Abstract

The digital world redefined the mode and manner of knowledge acquisition, being open to every human being. The internet extends a new world where everything is available; at any time, at anywhere. Thus, the importance of digital literacy is unavoidable as the wide network of google bestows upon us an uncontrollable world, apparently, doubly difficult to filter the most relevant information for our needs. Then enters the chatbot revolution in the Gen Z i.e., a generation of technologically competent adults, but not wanting to wait for long times to resolve their issues. So, MNC's like Amazon, Zomato, Swiggy, etc. have been ahead of the curve in such aspects, but not the Indian Government. When the GST laws were implemented, aimed to usurp corruption right from the roots, naturally the number of people being blindly looted in the name of GST, like truckers, shopkeepers, etc was on the rise. This is the target demographic of our chatbot as we have personally faced this innumerable times. Keeping this in mind, our project aims to develop a new-found chatbot based on Natural language understanding that makes the process of searching easy, timesaving and eliminate the probability of irrelevant information.

Methodology

Introduction:

A chatbot is an application that can initiate and continue a conversation using auditory and/or textual methods as a human would do. A chatbot can be either a simple rule-based engine or an intelligent application leveraging Natural Language Understanding. Many organizations today have started using chatbots extensively. Chatbots are becoming famous as they are available 24*7, provide a consistent customer experience, can handle several customers at a time, are cost-effective and hence, results in a better overall customer experience.

Uses

- Customer support
- Frequently Asked Questions
- Addressing Grievances
- Appointment Booking
- Automation of routine tasks
- Address a query

Prerequisites

The prerequisites for developing and understanding a chatbot using Microsoft Azure are:

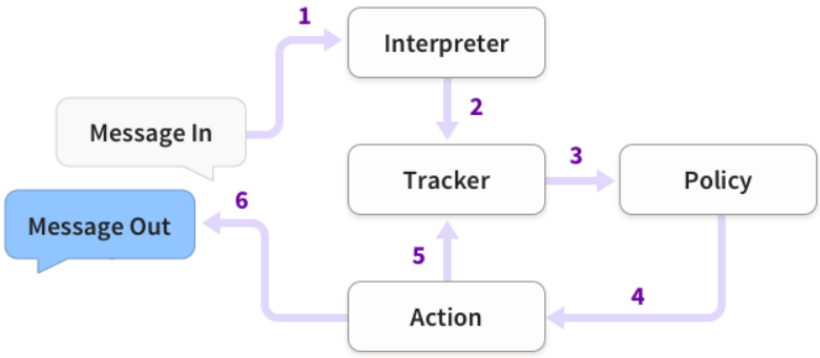
- Python 3.8.0 installed
- ujson

- tensorflow
- spacy
- pandas

Introduction to RASA

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

1.1 RASA Architecture:



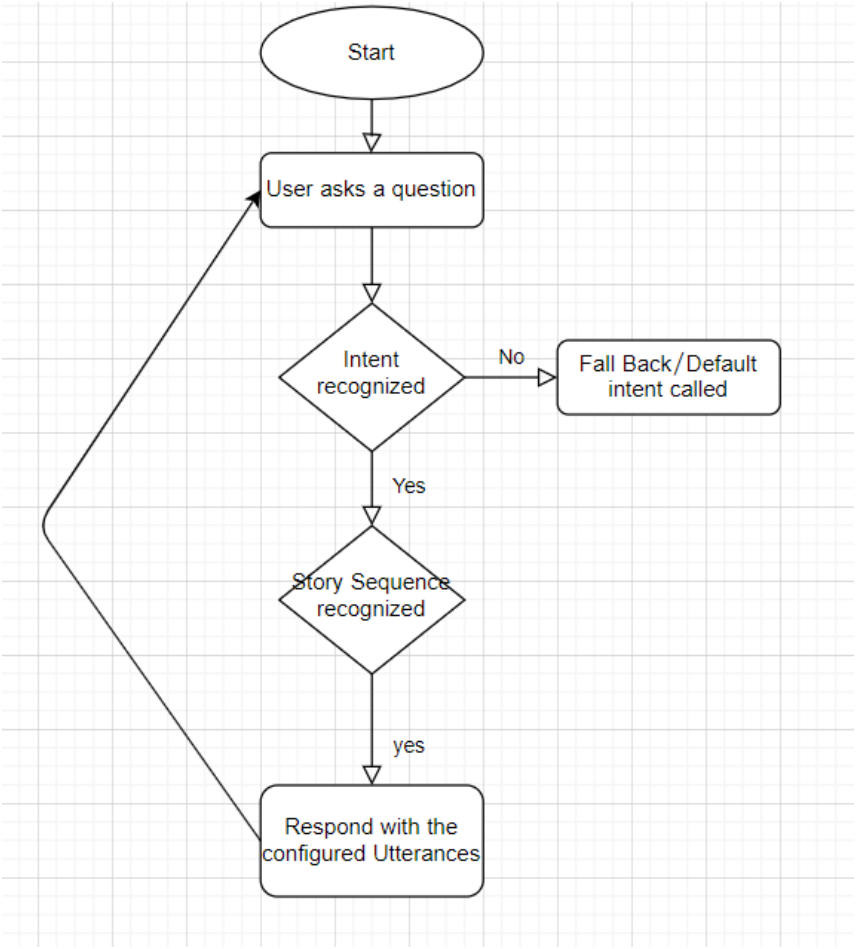
The problem statements

The goal here is to build a chatbot which can answer queries related to the GST.

Technical stack:

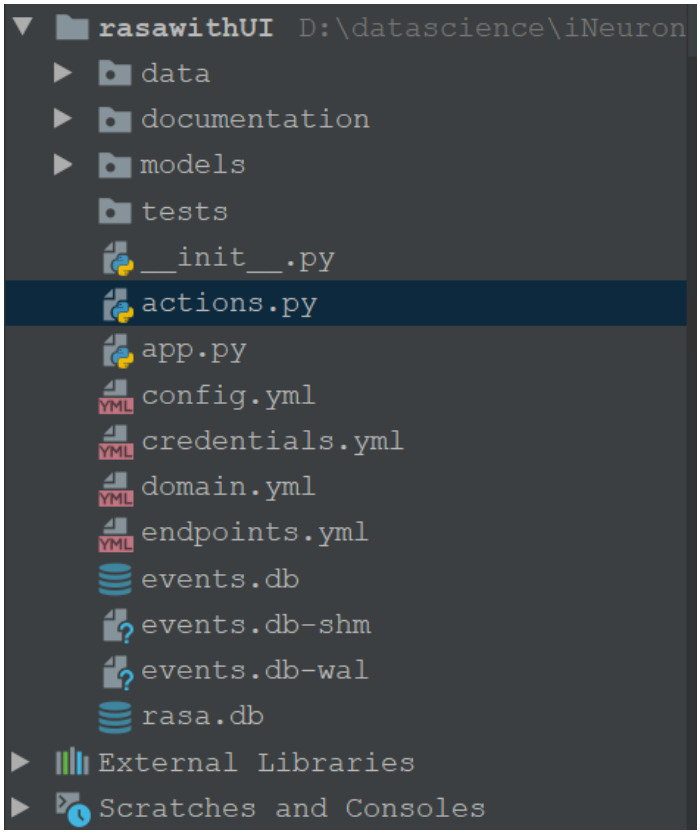
- Python 3.8.0
- ujson
- Tensorflow
- Rasa 2.1.2
- Flask

The application flows



Implementation:

- Install Anaconda
- Create a virtual environment using Anaconda prompt
- Install ujson and tensorflow in Virtual Environment
- Install Rasa 2.1.2 in Virtual Environment
- Enter the command **rasa init** and for all the subsequent actions choose Y (for training the predefined model etc.).
- You'll then end up with all the predefined structures which RASA would have built, as shown below:



- Open the ‘nlu.md’ file from the data folder and enter the following content:

```yaml
- intent: input_tax_credit
  examples: |
    - Can a person without GST registration claim ITC and collect tax?
    - without GST claim ITC
    - without GST collect tax
- intent: eff_date_reg
  examples: |
    - effective date of registration
    - What will be the effective date of registration?
    - date of registration
- intent: liab_for_reg_for_gst
  examples: |
    - Who are the persons liable to take a registration under the GST Law?
    - persons liable to take a registration
    - liable to register
    - persons liable to register
- intent: def_aggregate_turnover
  examples: |
    - What is aggregrate turnover
    - definition of aggregrate turnover
    - define aggregrate turnover
    - what is aggregrate turnover?
    - aggregrate turnover
- intent: reg_compulsion
  examples: |
    - Which are the cases in which registration is compulsory?
    - compulsion registration
    - is registraton compulsory?
    - compulsion of registration
```

This file is used to create all the intents and their sample utterances for conversation.

Open the ‘domain.yml’ file and put the following content:

```yaml
version: '2.0'
session_config:
  session_expiration_time: 60
  carry_over_slots_to_new_session: true
intents:
- greet
- mood_great
- advt_reg_GST
- goodbye
- affirm
- deny
- mood_unhappy
- bot_challenge
- input_tax_credit
- eff_date_reg
- liab_for_reg_for_gst
- def_aggregate_turnover
- reg_compulsion
```

This file is used to configure the bot responses.

Open the ‘stories.md’ file from the data folder and put the following content:

This file is used to create the conversation flows.

```
- story: main
  steps:
  - intent: advt_reg_GST
  - action: utter_advt_reg_GST

- story: main_1
  steps:
  - intent: input_tax_credit
  - action: utter_input_tax_credit

- story: main_2
  steps:
  - intent: eff_date_reg
  - action: utter_eff_date_reg

- story: main_3
  steps:
  - intent: liab_for_reg_for_gst
  - action: utter_liab_for_reg_for_gst

- story: main_3
  steps:
  - intent: def_aggregate_turnover
  - action: utter_def_aggregate_turnover

- story: main_4
  steps:
  - intent: reg_compulsion
  - action: utter_reg_compulsion
```

- We can train the bot with more custom intents and entities
- After all this, you can just enter the command '**rasa train'** to train the model with new conversation elements.

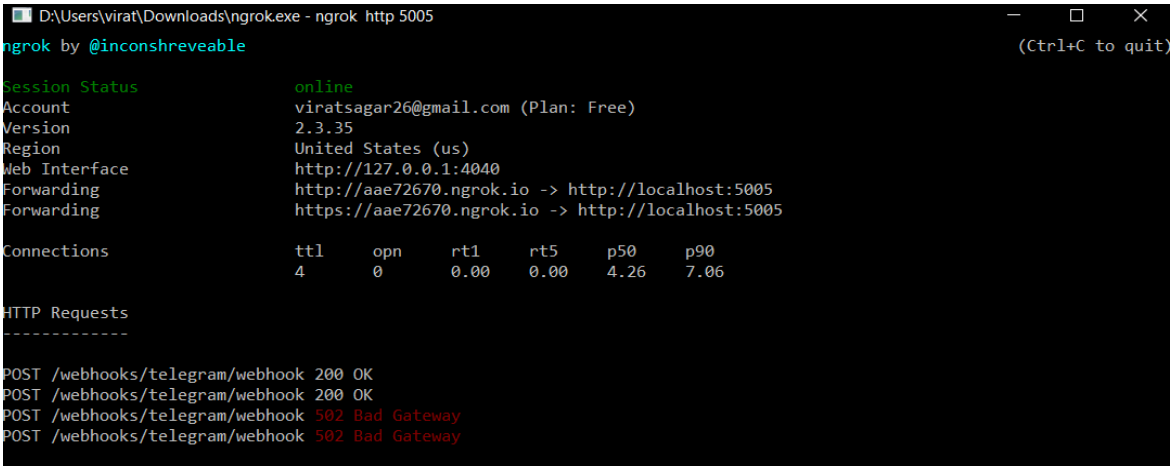
```
The server is running at http://localhost:5002/login?username=me&password=n8DDzwUAo9LL
```

- Copy this URL in your web browser and you'll see the web UI for your chatbot:

Telegram Integration:

- Download ngrok from https://ngrok.com/download
- After extracting the zip file, open the ngrok file and run it.

- In ngrok, enter the command '**ngrok http 5005** ':

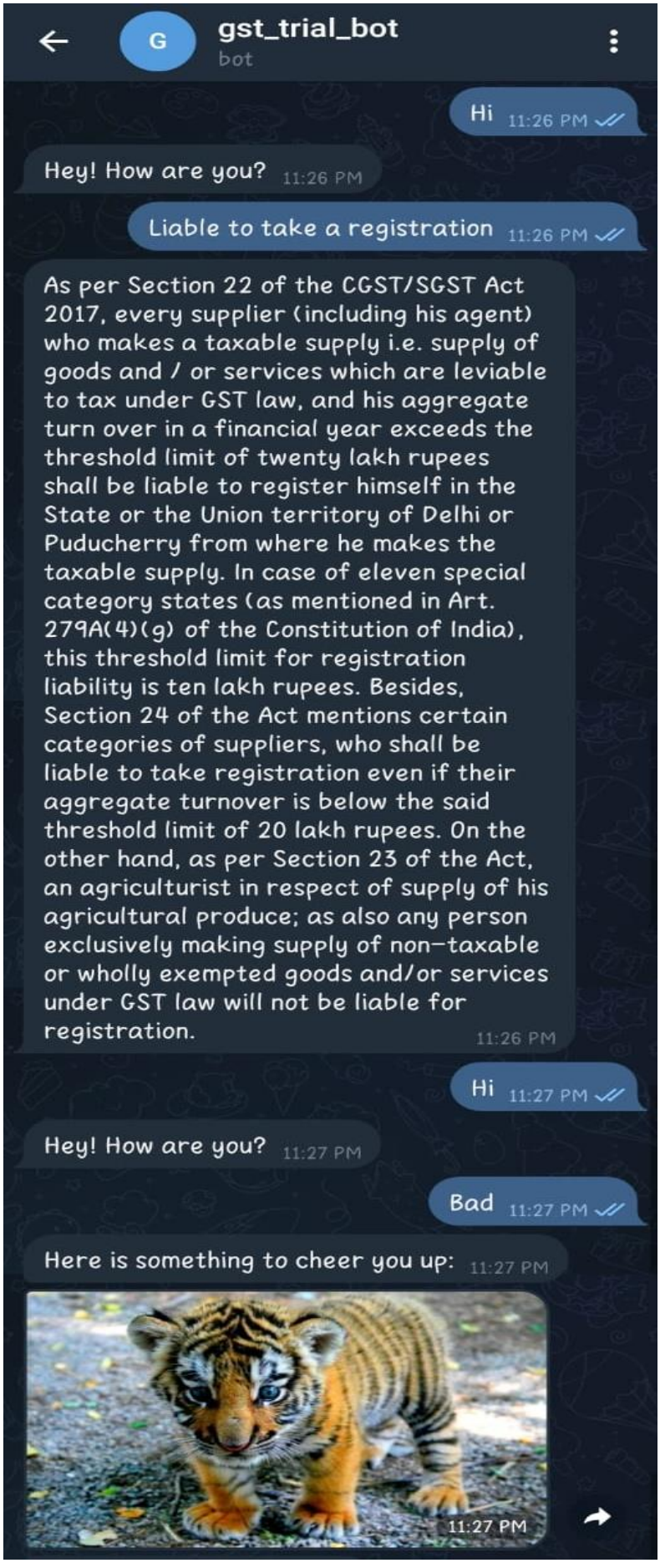


- Then go to telegram and create your own bot using botfather:
  a) Open the telegram app and search for botfather (it is an inbuilt bot used to create other bots)
  b) Start a conversation with botfather and enter /newbot to create a newbot.
  c) Give a name to your bot
  d) Give a username to your bot, which must end in _bot. This generates an access token.

- Open 'credentials.yml' and enter:

```
telegram:
access_token: "obtained from telegram"
verify: "your bot username"
webhook_url: "https://<ngrokurl>/webhooks/telegram/webhook"
```

- Go to terminal and enter the command 'rasa run'
- Open one more terminal and run the command 'rasa run actions'
- Now, you can chat with your bot from Telegram.


G
gst_trial_bot
bot
Hi 11:26 PM
Hey! How are you? 11:26 PM
Liable to take a registration 11:26 PM
As per Section 22 of the CGST/SGST Act 2017, every supplier (including his agent) who makes a taxable supply i.e. supply of goods and / or services which are leviable to tax under GST law, and his aggregate turn over in a financial year exceeds the threshold limit of twenty lakh rupees shall be liable to register himself in the State or the Union territory of Delhi or Puducherry from where he makes the taxable supply. In case of eleven special category states (as mentioned in Art. 279A(4)(g) of the Constitution of India), this threshold limit for registration liability is ten lakh rupees. Besides, Section 24 of the Act mentions certain categories of suppliers, who shall be liable to take registration even if their aggregate turnover is below the said threshold limit of 20 lakh rupees. On the other hand, as per Section 23 of the Act, an agriculturist in respect of supply of his agricultural produce; as also any person exclusively making supply of non-taxable or wholly exempted goods and/or services under GST law will not be liable for registration. 11:26 PM
Hi 11:27 PM
Hey! How are you? 11:27 PM
Bad 11:27 PM
Here is something to cheer you up: 11:27 PM
11:27 PM

GST chatbot UI using Flask API: