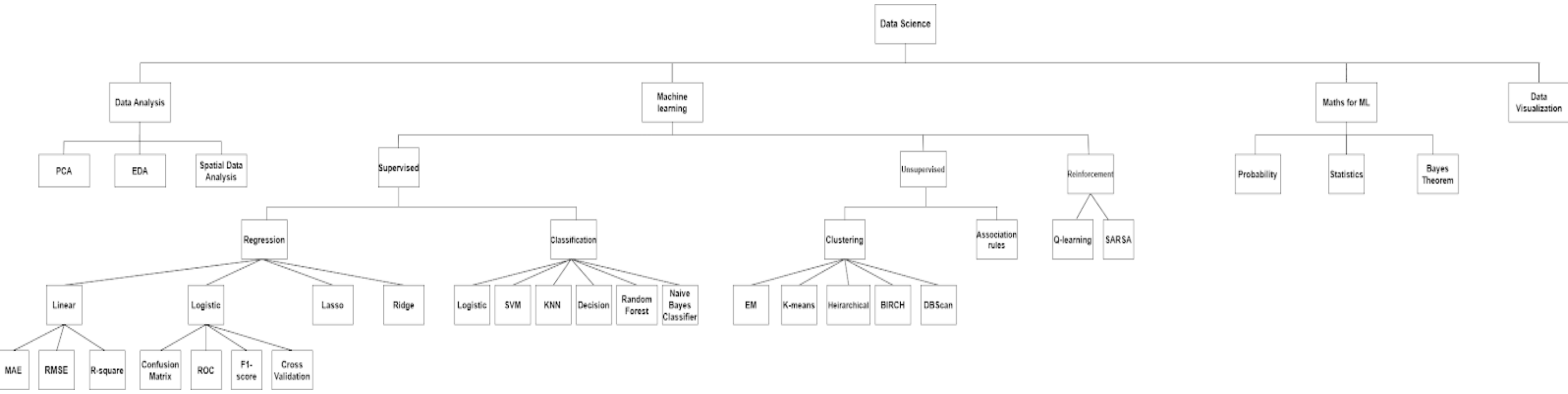
Components of a Chatbot

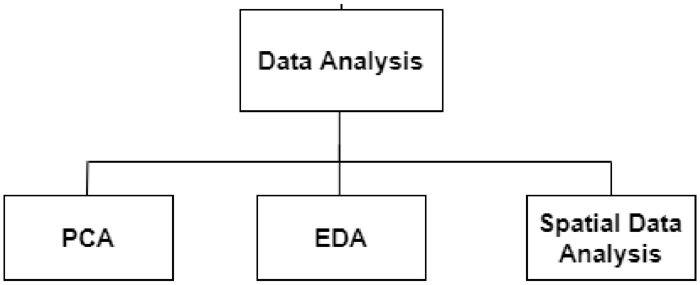
Any chatbot consists of some basic components like Intents, Entities and Utterances.

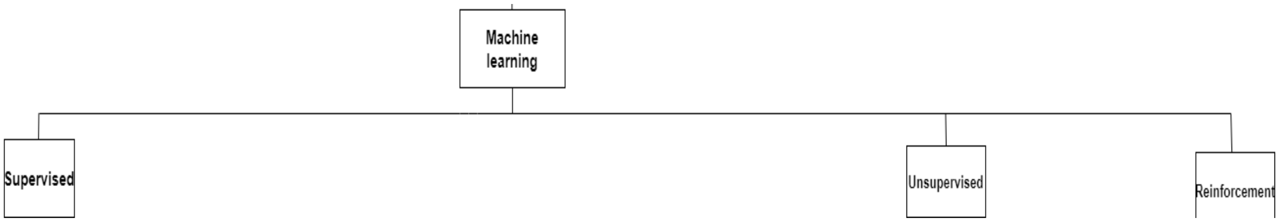
1. **Intents**- Within a chatbot, intent refers to the goal the customer has in mind when typing in a question or comment.
2. **Entities**- Within a chatbot, an entity, or slot, modifies user intent. Chatbot entities are connected to knowledge repositories in order to provide more personal and accurate responses on user search.
3. **Utterances**- Utterances are the input from the user which the chatbot needs to derive intents and entities from. To train any chatbot to accurately extract intents

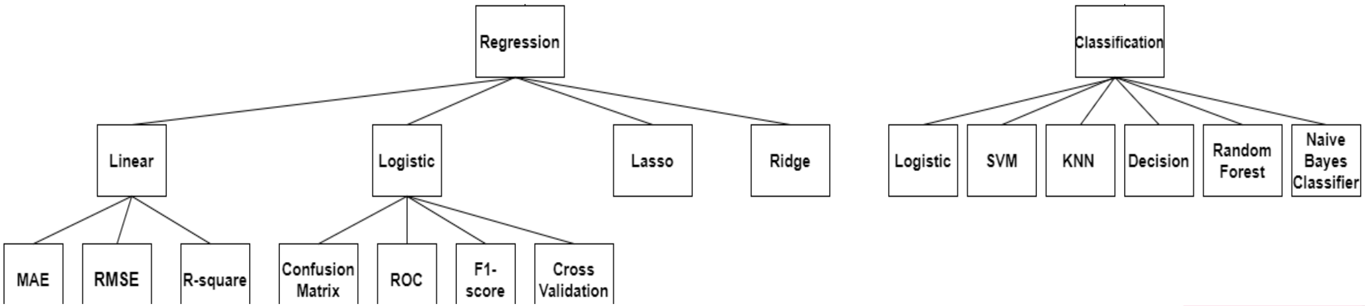
and entities from the user's dialog input, it is imperative to capture a variety of different example utterances for each and every intent.

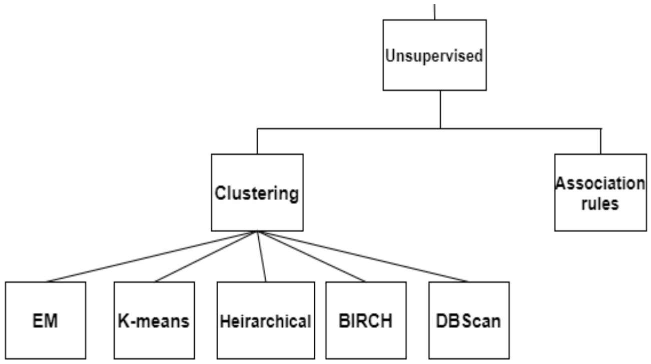
Flow of Intents in Chatbot

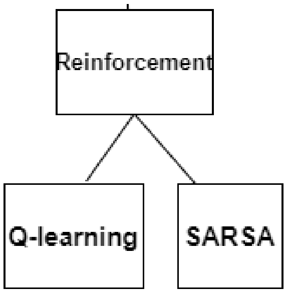


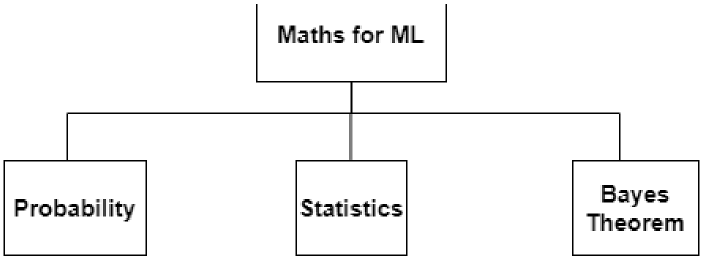


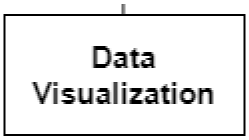




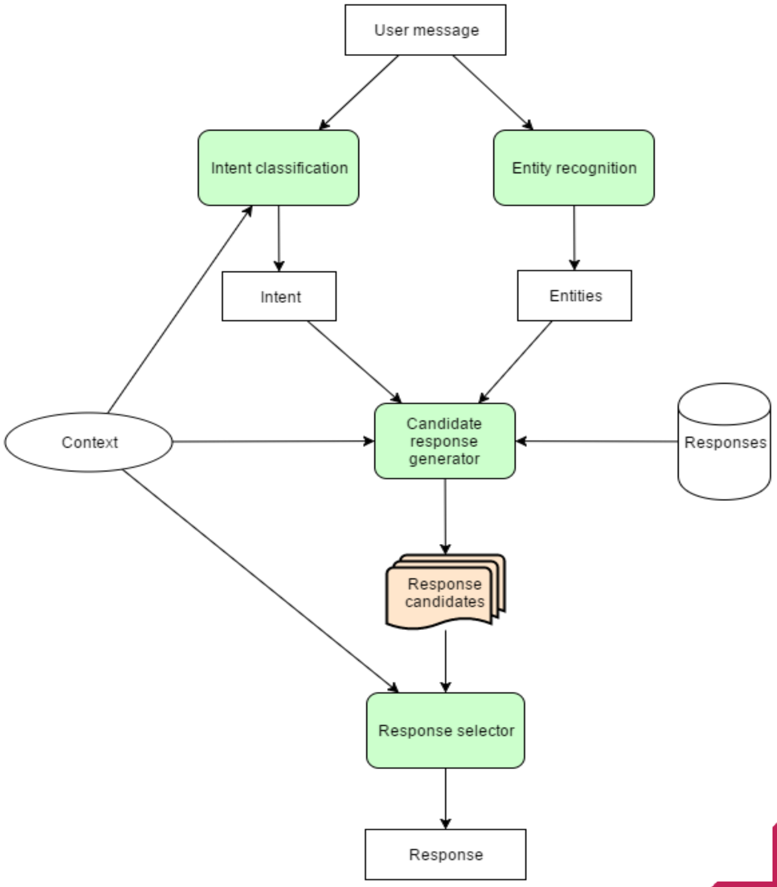








It is very important for a chatbot to give the correct answer as per the user's question. The responses are selected by the chatbot by using the below mechanism:-



Chatbot Frameworks

We have some open source as well as paid frameworks that can be used to make a Chatbot. Some of them are mentioned below:-

1. **Dialogflow**

   Dialogflow is backed by Google and is the most widely used tool to build Actions for more than 400M+ Google Assistant devices. It supports all the major messaging channels such as Facebook Messenger, Slack, Skype, Kik, Line, Telegram, Twitter, Viber etc. Dialogflow supports Natural Language Processing in 20+ languages.

2. **BotKit**

   Botkit is an open source chatbot framework, recently acquired by Microsoft. Botkit is the leading developer tool for building chatbots, apps and custom integrations for major messaging platforms.

   BotKit is NodeJs based SDK which supports publishing the chatbots on messaging channels such as Slack, Cisco Webex, Cisco Jabber, Microsoft Teams, Facebook Messenger Twilio SMS, Twilio IPM, Microsoft Bot Framework, Google Hangouts Chat.

3. **Rasa**

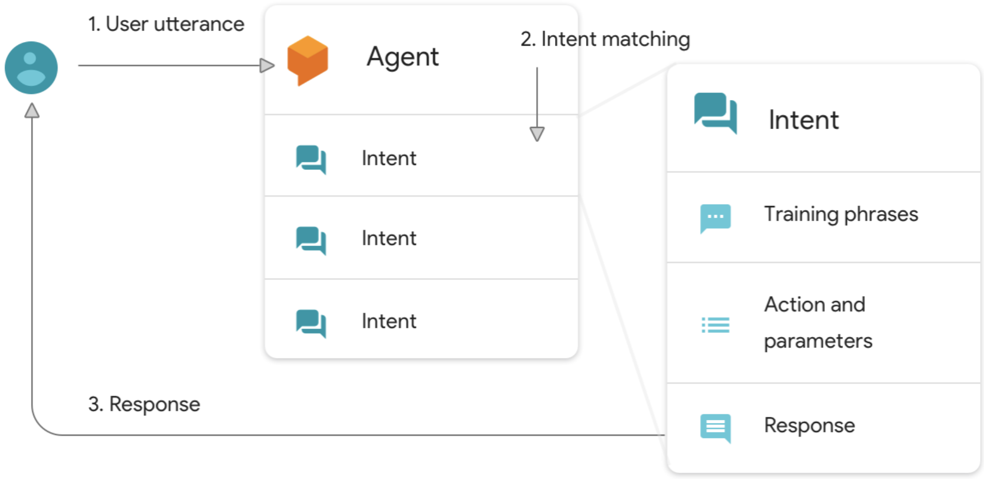
   Rasa is an open source framework. It has two major components Rasa NLU and Rasa Core. Rasa NLU is responsible for natural language understanding. Rasa core is a framework for building conversational chatbot. Rasa core allows more sophisticated dialogue, trained using interactive and supervised machine learning.

   The major advantage of using Rasa Stack is chatbot can be deployed on your own server by keeping all the components in-house. It is possible to use Rasa Core or Rasa NLU separately. Rasa is production ready and used in large companies everywhere. Rasa core supports Facebook Messenger, Rocket.Chat, `Slack, Telegram, Twilio.

4. **IBM Watson Assistant**

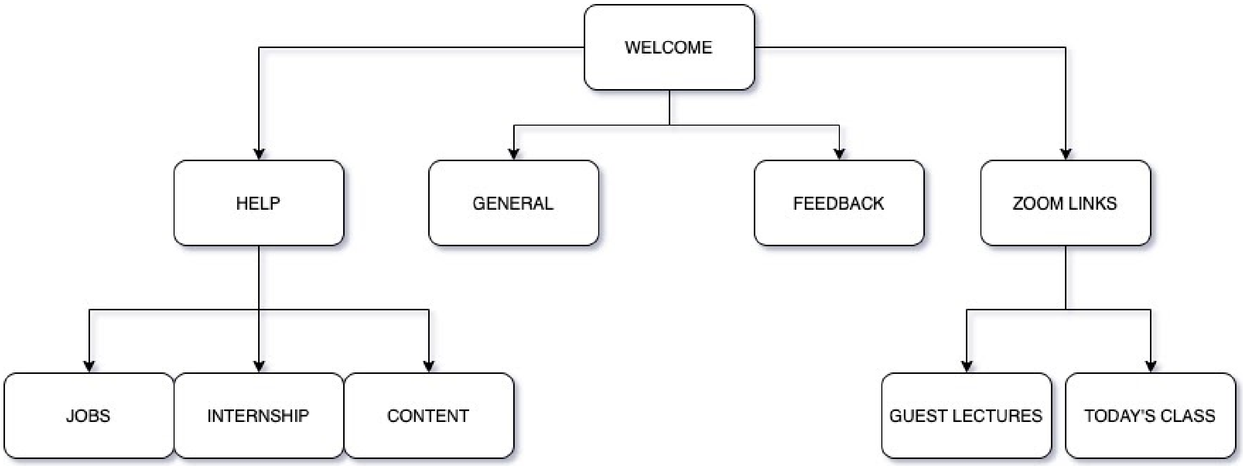
   Watson Assistant is an offering for building conversational interfaces into any application, device, or channel. You have the flexibility to deploy Watson Assistant on your site, in a mobile app, on the phone, in messaging channels, and to customer service tools. It supports 13 languages. It provides SDK for the developers to build applications around Watson Assistant. You can use SDKs in Java, Python, iOS. IBM offers free, standard, and premium plans.

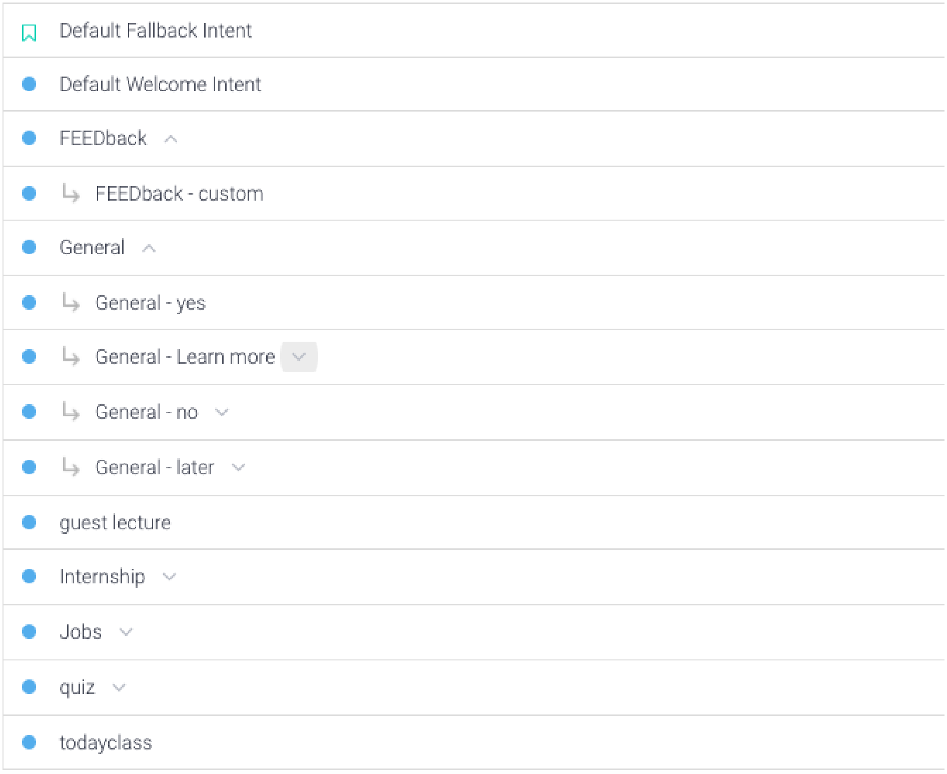
Working with Dialogflow



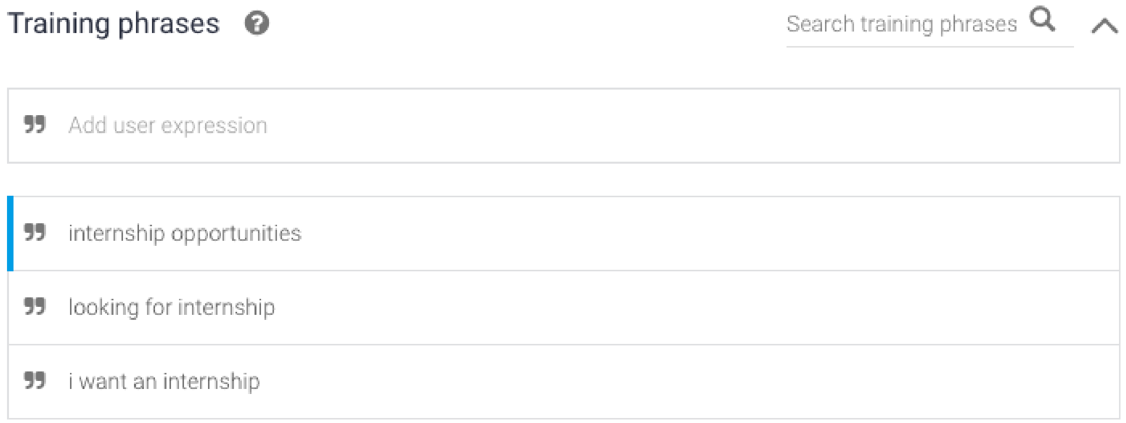
Main Components of Dialogflow

1. **Intents -** It represents a mapping between what a user says and what action our app needs to take. Usually we would have more than one intent in an agent. Think of this as a decision tree and we need to map a user's phrase to a specific outcome/response.



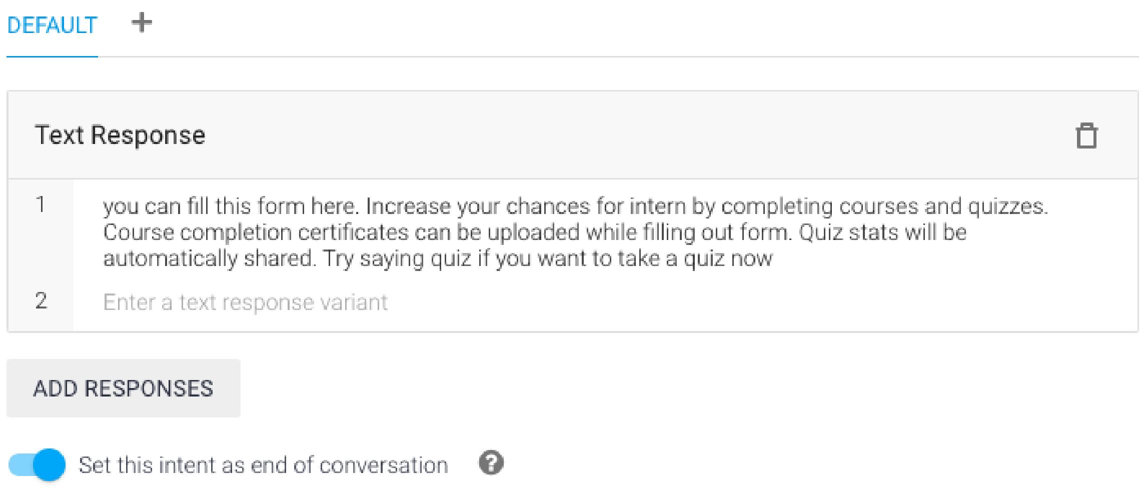


- Default Fallback Intent
- ● Default Welcome Intent
- ● FEEDback
- ● ↳ FEEDback - custom
- ● General
- ● ↳ General - yes
- ● ↳ General - Learn more
- ● ↳ General - no
- ● ↳ General - later
- ● guest lecture
- ● Internship
- ● Jobs
- ● quiz
- ● todayclass



Training phrases

- internship opportunities
- looking for internship
- i want an internship

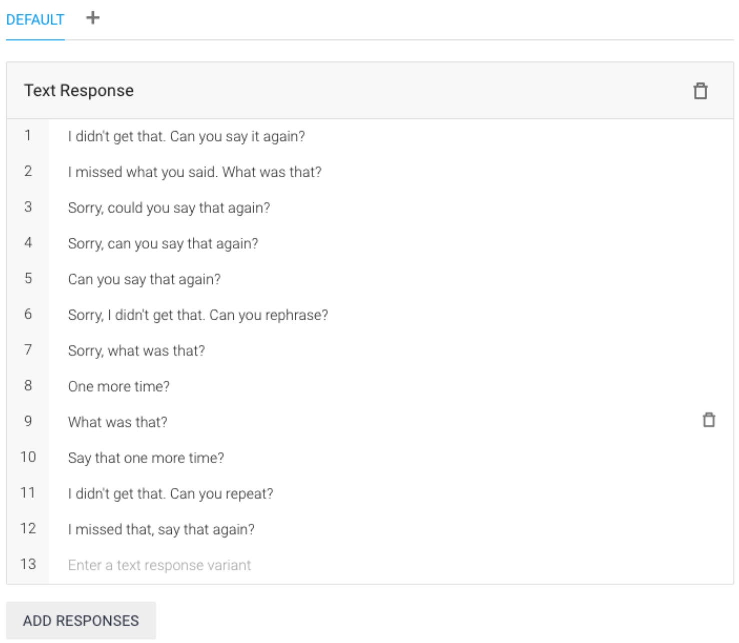


Text Response

1. you can fill this form here. Increase your chances for intern by completing courses and quizzes. Course completion certificates can be uploaded while filling out form. Quiz stats will be automatically shared. Try saying quiz if you want to take a quiz now
2. Enter a text response variant

Set this intent as end of conversation

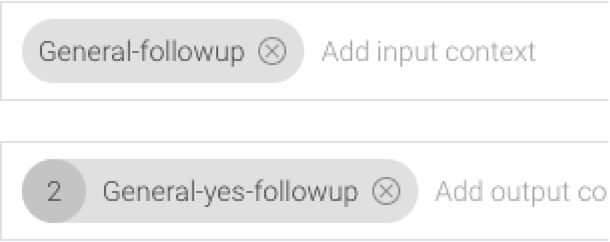
There is also this concept of a fallback intent, in case the agent did not know how to respond to something.



Text Response

1. I didn't get that. Can you say it again?
2. I missed what you said. What was that?
3. Sorry, could you say that again?
4. Sorry, can you say that again?
5. Can you say that again?
6. Sorry, I didn't get that. Can you rephrase?
7. Sorry, what was that?
8. One more time?
9. What was that?
10. Say that one more time?
11. I didn't get that. Can you repeat?
12. I missed that, say that again?
13. Enter a text response variant

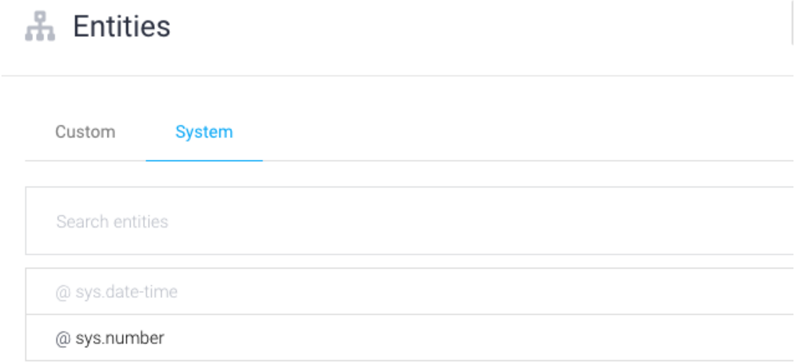
2. Contexts - Contexts guide the conversation because they tell us what we know at any given point.



General-followup ⊗ Add input context

2 General-yes-followup ⊗ Add output co

2. **Entities -** They are the values the user says like location, date, etc. DialogFlow has some predefined entities like address, city, etc..., these are called system entities. There are also the entities we define in our agent, which are called developer entities.



Entities

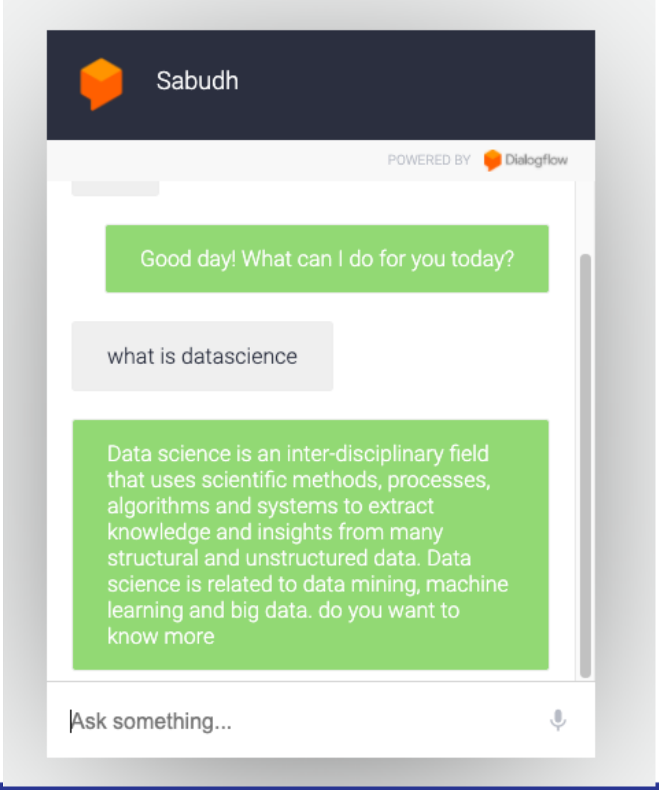
Custom | System

Search entities

@ sys.date-time

@ sys.number

Dialogflow gives a better UI. Here we can see how we set up intents with proper flowchart kind of view. And it gives a direct website integration demo too



Sabudh

POWERED BY Dialogflow

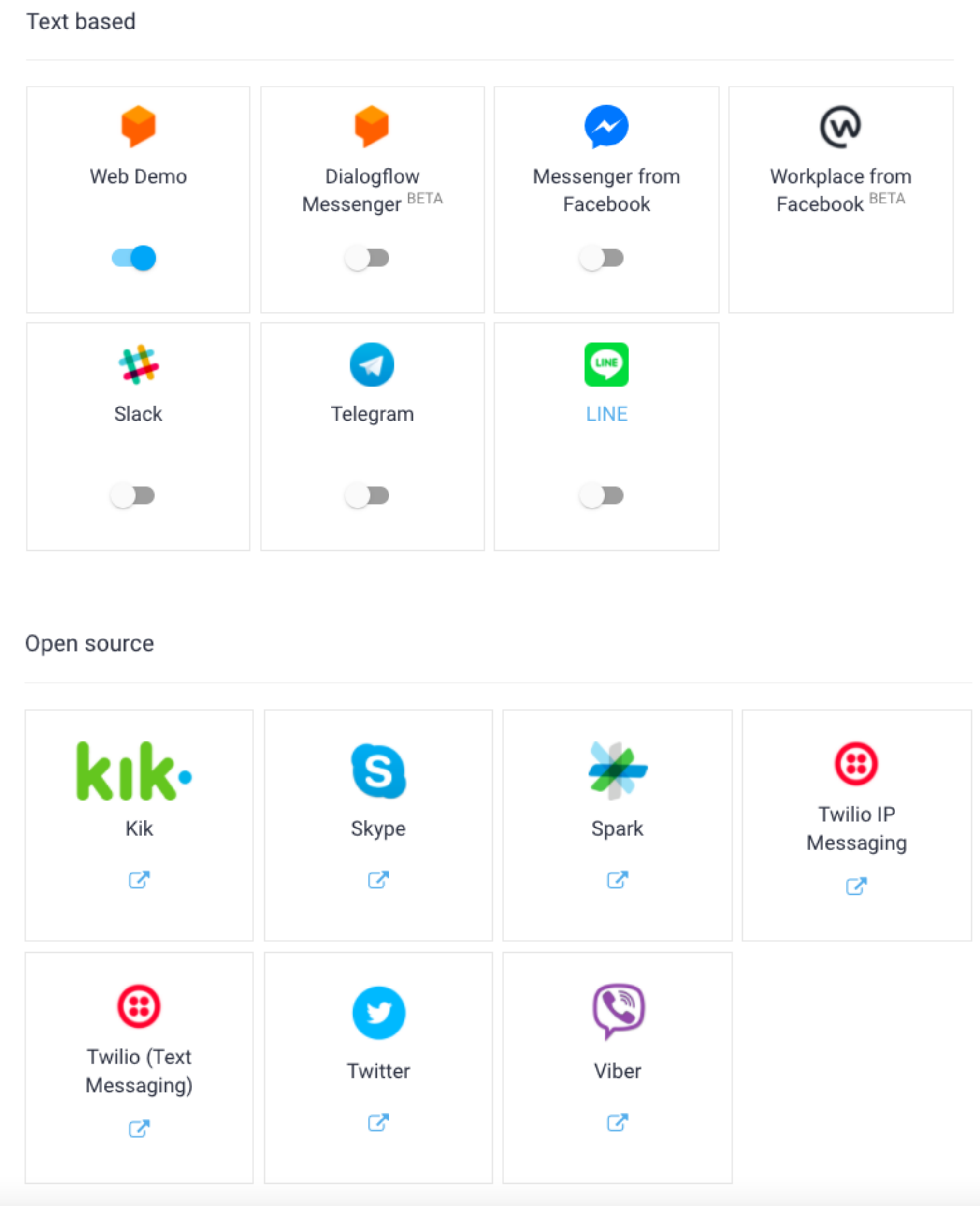
Good day! What can I do for you today?

what is datascience

Data science is an inter-disciplinary field that uses scientific methods, processes, algorithms and systems to extract knowledge and insights from many structural and unstructured data. Data science is related to data mining, machine learning and big data. do you want to know more

Ask something...

Some other integrations offered by Dailogflow are shown below:-



Disadvantages of using Dialogflow

1. **Flexibility and Developer experience-** Although Dialogflow is fairly intuitive on the surface, you'll find that it's not as flexible a platform as you would have hoped for. For example, if you decide that I want to move a follow-up

response under a different Intent, you cannot simply drag that under the desired Intent. Instead, you'll need to delete the existing Intent, create a new Intent in a different location, and re-type all of the training phrases you've already created. This poses a problem because it results in a lot of tedious repetitions which are quite frankly a waste of time and it forces developers to think well in advance about the hierarchical dialog flow (no pun intended) of their conversation. This means there's limited flexibility to change things you create in the future, which needless to say is a major issue.

2. **Limited Webhooks and Integrations available-** You can only provide one webhook for each project. This essentially means that the entire chatbot must have exactly one webhook instead of choosing multiple webhooks on an intent-by-intent basis.

3. **Lots of manual work and Training required-** In many instances, Dialogflow makes it harder than it should be to automate processes and expand your conversational agent's learning. This can get annoying because you have to input many things manually, especially when you consider the need to train your bot over time. Therefore, the ability to even make your chatbot better is hindered, which unfortunately defeats one of the platform's main purposes.

Working with Rasa

Advantages of Rasa over other Frameworks

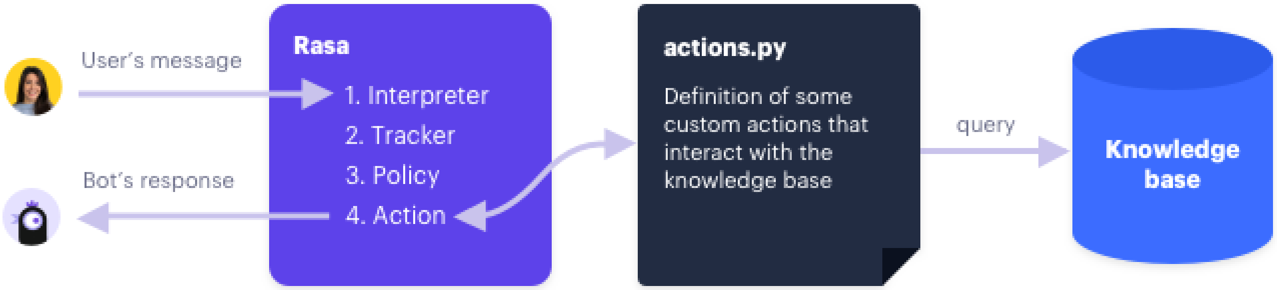
1. **Ease of Access-** With Rasa you do not need a specialized hardware, GPU's etc.There is no mandatory cloud involved. You can run it locally on your machine.

2. **Hiding Complexity-** Most chatbot environments grow hugely in complexity as they mature and add functionality. A case in point is Amazon's Alexa Console and especially Alexa Conversations. Also IBM Watson Assistant. Rasa-X & CDD fall into the same category. As Rasa grows in functionality, the user environment actually becomes more simplistic and intuitive. To achieve this is no easy feat.

3. **Conversation driven development-** Rasa has taken a very innovative and unique approach to continuous improvement of chatbots. This approach they have named Conversation-Driven Development (**CDD**). The enabler or tool for CDD is called Rasa X.

4. **Dialog management in Rasa-** You cannot write rules for every single possible dialog path. The juxtaposition is evident, we want to introduce Conversational AI, but with a huge set of rules to manage our conversational tree. Rasa uses machine learning to learn conversational patterns and predict response; based on the context etc. Training data is in the form of examples of conversations defined in a simple format.

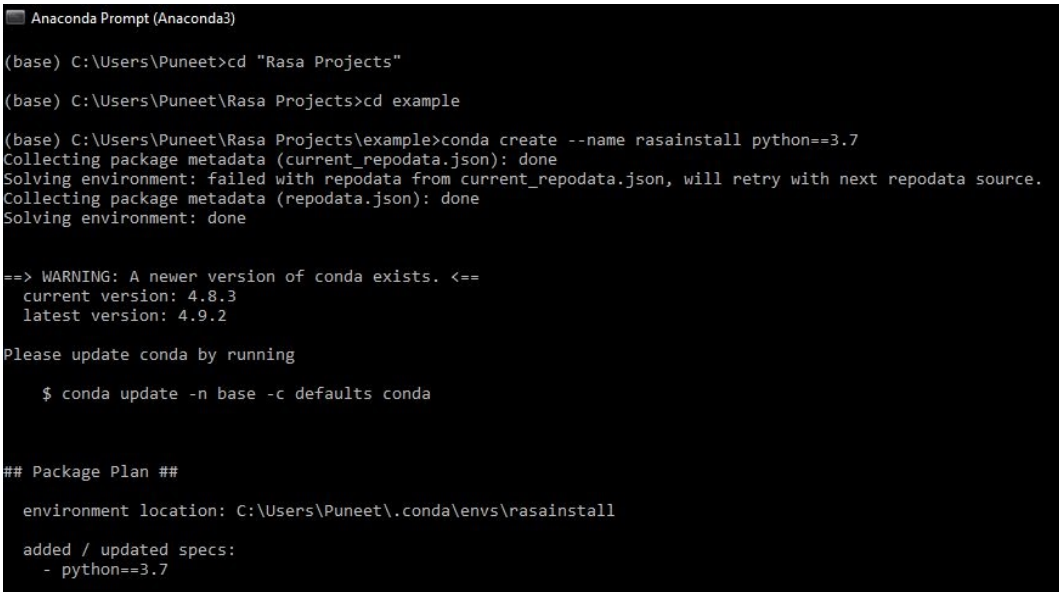
5. **Ease of configuration and change of pipeline-** You can fully customize your NLU pipeline by combining components in the *config.yml* file. You can change this on the fly to test performance. So, you can choose a starting point, and from there customize and adjust your configuration.



Installation Of Rasa

Virtual Commands for Rasa

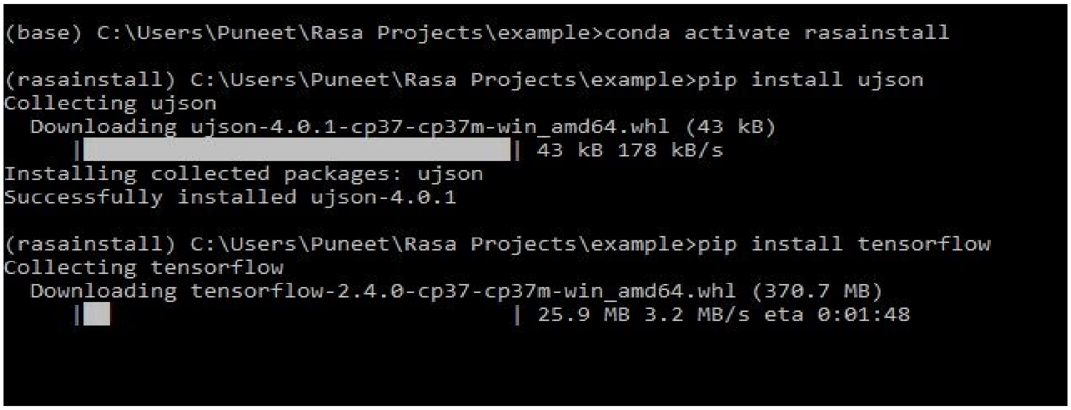
1. You can install Rasa Open Source using pip (requires Python 3.6, 3.7 or 3.8). In Anaconda terminal, prior installation, move to the folder where you want to install rasa. Give name to your rasa environment and specify version.Command for the same is ***conda create --name <name of env> python==<version of python>***



Anaconda Prompt (Anaconda3)

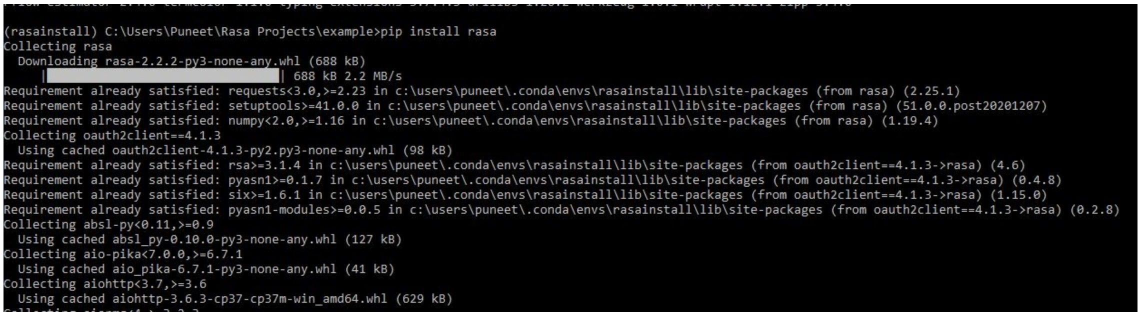
```
(base) C:\Users\Puneet>cd "Rasa Projects"

(base) C:\Users\Puneet\Rasa Projects>cd example

(base) C:\Users\Puneet\Rasa Projects\example>conda create --name rasainstall python==3.7
Collecting package metadata (current_repodata.json): done
Solving environment: failed with repodata from current_repodata.json, will retry with next repodata source.
Collecting package metadata (repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.8.3
  latest version: 4.9.2

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

  environment location: C:\Users\Puneet\.conda\envs\rasainstall

  added / updated specs:
    - python==3.7
```

2. Activate the environment and pip install the ujson and tensorflow package which are required for the smooth working of rasa. Commands for the same are :
    - ***conda activate <name of env>***
    - ***pip install ujson***
    - ***pip install tensorflow***



```
(base) C:\Users\Puneet\Rasa Projects\example>conda activate rasainstall

(rasainstall) C:\Users\Puneet\Rasa Projects\example>pip install ujson
Collecting ujson
  Downloading ujson-4.0.1-cp37-cp37m-win_amd64.whl (43 kB)
     |████████████████████████████████| 43 kB 178 kB/s
Installing collected packages: ujson
Successfully installed ujson-4.0.1

(rasainstall) C:\Users\Puneet\Rasa Projects\example>pip install tensorflow
Collecting tensorflow
  Downloading tensorflow-2.4.0-cp37-cp37m-win_amd64.whl (370.7 MB)
     |██                              | 25.9 MB 3.2 MB/s eta 0:01:48
```

3. The command used is given by ***pip3 install rasa.*** Along with Rasa we also need to install ***vs build tools.***

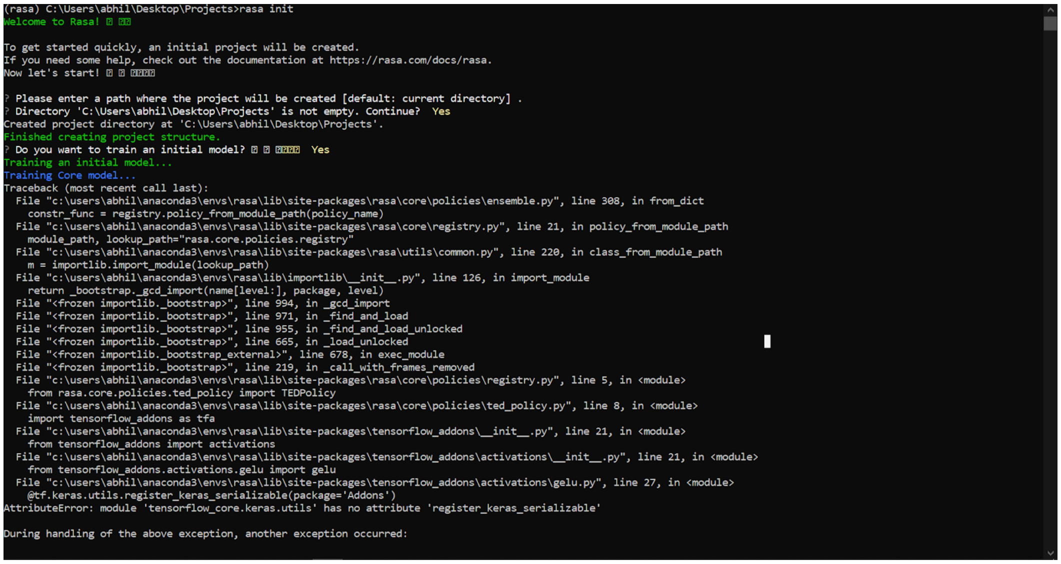


```
(rasainstall) C:\Users\Puneet\Rasa Projects\example>pip install rasa
Collecting rasa
  Downloading rasa-2.2.2-py3-none-any.whl (688 kB)
     |████████████████████████████████| 688 kB 2.2 MB/s
Requirement already satisfied: requests<3.0,>=2.23 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from rasa) (2.25.1)
Requirement already satisfied: setuptools>=41.0.0 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from rasa) (51.0.0.post20201207)
Requirement already satisfied: numpy<2.0,>=1.16 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from rasa) (1.19.4)
Collecting oauth2client==4.1.3
  Using cached oauth2client-4.1.3-py2.py3-none-any.whl (98 kB)
Requirement already satisfied: rsa>=3.1.4 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from oauth2client==4.1.3->rasa) (4.6)
Requirement already satisfied: pyasn1>=0.1.7 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from oauth2client==4.1.3->rasa) (0.4.8)
Requirement already satisfied: six>=1.6.1 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from oauth2client==4.1.3->rasa) (1.15.0)
Requirement already satisfied: pyasn1-modules>=0.0.5 in c:\users\puneet\.conda\envs\rasainstall\lib\site-packages (from oauth2client==4.1.3->rasa) (0.2.8)
Collecting absl-py<0.11,>=0.9
  Using cached absl_py-0.10.0-py3-none-any.whl (127 kB)
Collecting aio-pika<7.0.0,>=6.7.1
  Using cached aio_pika-6.7.1-py3-none-any.whl (41 kB)
Collecting aiohttp<3.7,>=3.6
  Using cached aiohttp-3.6.3-cp37-cp37m-win_amd64.whl (629 kB)
```

Basic Rasa Commands

1. *rasa init*

Creates a new project with example training data, actions, and config files.

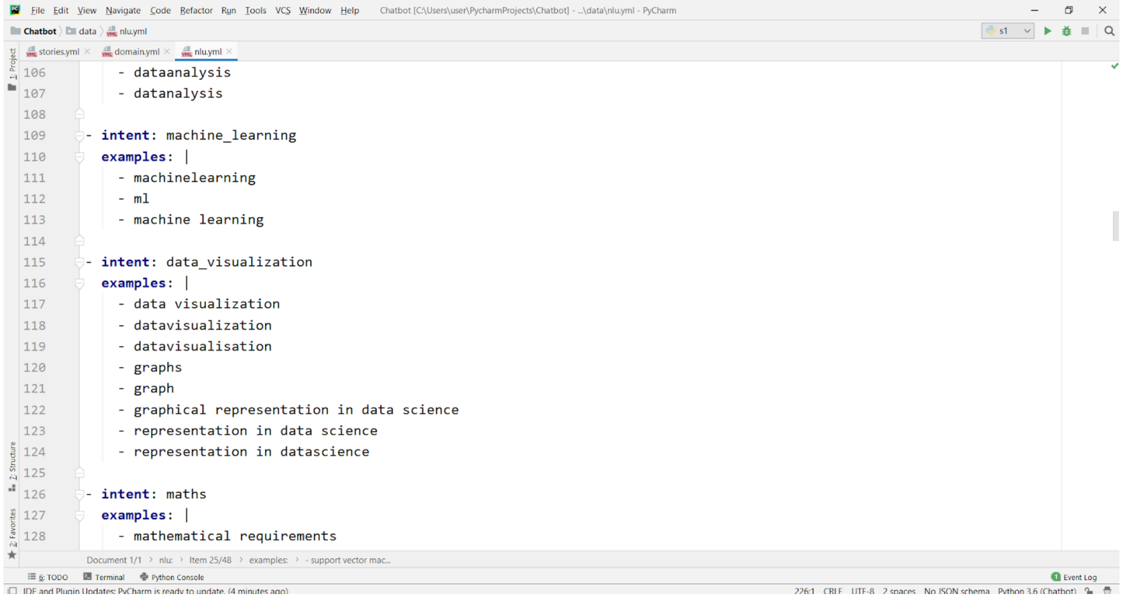


```
(rasa) C:\Users\abhil\Desktop\Projects>rasa init
Welcome to Rasa! ⍰ ⍰⍰

To get started quickly, an initial project will be created.
If you need some help, check out the documentation at https://rasa.com/docs/rasa.
Now let's start! ⍰ ⍰ ⍰⍰⍰⍰

? Please enter a path where the project will be created [default: current directory] .
? Directory 'C:\Users\abhil\Desktop\Projects' is not empty. Continue?  Yes
Created project directory at 'C:\Users\abhil\Desktop\Projects'.
Finished creating project structure.
? Do you want to train an initial model? ⍰ ⍰ ⍰⍰⍰⍰  Yes
Training an initial model...
Training Core model...
Traceback (most recent call last):
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\rasa\core\policies\ensemble.py", line 308, in from_dict
    constr_func = registry.policy_from_module_path(policy_name)
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\rasa\core\registry.py", line 21, in policy_from_module_path
    module_path, lookup_path="rasa.core.policies.registry"
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\rasa\utils\common.py", line 220, in class_from_module_path
    m = importlib.import_module(lookup_path)
  File "c:\users\abhil\anaconda3\envs\rasa\lib\importlib\__init__.py", line 126, in import_module
    return _bootstrap._gcd_import(name[level:], package, level)
  File "<frozen importlib._bootstrap>", line 994, in _gcd_import
  File "<frozen importlib._bootstrap>", line 971, in _find_and_load
  File "<frozen importlib._bootstrap>", line 955, in _find_and_load_unlocked
  File "<frozen importlib._bootstrap>", line 665, in _load_unlocked
  File "<frozen importlib._bootstrap_external>", line 678, in exec_module
  File "<frozen importlib._bootstrap>", line 219, in _call_with_frames_removed
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\rasa\core\policies\registry.py", line 5, in <module>
    from rasa.core.policies.ted_policy import TEDPolicy
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\rasa\core\policies\ted_policy.py", line 8, in <module>
    import tensorflow_addons as tfa
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\tensorflow_addons\__init__.py", line 21, in <module>
    from tensorflow_addons import activations
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\tensorflow_addons\activations\__init__.py", line 21, in <module>
    from tensorflow_addons.activations.gelu import gelu
  File "c:\users\abhil\anaconda3\envs\rasa\lib\site-packages\tensorflow_addons\activations\gelu.py", line 27, in <module>
    @tf.keras.utils.register_keras_serializable(package='Addons')
AttributeError: module 'tensorflow_core.keras.utils' has no attribute 'register_keras_serializable'

During handling of the above exception, another exception occurred:
```

Through rasa init command the following files are created under the project:-

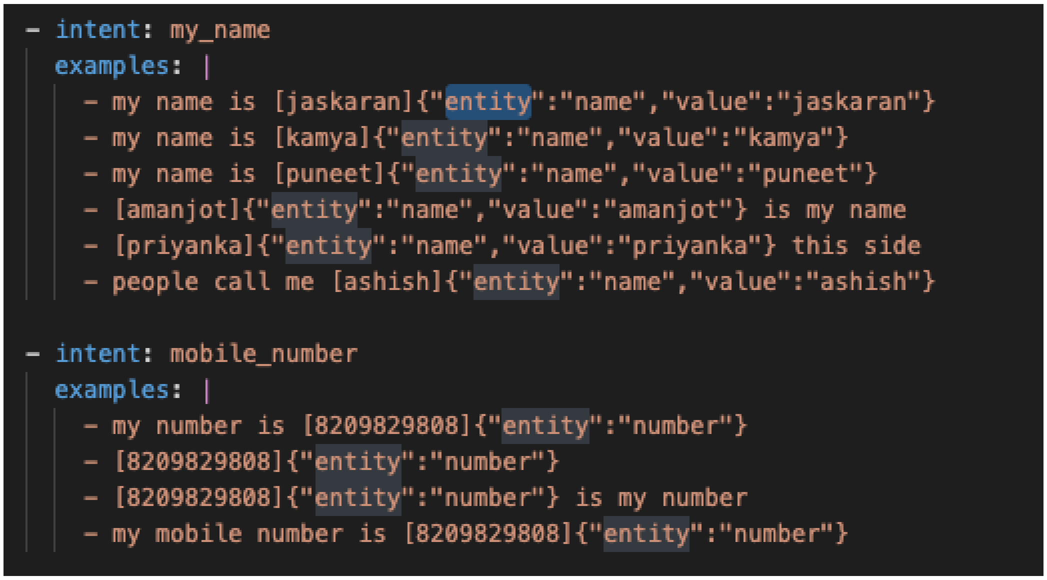
1. **Data(nlu.yml, stories.yml, rules.yml)**
    - **nlu.yml-** The goal of NLU (Natural Language Understanding) is to extract structured information from user messages. This usually includes the user's intent and any entities their message contains.

      NLU training data consists of example user utterances categorized by intent.



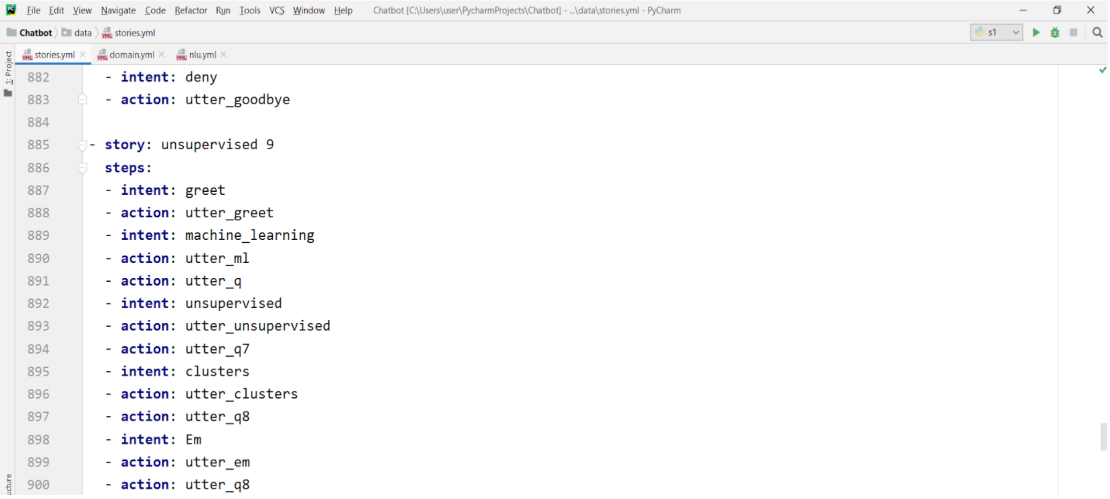
```
    - dataanalysis
    - datanalysis

- intent: machine_learning
  examples: |
    - machinelearning
    - ml
    - machine learning

- intent: data_visualization
  examples: |
    - data visualization
    - datavisualization
    - datavisualisation
    - graphs
    - graph
    - graphical representation in data science
    - representation in data science
    - representation in datascience

- intent: maths
  examples: |
    - mathematical requirements
```

Infact entities can also be added with intent itself in such a way.



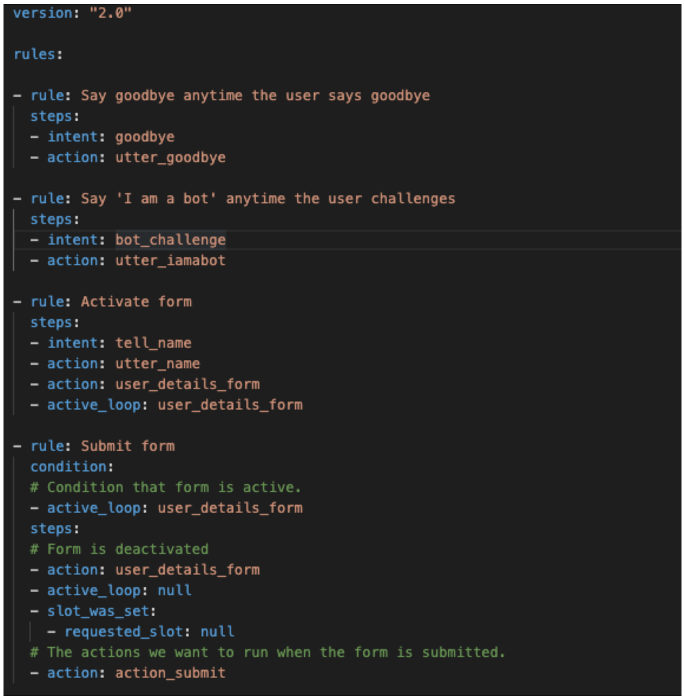
```
- intent: my_name
  examples: |
    - my name is [jaskaran]{"entity":"name","value":"jaskaran"}
    - my name is [kamya]{"entity":"name","value":"kamya"}
    - my name is [puneet]{"entity":"name","value":"puneet"}
    - [amanjot]{"entity":"name","value":"amanjot"} is my name
    - [priyanka]{"entity":"name","value":"priyanka"} this side
    - people call me [ashish]{"entity":"name","value":"ashish"}

- intent: mobile_number
  examples: |
    - my number is [8209829808]{"entity":"number"}
    - [8209829808]{"entity":"number"}
    - [8209829808]{"entity":"number"} is my number
    - my mobile number is [8209829808]{"entity":"number"}
```

- **stories.yml -** Stories are like how you would like the conversation to flow. Different responses you give to the chatbot will have different replies. You are telling the chatbot the various combinations that are possible in the conversation.


File Edit View Navigate Code Refactor Run Tools VCS Window Help
Chatbot [C:\Users\user\PycharmProjects\Chatbot] - ...\data\stories.yml - PyCharm
Chatbot
data
stories.yml
s1
stories.yml
domain.yml
nlu.yml
1: Project

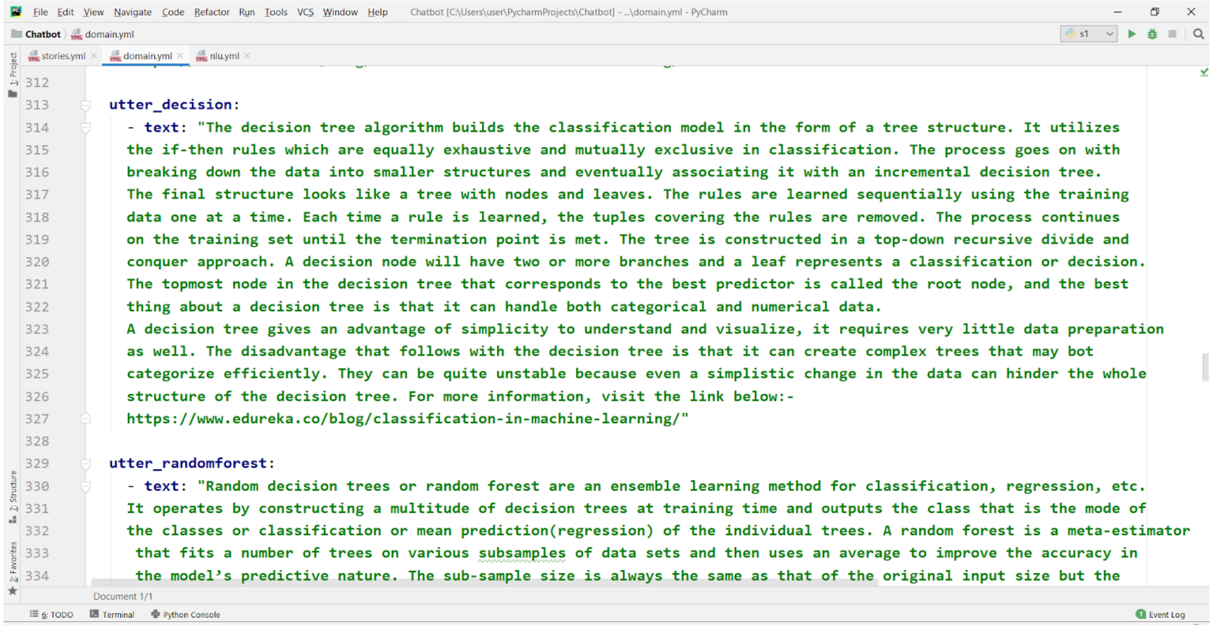
```
  - intent: deny
  - action: utter_goodbye

- story: unsupervised 9
  steps:
  - intent: greet
  - action: utter_greet
  - intent: machine_learning
  - action: utter_ml
  - action: utter_q
  - intent: unsupervised
  - action: utter_unsupervised
  - action: utter_q7
  - intent: clusters
  - action: utter_clusters
  - action: utter_q8
  - intent: Em
  - action: utter_em
  - action: utter_q8
```

- **rules.yml-** Rules are a type of training data used to train your assistant's dialogue management model. Rules describe short pieces of conversations that should always follow the same path.



```
version: "2.0"

rules:

- rule: Say goodbye anytime the user says goodbye
  steps:
  - intent: goodbye
  - action: utter_goodbye

- rule: Say 'I am a bot' anytime the user challenges
  steps:
  - intent: bot_challenge
  - action: utter_iamabot

- rule: Activate form
  steps:
  - intent: tell_name
  - action: utter_name
  - action: user_details_form
  - active_loop: user_details_form

- rule: Submit form
  condition:
  # Condition that form is active.
  - active_loop: user_details_form
  steps:
  # Form is deactivated
  - action: user_details_form
  - active_loop: null
  - slot_was_set:
    - requested_slot: null
  # The actions we want to run when the form is submitted.
  - action: action_submit
```

2. **domain.yml -** The domain defines the universe in which your assistant operates. It specifies the intents, entities, slots, responses, forms, and actions your bot should know about. It also defines a configuration for conversation sessions.

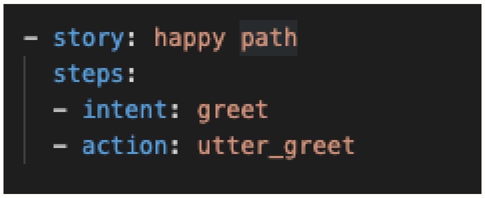

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Chatbot [C:\Users\user\PycharmProjects\Chatbot] - ...\domain.yml - PyCharm
Chatbot
domain.yml
s1
stories.yml
domain.yml
nlu.yml
1: Project
Z: Structure
2: Favorites
Document 1/1
6: TODO
Terminal
Python Console
Event Log

```yaml

  utter_decision:
    - text: "The decision tree algorithm builds the classification model in the form of a tree structure. It utilizes
    the if-then rules which are equally exhaustive and mutually exclusive in classification. The process goes on with
    breaking down the data into smaller structures and eventually associating it with an incremental decision tree.
    The final structure looks like a tree with nodes and leaves. The rules are learned sequentially using the training
    data one at a time. Each time a rule is learned, the tuples covering the rules are removed. The process continues
    on the training set until the termination point is met. The tree is constructed in a top-down recursive divide and
    conquer approach. A decision node will have two or more branches and a leaf represents a classification or decision.
    The topmost node in the decision tree that corresponds to the best predictor is called the root node, and the best
    thing about a decision tree is that it can handle both categorical and numerical data.
    A decision tree gives an advantage of simplicity to understand and visualize, it requires very little data preparation
    as well. The disadvantage that follows with the decision tree is that it can create complex trees that may bot
    categorize efficiently. They can be quite unstable because even a simplistic change in the data can hinder the whole
    structure of the decision tree. For more information, visit the link below:-
    https://www.edureka.co/blog/classification-in-machine-learning/"

  utter_randomforest:
    - text: "Random decision trees or random forest are an ensemble learning method for classification, regression, etc.
    It operates by constructing a multitude of decision trees at training time and outputs the class that is the mode of
    the classes or classification or mean prediction(regression) of the individual trees. A random forest is a meta-estimator
     that fits a number of trees on various subsamples of data sets and then uses an average to improve the accuracy in
     the model’s predictive nature. The sub-sample size is always the same as that of the original input size but the
```

3. Actions (__init__.py, actions.py)

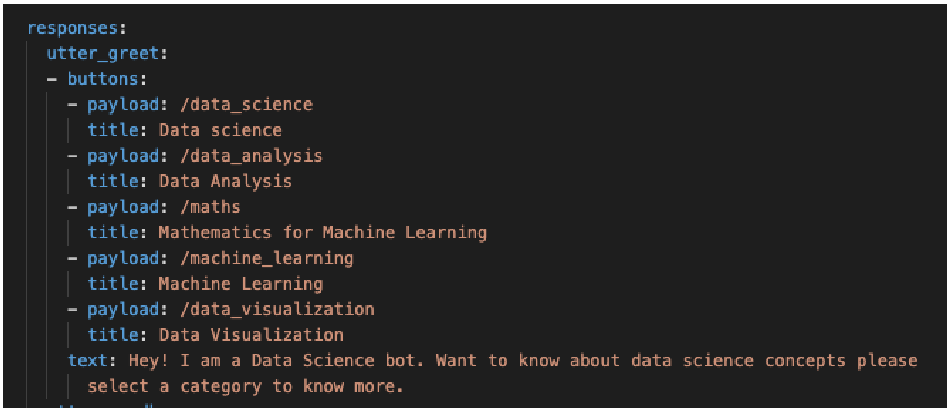
- After each user message, the model will predict an action that the assistant should perform next.
- **Responses** - A response is a message the assistant will send back to the user. This is the action you will use most often, when you want the assistant to send text, images, buttons or similar to the user.

Here we see how we try to give a response in return of intent greet



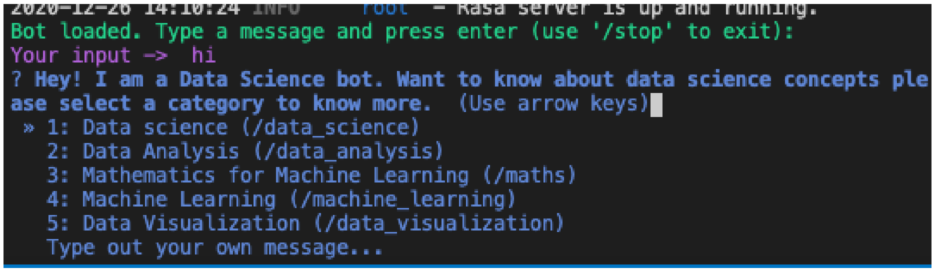
```yaml
- story: happy path
  steps:
  - intent: greet
  - action: utter_greet
```

This is what we wish to respond:-



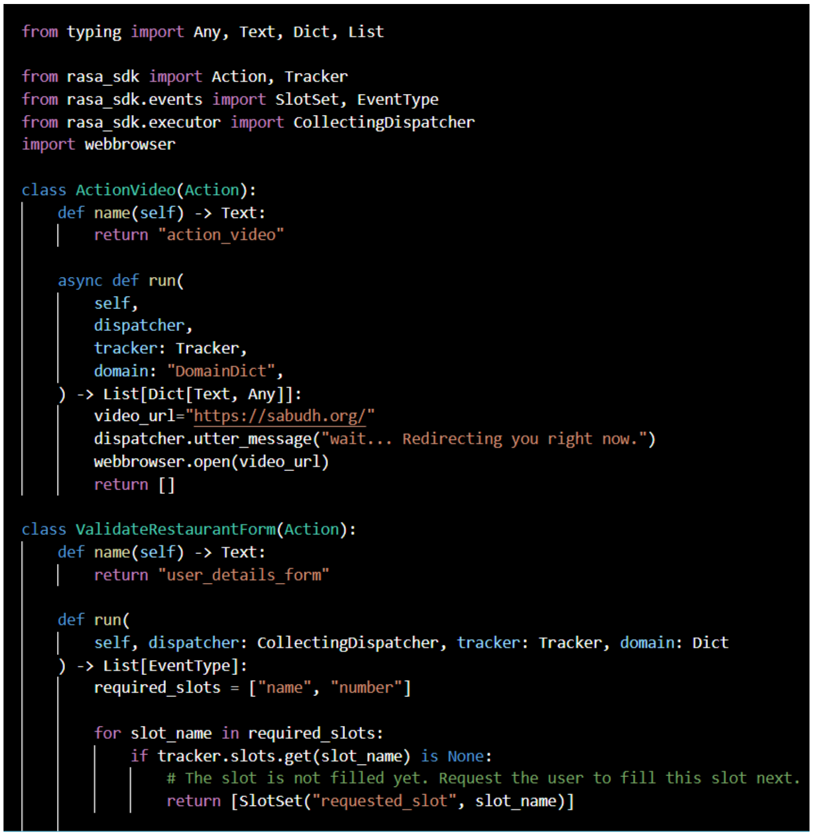
```yaml
responses:
  utter_greet:
  - buttons:
    - payload: /data_science
      title: Data science
    - payload: /data_analysis
      title: Data Analysis
    - payload: /maths
      title: Mathematics for Machine Learning
    - payload: /machine_learning
      title: Machine Learning
    - payload: /data_visualization
      title: Data Visualization
    text: Hey! I am a Data Science bot. Want to know about data science concepts please
      select a category to know more.
```

The final working of a response in play



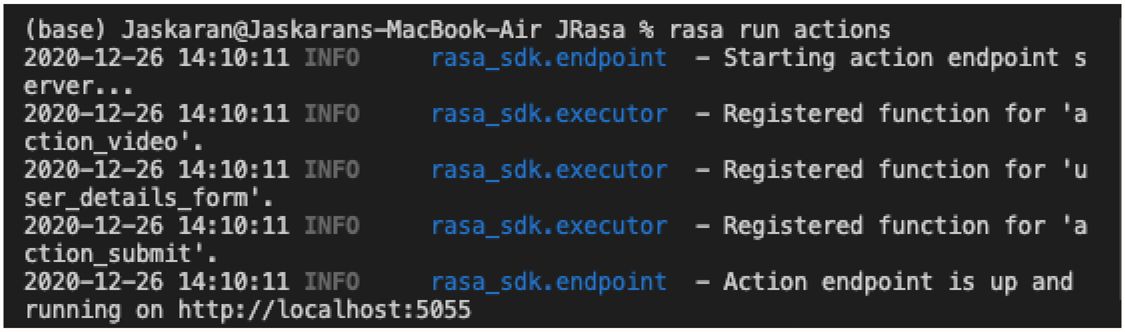
```
2020-12-26 14:10:24 INFO     root  - Rasa server is up and running.
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hi
? Hey! I am a Data Science bot. Want to know about data science concepts ple
ase select a category to know more.  (Use arrow keys)
 » 1: Data science (/data_science)
   2: Data Analysis (/data_analysis)
   3: Mathematics for Machine Learning (/maths)
   4: Machine Learning (/machine_learning)
   5: Data Visualization (/data_visualization)
   Type out your own message...
```

- **Custom Actions** - A custom action is an action that can run any code you want. This can be used to make an API call, or to query a database for example, for custom actions, we first need to write a custom code in action.py file.



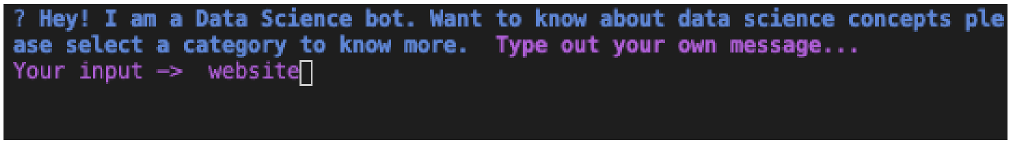
```python
from typing import Any, Text, Dict, List

from rasa_sdk import Action, Tracker
from rasa_sdk.events import SlotSet, EventType
from rasa_sdk.executor import CollectingDispatcher
import webbrowser

class ActionVideo(Action):
    def name(self) -> Text:
        return "action_video"

    async def run(
        self,
        dispatcher,
        tracker: Tracker,
        domain: "DomainDict",
    ) -> List[Dict[Text, Any]]:
        video_url="https://sabudh.org/"
        dispatcher.utter_message("wait... Redirecting you right now.")
        webbrowser.open(video_url)
        return []

class ValidateRestaurantForm(Action):
    def name(self) -> Text:
        return "user_details_form"

    def run(
        self, dispatcher: CollectingDispatcher, tracker: Tracker, domain: Dict
    ) -> List[EventType]:
        required_slots = ["name", "number"]

        for slot_name in required_slots:
            if tracker.slots.get(slot_name) is None:
                # The slot is not filled yet. Request the user to fill this slot next.
                return [SlotSet("requested_slot", slot_name)]
```

And we need to start up our server.



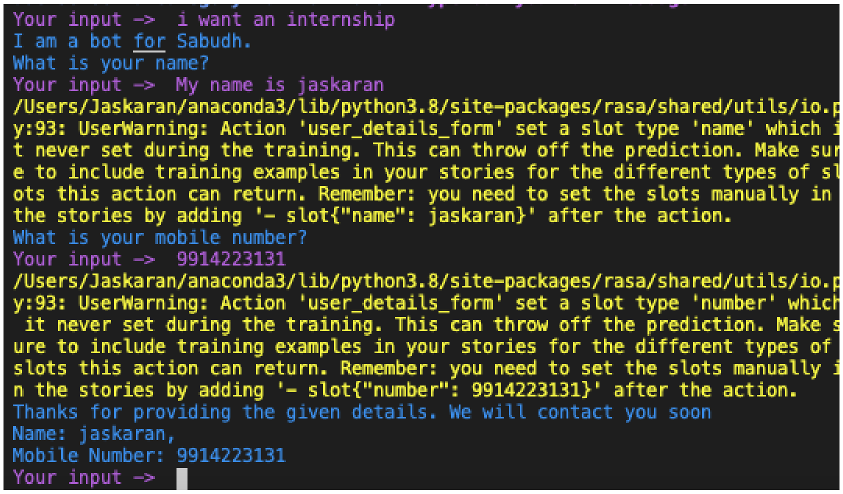
```
(base) Jaskaran@Jaskarans-MacBook-Air JRasa % rasa run actions
2020-12-26 14:10:11 INFO     rasa_sdk.endpoint  - Starting action endpoint s
erver...
2020-12-26 14:10:11 INFO     rasa_sdk.executor  - Registered function for 'a
ction_video'.
2020-12-26 14:10:11 INFO     rasa_sdk.executor  - Registered function for 'u
ser_details_form'.
2020-12-26 14:10:11 INFO     rasa_sdk.executor  - Registered function for 'a
ction_submit'.
2020-12-26 14:10:11 INFO     rasa_sdk.endpoint  - Action endpoint is up and
running on http://localhost:5055
```

We used a custom action to open sabudh.org whenever we feel the intent to be website



```
? Hey! I am a Data Science bot. Want to know about data science concepts ple
ase select a category to know more.  Type out your own message...
Your input ->  website
```

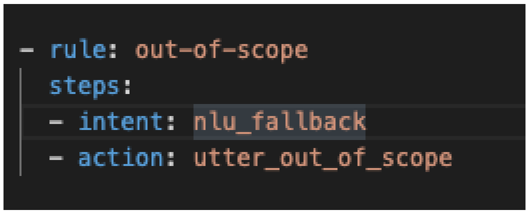
- **Forms** - Forms are a special type of custom action, designed to handle business logic. If you have any conversation designs where you expect the assistant to ask for a specific set of information, you should use forms.



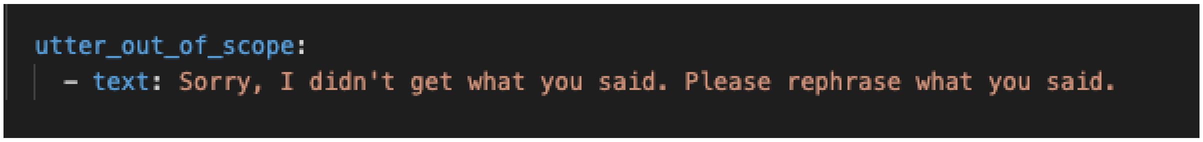
```
Your input ->  i want an internship
I am a bot for Sabudh.
What is your name?
Your input ->  My name is jaskaran
/Users/Jaskaran/anaconda3/lib/python3.8/site-packages/rasa/shared/utils/io.p
y:93: UserWarning: Action 'user_details_form' set a slot type 'name' which i
t never set during the training. This can throw off the prediction. Make sur
e to include training examples in your stories for the different types of sl
ots this action can return. Remember: you need to set the slots manually in
the stories by adding '- slot{"name": jaskaran}' after the action.
What is your mobile number?
Your input ->  9914223131
/Users/Jaskaran/anaconda3/lib/python3.8/site-packages/rasa/shared/utils/io.p
y:93: UserWarning: Action 'user_details_form' set a slot type 'number' which
 it never set during the training. This can throw off the prediction. Make s
ure to include training examples in your stories for the different types of
slots this action can return. Remember: you need to set the slots manually i
n the stories by adding '- slot{"number": 9914223131}' after the action.
Thanks for providing the given details. We will contact you soon
Name: jaskaran,
Mobile Number: 9914223131
Your input ->
```

For our chatbot we made a bot that can collect details of users whenever it feels an intent is reached where user interests for an internship. These can then be saved in a database.

- There is also this concept of a fallback intent, in case the agent did not know how to respond to something.

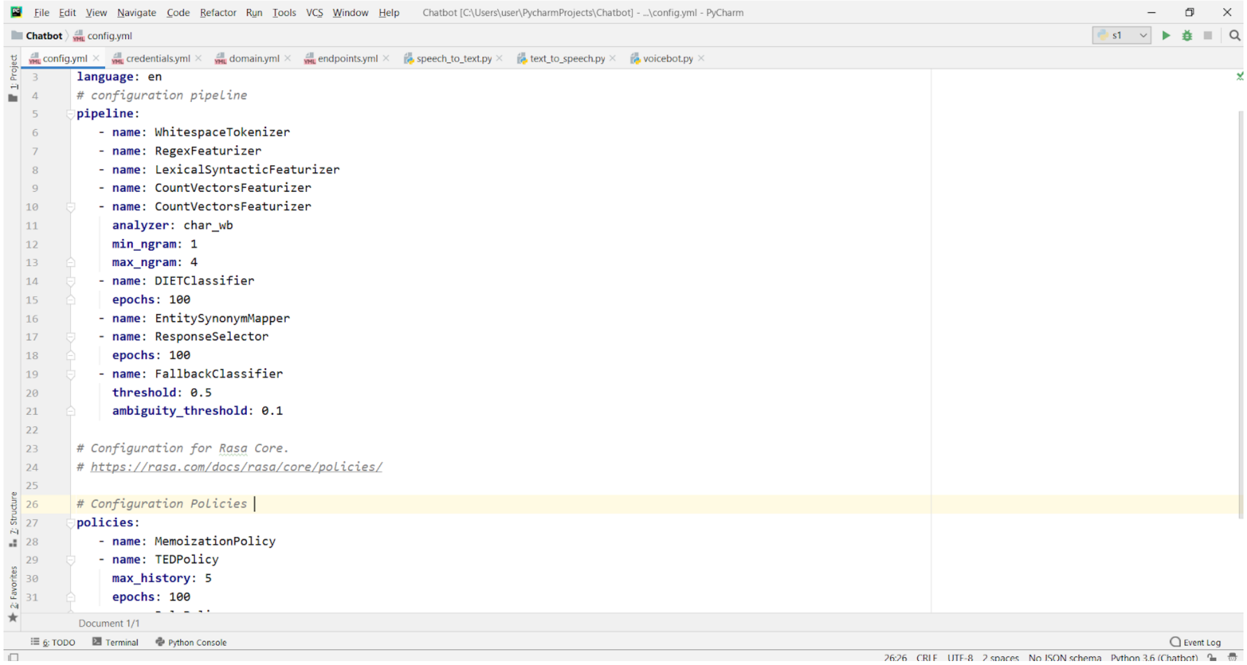


```
- rule: out-of-scope
  steps:
  - intent: nlu_fallback
  - action: utter_out_of_scope
```



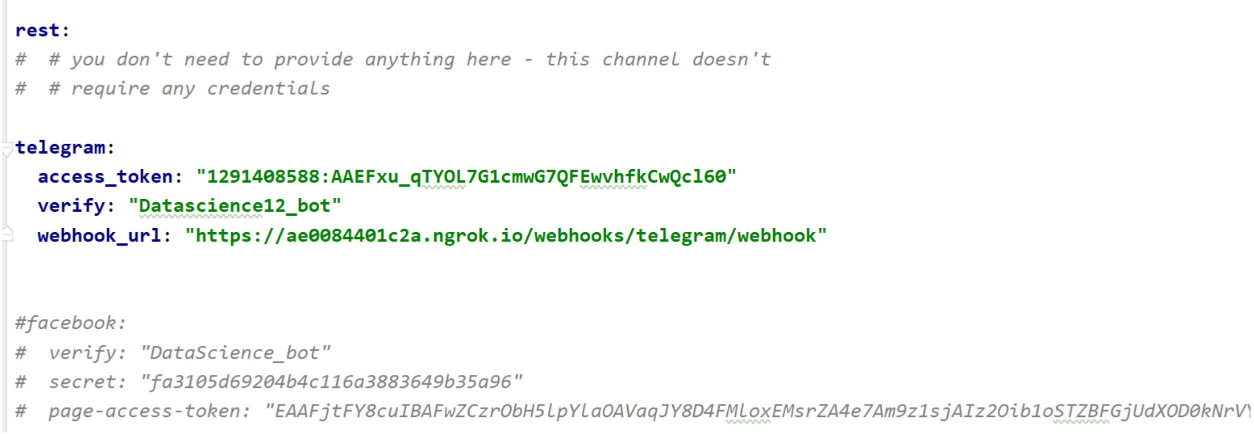
```
utter_out_of_scope:
  - text: Sorry, I didn't get what you said. Please rephrase what you said.
```

4. **config.yml -** The configuration file defines the components and policies that your model will use to make predictions based on user input.



```
language: en
# configuration pipeline
pipeline:
   - name: WhitespaceTokenizer
   - name: RegexFeaturizer
   - name: LexicalSyntacticFeaturizer
   - name: CountVectorsFeaturizer
   - name: CountVectorsFeaturizer
     analyzer: char_wb
     min_ngram: 1
     max_ngram: 4
   - name: DIETClassifier
     epochs: 100
   - name: EntitySynonymMapper
   - name: ResponseSelector
     epochs: 100
   - name: FallbackClassifier
     threshold: 0.5
     ambiguity_threshold: 0.1

# Configuration for Rasa Core.
# https://rasa.com/docs/rasa/core/policies/

# Configuration Policies
policies:
   - name: MemoizationPolicy
   - name: TEDPolicy
     max_history: 5
     epochs: 100
```

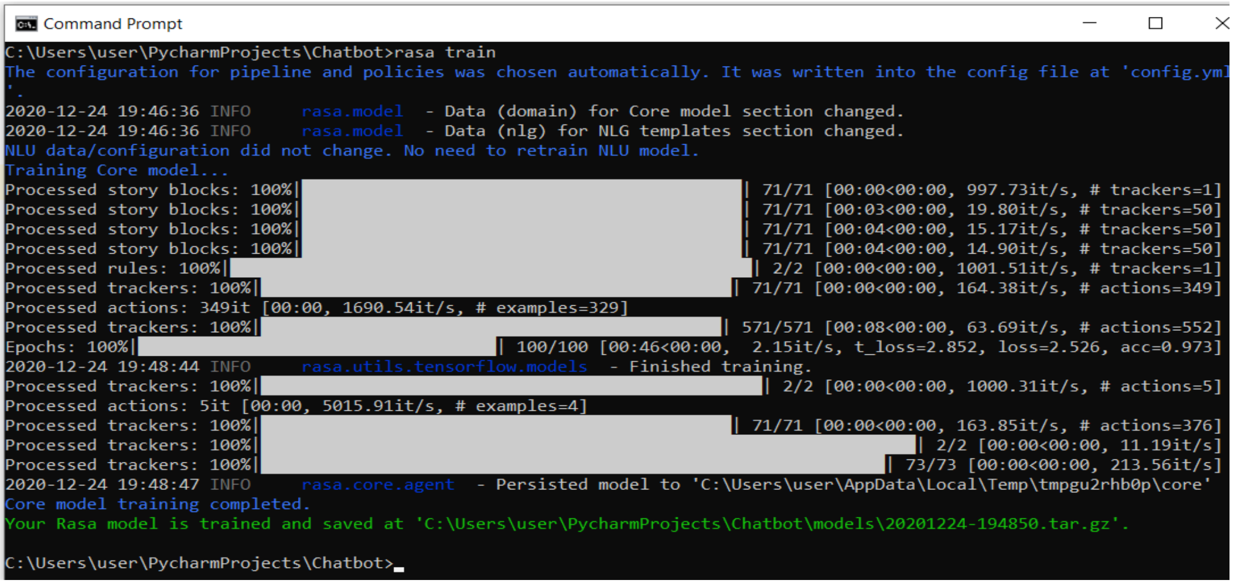
5. **credentials.yml -** This file contains the credentials of the voice and chat platforms we are using.



```
rest:
#  # you don't need to provide anything here - this channel doesn't
#  # require any credentials

telegram:
  access_token: "1291408588:AAEFxu_qTYOL7G1cmwG7QFEwvhfkCwQcl60"
  verify: "Datascience12_bot"
  webhook_url: "https://ae0084401c2a.ngrok.io/webhooks/telegram/webhook"


#facebook:
#  verify: "DataScience_bot"
#  secret: "fa3105d69204b4c116a3883649b35a96"
#  page-access-token: "EAAFjtFY8cuIBAFwZCzrObH5lpYlaOAVaqJY8D4FMLoxEMsrZA4e7Am9z1sjAIz2Oib1oSTZBFGjUdXOD0kNrV
```

6. **endpoints.yml -** This file contains the end points for our bot, the servers from where our bots are pulled.
7. **Tests (test.yml) -** This file contains the stories those are tested to check the accuracy and precision of the stories formed.

2. *rasa train*



```
C:\Users\user\PycharmProjects\Chatbot>rasa train
The configuration for pipeline and policies was chosen automatically. It was written into the config file at 'config.yml
'.
2020-12-24 19:46:36 INFO     rasa.model  - Data (domain) for Core model section changed.
2020-12-24 19:46:36 INFO     rasa.model  - Data (nlg) for NLG templates section changed.
NLU data/configuration did not change. No need to retrain NLU model.
Training Core model...
Processed story blocks: 100%|█████████████████████████████| 71/71 [00:00<00:00, 997.73it/s, # trackers=1]
Processed story blocks: 100%|█████████████████████████████| 71/71 [00:03<00:00, 19.80it/s, # trackers=50]
Processed story blocks: 100%|█████████████████████████████| 71/71 [00:04<00:00, 15.17it/s, # trackers=50]
Processed story blocks: 100%|█████████████████████████████| 71/71 [00:04<00:00, 14.90it/s, # trackers=50]
Processed rules: 100%|███████████████████████████████████| 2/2 [00:00<00:00, 1001.51it/s, # trackers=1]
Processed trackers: 100%|██████████████████████████████| 71/71 [00:00<00:00, 164.38it/s, # actions=349]
Processed actions: 349it [00:00, 1690.54it/s, # examples=329]
Processed trackers: 100%|█████████████████████████████| 571/571 [00:08<00:00, 63.69it/s, # actions=552]
Epochs: 100%|███████████████████████| 100/100 [00:46<00:00,  2.15it/s, t_loss=2.852, loss=2.526, acc=0.973]
2020-12-24 19:48:44 INFO     rasa.utils.tensorflow.models  - Finished training.
Processed trackers: 100%|█████████████████████████████████| 2/2 [00:00<00:00, 1000.31it/s, # actions=5]
Processed actions: 5it [00:00, 5015.91it/s, # examples=4]
Processed trackers: 100%|██████████████████████████████| 71/71 [00:00<00:00, 163.85it/s, # actions=376]
Processed trackers: 100%|█████████████████████████████████████████| 2/2 [00:00<00:00, 11.19it/s]
Processed trackers: 100%|███████████████████████████████████████| 73/73 [00:00<00:00, 213.56it/s]
2020-12-24 19:48:47 INFO     rasa.core.agent  - Persisted model to 'C:\Users\user\AppData\Local\Temp\tmpgu2rhb0p\core'
Core model training completed.
Your Rasa model is trained and saved at 'C:\Users\user\PycharmProjects\Chatbot\models\20201224-194850.tar.gz'.

C:\Users\user\PycharmProjects\Chatbot>_
```

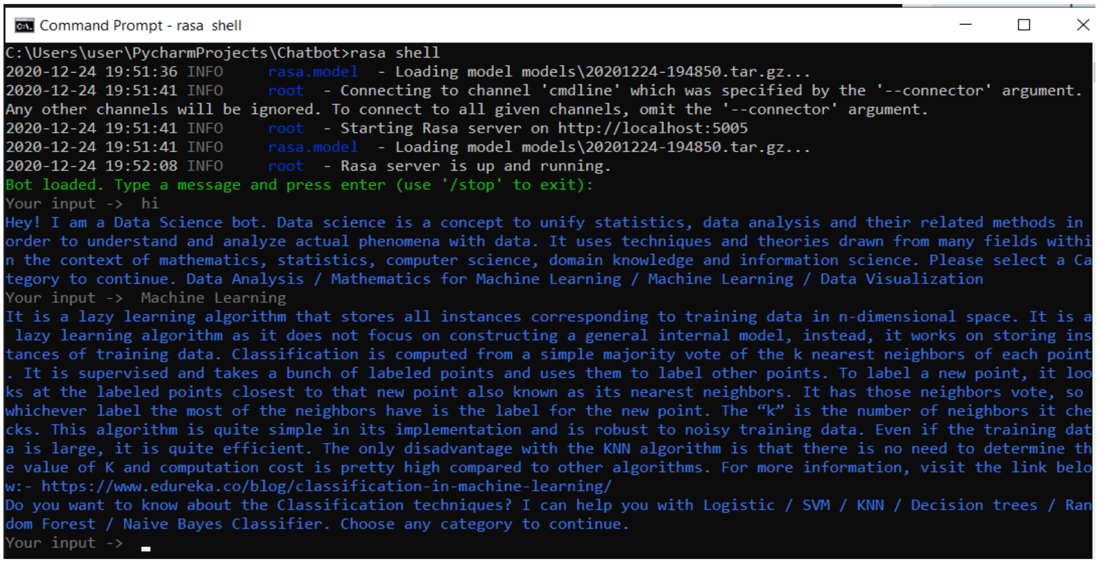
Trains a model using your NLU data and stories and saves trained models in ./models.

If you have existing models in your directory (under models/ by default), only the parts of your model that have changed will be re-trained. For example, if you edit your NLU training data and nothing else, only the NLU part will be trained.

If you want to train an NLU or dialogue model individually, you can run rasa train nlu or rasa train core. If you provide training data only for one one of these, rasa train will fall back to one of these commands by default.

3. *rasa shell*

Loads your trained model and lets you talk to your assistant on the command line.



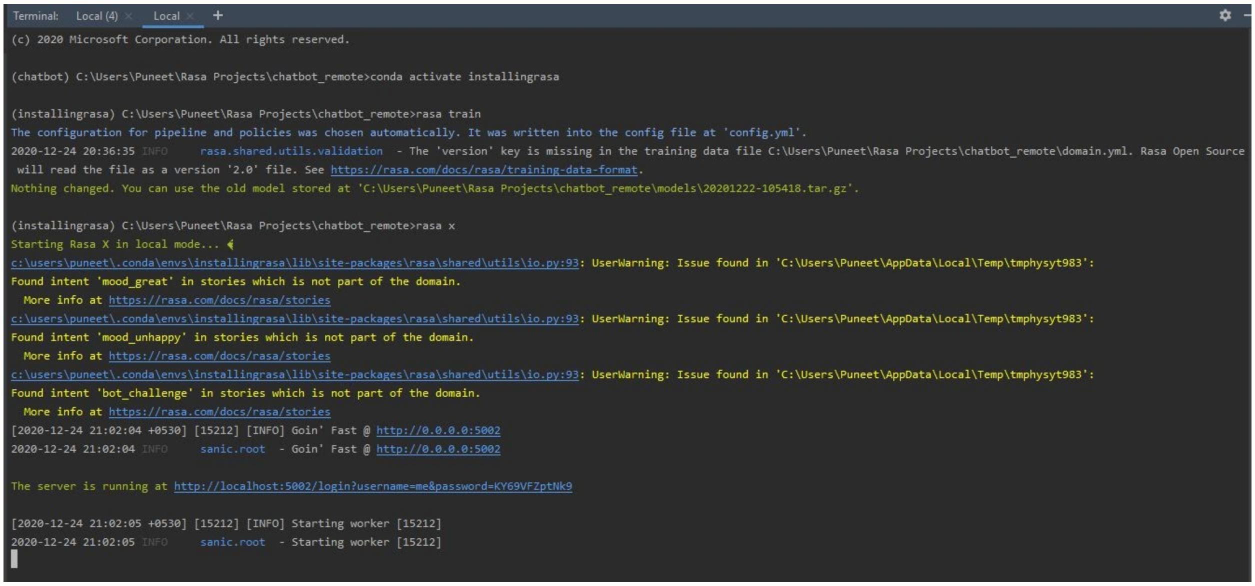
```
C:\Users\user\PycharmProjects\Chatbot>rasa shell
2020-12-24 19:51:36 INFO     rasa.model  - Loading model models\20201224-194850.tar.gz...
2020-12-24 19:51:41 INFO     root  - Connecting to channel 'cmdline' which was specified by the '--connector' argument.
Any other channels will be ignored. To connect to all given channels, omit the '--connector' argument.
2020-12-24 19:51:41 INFO     root  - Starting Rasa server on http://localhost:5005
2020-12-24 19:51:41 INFO     rasa.model  - Loading model models\20201224-194850.tar.gz...
2020-12-24 19:52:08 INFO     root  - Rasa server is up and running.
Bot loaded. Type a message and press enter (use '/stop' to exit):
Your input ->  hi
Hey! I am a Data Science bot. Data science is a concept to unify statistics, data analysis and their related methods in
order to understand and analyze actual phenomena with data. It uses techniques and theories drawn from many fields withi
n the context of mathematics, statistics, computer science, domain knowledge and information science. Please select a Ca
tegory to continue. Data Analysis / Mathematics for Machine Learning / Machine Learning / Data Visualization
Your input ->  Machine Learning
It is a lazy learning algorithm that stores all instances corresponding to training data in n-dimensional space. It is a
 lazy learning algorithm as it does not focus on constructing a general internal model, instead, it works on storing ins
tances of training data. Classification is computed from a simple majority vote of the k nearest neighbors of each point
. It is supervised and takes a bunch of labeled points and uses them to label other points. To label a new point, it loo
ks at the labeled points closest to that new point also known as its nearest neighbors. It has those neighbors vote, so
whichever label the most of the neighbors have is the label for the new point. The “k” is the number of neighbors it che
cks. This algorithm is quite simple in its implementation and is robust to noisy training data. Even if the training dat
a is large, it is quite efficient. The only disadvantage with the KNN algorithm is that there is no need to determine th
e value of K and computation cost is pretty high compared to other algorithms. For more information, visit the link belo
w:- https://www.edureka.co/blog/classification-in-machine-learning/
Do you want to know about the Classification techniques? I can help you with Logistic / SVM / KNN / Decision trees / Ran
dom Forest / Naive Bayes Classifier. Choose any category to continue.
Your input ->  
```

Rasa X or Rasa UI

Rasa has taken a very innovative and unique approach to continuous improvement of chatbots. This approach they have named Conversation-Driven Development (**CDD**). The enabler or tool for CDD is called Rasa X.

Installation of Rasa x

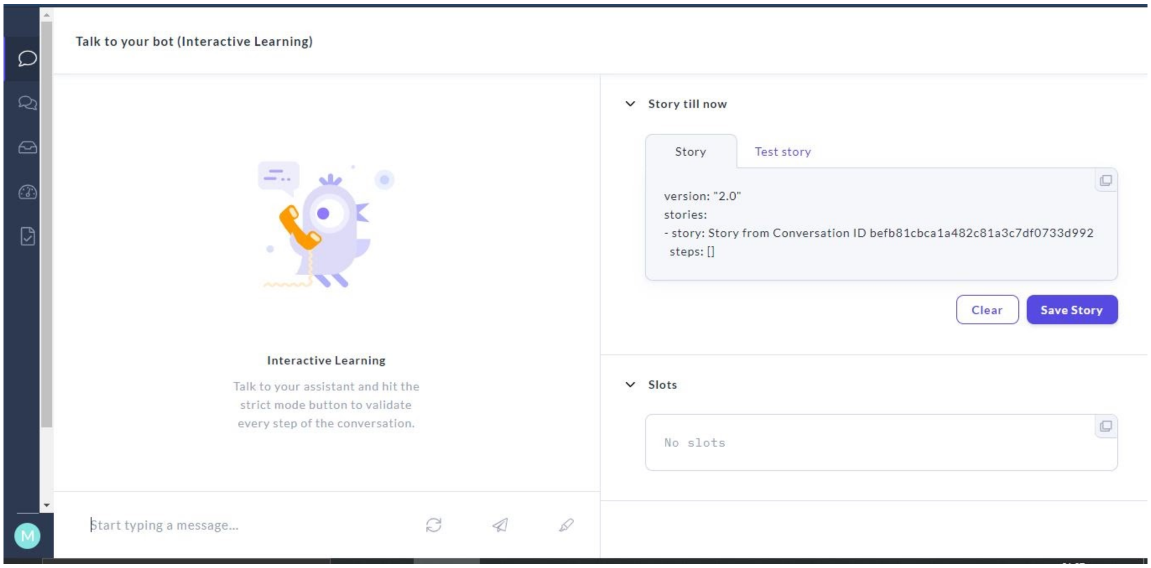
After installation of Rasa for the user interface we can go for rasa x which is Conversation driven development tool for rasa. For rasa x installation we use command ***rasa x*** which is implemented after training your model using ***rasa train*** command.



```
(c) 2020 Microsoft Corporation. All rights reserved.

(chatbot) C:\Users\Puneet\Rasa Projects\chatbot_remote>conda activate installingrasa

(installingrasa) C:\Users\Puneet\Rasa Projects\chatbot_remote>rasa train
The configuration for pipeline and policies was chosen automatically. It was written into the config file at 'config.yml'.
2020-12-24 20:36:35 INFO     rasa.shared.utils.validation  - The 'version' key is missing in the training data file C:\Users\Puneet\Rasa Projects\chatbot_remote\domain.yml. Rasa Open Source
 will read the file as a version '2.0' file. See https://rasa.com/docs/rasa/training-data-format.
Nothing changed. You can use the old model stored at 'C:\Users\Puneet\Rasa Projects\chatbot_remote\models\20201222-105418.tar.gz'.

(installingrasa) C:\Users\Puneet\Rasa Projects\chatbot_remote>rasa x
Starting Rasa X in local mode... 🚀
c:\users\puneet\.conda\envs\installingrasa\lib\site-packages\rasa\shared\utils\io.py:93: UserWarning: Issue found in 'C:\Users\Puneet\AppData\Local\Temp\tmphysyt983':
Found intent 'mood_great' in stories which is not part of the domain.
  More info at https://rasa.com/docs/rasa/stories
c:\users\puneet\.conda\envs\installingrasa\lib\site-packages\rasa\shared\utils\io.py:93: UserWarning: Issue found in 'C:\Users\Puneet\AppData\Local\Temp\tmphysyt983':
Found intent 'mood_unhappy' in stories which is not part of the domain.
  More info at https://rasa.com/docs/rasa/stories
c:\users\puneet\.conda\envs\installingrasa\lib\site-packages\rasa\shared\utils\io.py:93: UserWarning: Issue found in 'C:\Users\Puneet\AppData\Local\Temp\tmphysyt983':
Found intent 'bot_challenge' in stories which is not part of the domain.
  More info at https://rasa.com/docs/rasa/stories
[2020-12-24 21:02:04 +0530] [15212] [INFO] Goin' Fast @ http://0.0.0.0:5002
2020-12-24 21:02:04 INFO     sanic.root  - Goin' Fast @ http://0.0.0.0:5002

The server is running at http://localhost:5002/login?username=me&password=KY69VFZptNk9

[2020-12-24 21:02:05 +0530] [15212] [INFO] Starting worker [15212]
2020-12-24 21:02:05 INFO     sanic.root  - Starting worker [15212]
```

Working of Rasa x

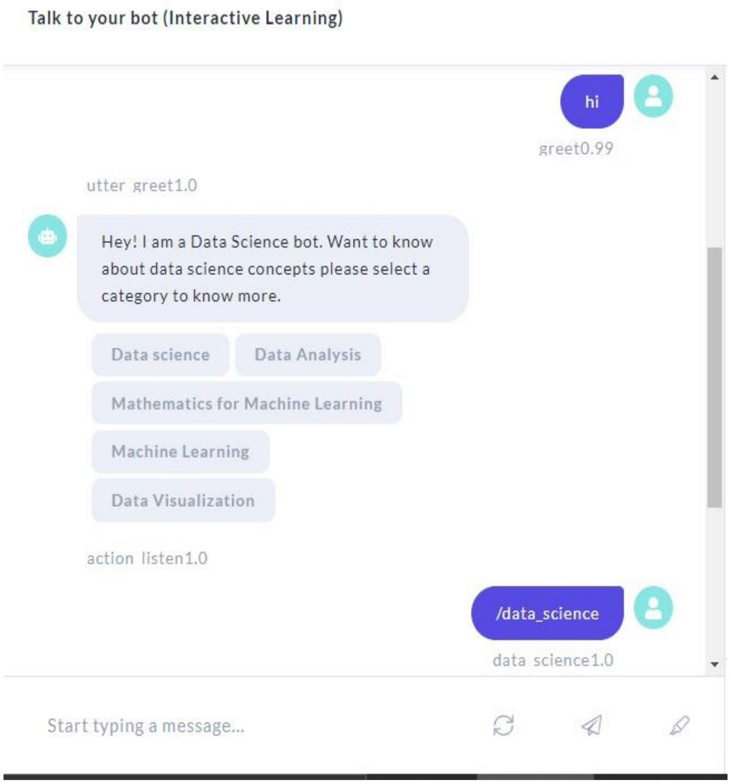
- After running the rasa x command your terminal redirects you to your browser user interface.



Talk to your bot (Interactive Learning)
Story till now
Story
Test story
Clear
Save Story
Interactive Learning
Talk to your assistant and hit the strict mode button to validate every step of the conversation.
Slots
No slots
Start typing a message...
M

```
version: "2.0"
stories:
- story: Story from Conversation ID befb81cbca1a482c81a3c7df0733d992
  steps: []
```

- Here in the side menu bar, there are various functions which help to make your chatbot more interactive.
    1. **Talk to your bot** : This feature provides us an interface as shown above where we can have interactive conversation with our bot after training the model.



Talk to your bot (Interactive Learning)

hi

greet0.99

utter greet1.0

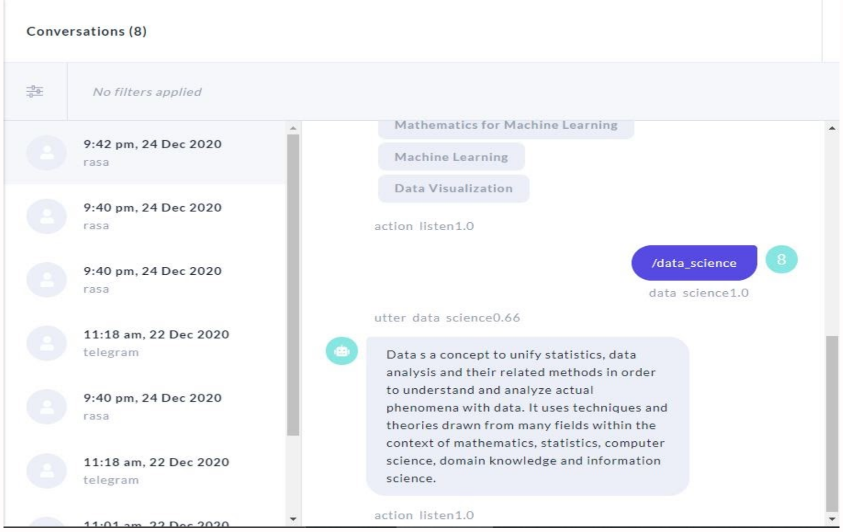
Hey! I am a Data Science bot. Want to know about data science concepts please select a category to know more.

action listen1.0

/data_science

data science1.0

2. **Conversations** : This feature helps us to check our previous conversations with our bot, which are saved at rasa x.



Conversations (8)

9:42 pm, 24 Dec 2020
rasa

9:40 pm, 24 Dec 2020
rasa

9:40 pm, 24 Dec 2020
rasa

11:18 am, 22 Dec 2020
telegram

9:40 pm, 24 Dec 2020
rasa

11:18 am, 22 Dec 2020
telegram

11:01 am, 22 Dec 2020

action listen1.0

/data_science

data science1.0

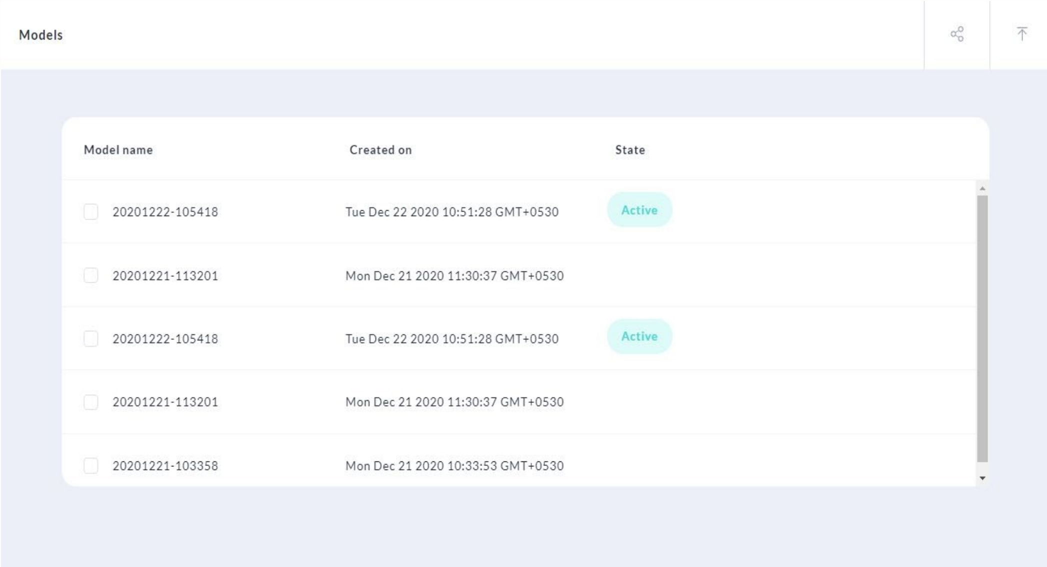
utter data science0.66

Data s a concept to unify statistics, data analysis and their related methods in order to understand and analyze actual phenomena with data. It uses techniques and theories drawn from many fields within the context of mathematics, statistics, computer science, domain knowledge and information science.

action listen1.0

3. **NLU Box** : This box records our intents on the basis of the conversation we had with our chatbot. It records the intents from the sentences which we used to converse with our bot.

4. **Models** : This feature includes all your trained models. You can select the model which you want, and make it active.



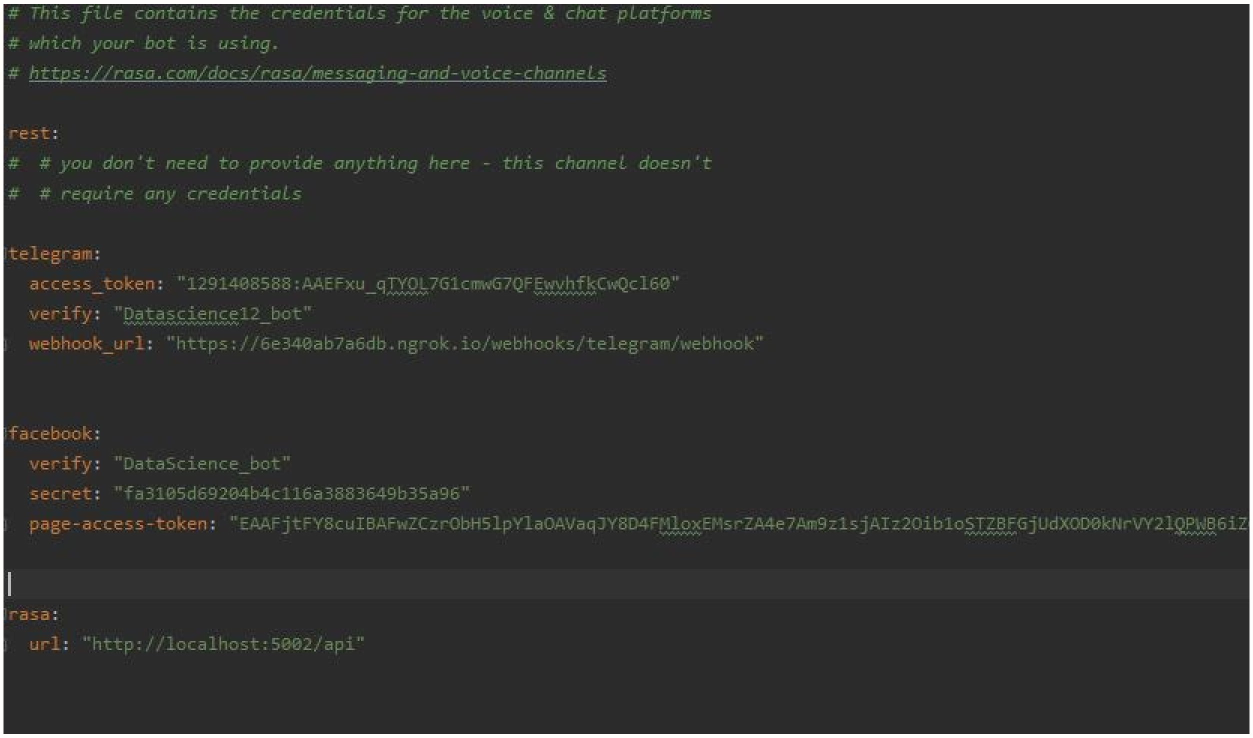
Models

Model name	Created on	State
□ 20201222-105418	Tue Dec 22 2020 10:51:28 GMT+0530	Active
□ 20201221-113201	Mon Dec 21 2020 11:30:37 GMT+0530	
□ 20201222-105418	Tue Dec 22 2020 10:51:28 GMT+0530	Active
□ 20201221-113201	Mon Dec 21 2020 11:30:37 GMT+0530	
□ 20201221-103358	Mon Dec 21 2020 10:33:53 GMT+0530	

5. **Training Data** : This feature has sub features like nlu , stories, domain etc , through which you can add your recent data and can train the model in rasa x only.

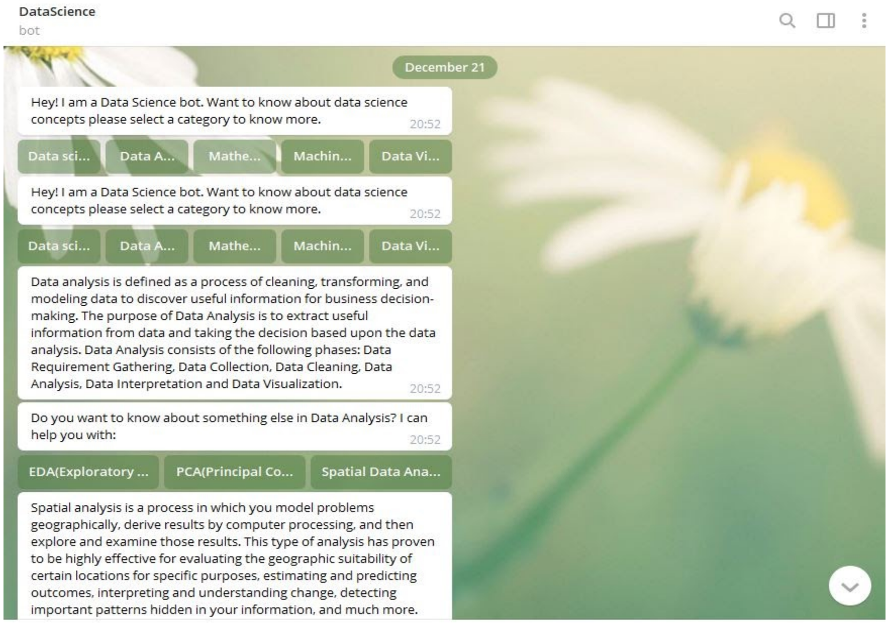
Integrating Telegram with Rasa

- For integrating telegram with rasa, we need to make changes in credentials.yml file, where we need to specify the bot name, access token and the url which directs it to telegram.



```
# This file contains the credentials for the voice & chat platforms
# which your bot is using.
# https://rasa.com/docs/rasa/messaging-and-voice-channels

rest:
#  # you don't need to provide anything here - this channel doesn't
#  # require any credentials

telegram:
  access_token: "1291408588:AAEFxu_qTYOL7G1cmwG7QFEwvhfkCwQcl60"
  verify: "Datascience12_bot"
  webhook_url: "https://6e340ab7a6db.ngrok.io/webhooks/telegram/webhook"


facebook:
  verify: "DataScience_bot"
  secret: "fa3105d69204b4c116a3883649b35a96"
  page-access-token: "EAAFjtFY8cuIBAFwZCzrObH5lpYlaOAVaqJY8D4FMloxEMsrZA4e7Am9z1sjAIz2Oib1oSTZBFGjUdXOD0kNrVY2lQPWB6iZ

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

- After specifying these credentials we need to run the commands below simultaneously:-
    1. ***rasa run***
    2. ***rasa run actions***



Hey! I am a Data Science bot. Want to know about data science concepts please select a category to know more. 20:52

Hey! I am a Data Science bot. Want to know about data science concepts please select a category to know more. 20:52

Data analysis is defined as a process of cleaning, transforming, and modeling data to discover useful information for business decision-making. The purpose of Data Analysis is to extract useful information from data and taking the decision based upon the data analysis. Data Analysis consists of the following phases: Data Requirement Gathering, Data Collection, Data Cleaning, Data Analysis, Data Interpretation and Data Visualization. 20:52

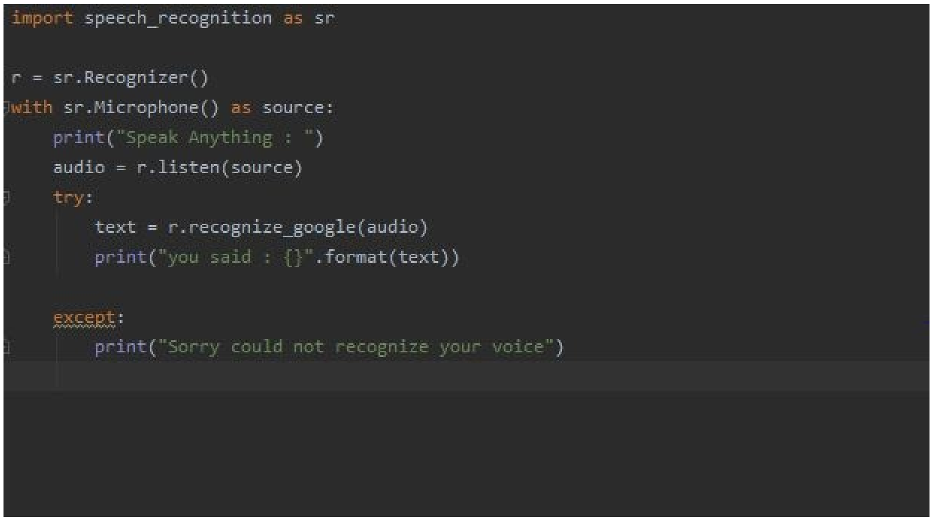
Do you want to know about something else in Data Analysis? I can help you with: 20:52

Spatial analysis is a process in which you model problems geographically, derive results by computer processing, and then explore and examine those results. This type of analysis has proven to be highly effective for evaluating the geographic suitability of certain locations for specific purposes, estimating and predicting outcomes, interpreting and understanding change, detecting important patterns hidden in your information, and much more.

Integrating Voice Assistance

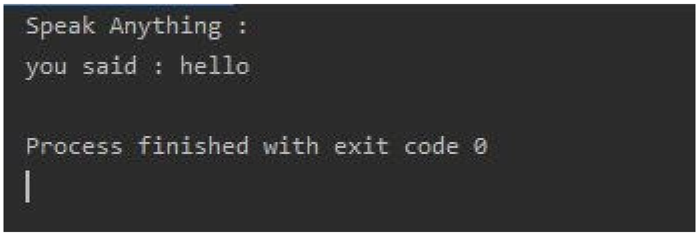
Voice assistance can be used where the user can give input to bot using speech and get a response in return as voice or text. It consists of :-

1. **Speech to Text :** This is an integrated python file which is used to recognize the speech of the user to give a required response.



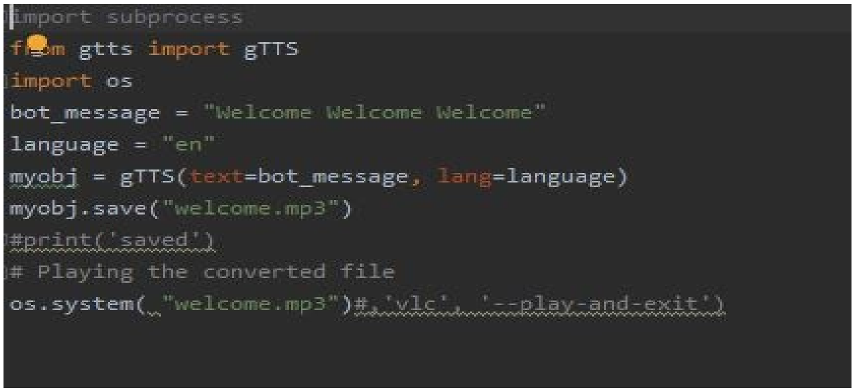
```
import speech_recognition as sr

r = sr.Recognizer()
with sr.Microphone() as source:
    print("Speak Anything : ")
    audio = r.listen(source)
    try:
        text = r.recognize_google(audio)
        print("you said : {}".format(text))

    except:
        print("Sorry could not recognize your voice")
```

This is how it run on command line:



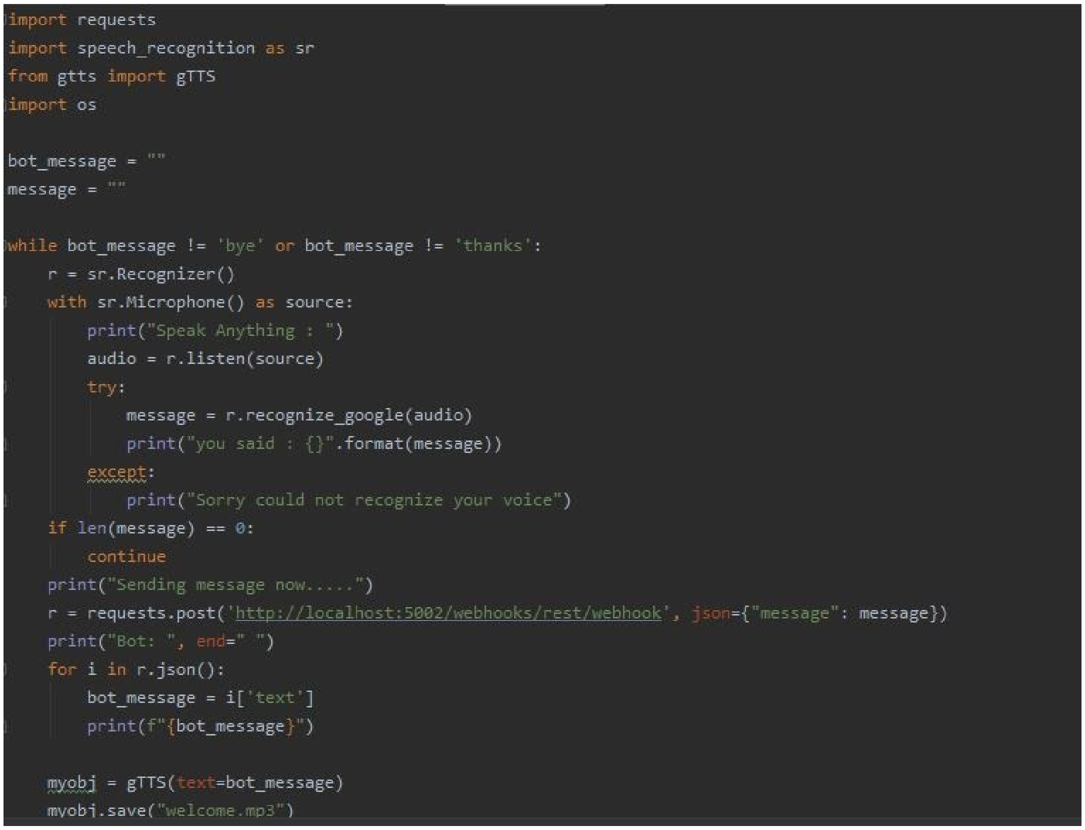
```
Speak Anything :
you said : hello

Process finished with exit code 0
```

2. **Text to speech :** This is an integrated python file which recognizes the words or sentences spoken by the user. For the same we have used the google text to speech library, which saves the text in the mp3 file version.

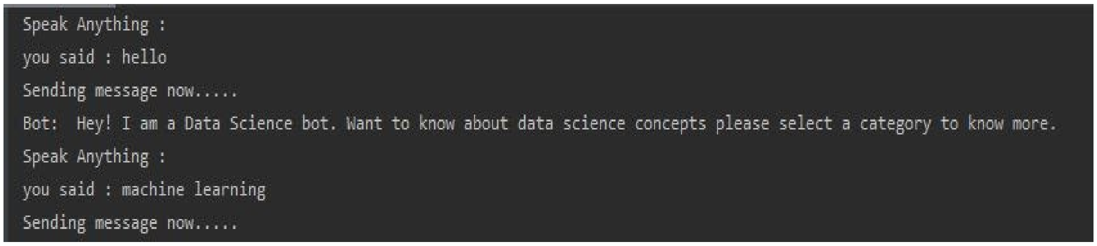


```
import subprocess
from gtts import gTTS
import os
bot_message = "Welcome Welcome Welcome"
language = "en"
myobj = gTTS(text=bot_message, lang=language)
myobj.save("welcome.mp3")
#print('saved')
# Playing the converted file
os.system( "welcome.mp3")#,'vlc', '--play-and-exit')
```

3. **Voicebot :** This python file is the combination of both speech to text and text to speech file. While running this file, we get the required output. To integrate it to the intents and utterances of our bot, we are required to run the following commands simultaneously.
    - ***conda activate <name of env>***
    - ***rasa run -m models --endpoints endpoints.yml --port 5002 --credentials credentials.yml***
    - ***rasa run actions***



```
import requests
import speech_recognition as sr
from gtts import gTTS
import os

bot_message = ""
message = ""

while bot_message != 'bye' or bot_message != 'thanks':
    r = sr.Recognizer()
    with sr.Microphone() as source:
        print("Speak Anything : ")
        audio = r.listen(source)
        try:
            message = r.recognize_google(audio)
            print("you said : {}".format(message))
        except:
            print("Sorry could not recognize your voice")
    if len(message) == 0:
        continue
    print("Sending message now.....")
    r = requests.post('http://localhost:5002/webhooks/rest/webhook', json={"message": message})
    print("Bot: ", end=" ")
    for i in r.json():
        bot_message = i['text']
        print(f"{bot_message}")

    myobj = gTTS(text=bot_message)
    myobj.save("welcome.mp3")
```



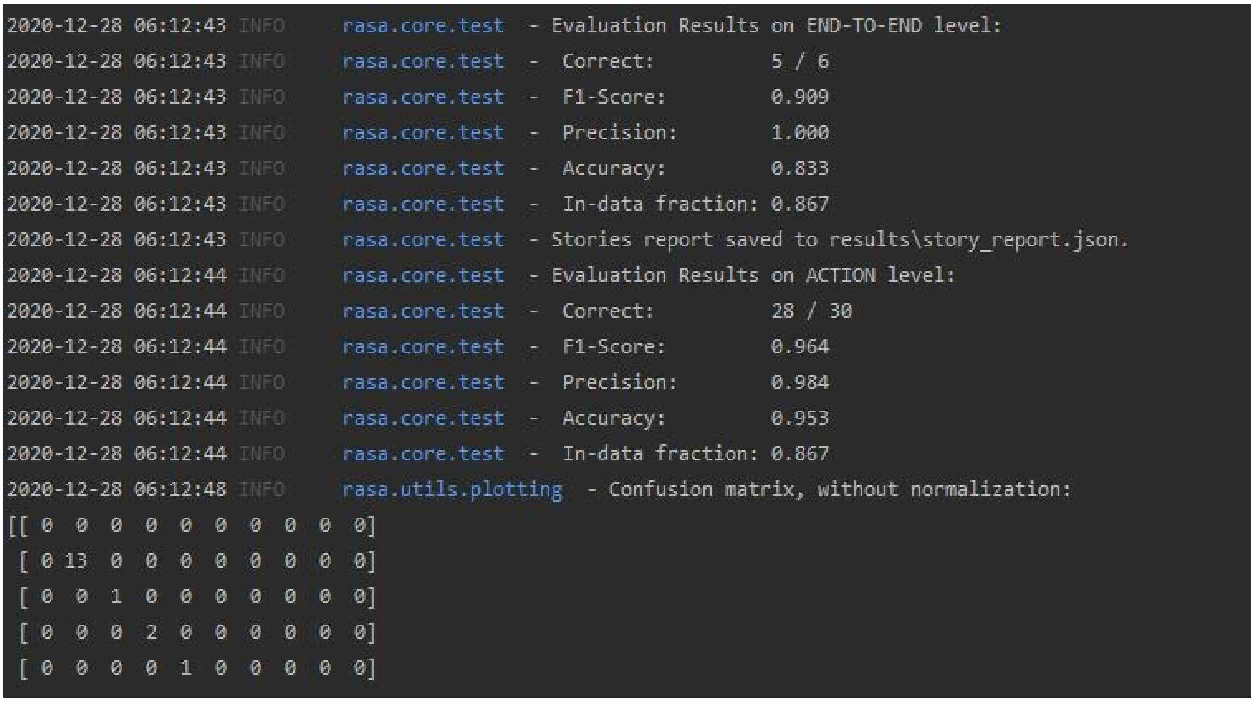
```
Speak Anything :
you said : hello
Sending message now.....
Bot:  Hey! I am a Data Science bot. Want to know about data science concepts please select a category to know more.
Speak Anything :
you said : machine learning
Sending message now.....
```

Accuracy of the Model

Rasa Open Source lets you test dialogues end-to-end by running through test stories. In addition, we can also test the dialogue management and the message processing (NLU) separately.

rasa test command reports recall, precision, and f1-score for each entity type that your trainable entity extractors are trained to recognize.

As seen from the image below, the accuracy of our bot at END to END level is 83.3% with F1-score of 0.909. The accuracy of our bot on ACTION level is 95.3% with F1-score of 0.964.



```
2020-12-28 06:12:43 INFO     rasa.core.test  - Evaluation Results on END-TO-END level:
2020-12-28 06:12:43 INFO     rasa.core.test  -  Correct:          5 / 6
2020-12-28 06:12:43 INFO     rasa.core.test  -  F1-Score:         0.909
2020-12-28 06:12:43 INFO     rasa.core.test  -  Precision:        1.000
2020-12-28 06:12:43 INFO     rasa.core.test  -  Accuracy:         0.833
2020-12-28 06:12:43 INFO     rasa.core.test  -  In-data fraction: 0.867
2020-12-28 06:12:43 INFO     rasa.core.test  - Stories report saved to results\story_report.json.
2020-12-28 06:12:44 INFO     rasa.core.test  - Evaluation Results on ACTION level:
2020-12-28 06:12:44 INFO     rasa.core.test  -  Correct:          28 / 30
2020-12-28 06:12:44 INFO     rasa.core.test  -  F1-Score:         0.964
2020-12-28 06:12:44 INFO     rasa.core.test  -  Precision:        0.984
2020-12-28 06:12:44 INFO     rasa.core.test  -  Accuracy:         0.953
2020-12-28 06:12:44 INFO     rasa.core.test  -  In-data fraction: 0.867
2020-12-28 06:12:48 INFO     rasa.utils.plotting  - Confusion matrix, without normalization:
[[ 0  0  0  0  0  0  0  0  0  0]
 [ 0 13  0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0  0  0]
 [ 0  0  0  2  0  0  0  0  0  0]
 [ 0  0  0  0  1  0  0  0  0  0]
```