GST FAQ Bot with Rasa-NLU

Yogesh H. Kulkarni

Introduction

Command line interface programs have become a thing of the past. We started using GUI (Graphical User Interfaces) programs on Desktops, then similar ones as apps on mobiles. That's also fading away slowly, especially on mobiles. Now is the era of conversational apps, popularly known as chatbots or just bots, such as WhatsApp, Messenger, Slack, etc. not just for friendly chatting but also to get some real information and to get work done. Bots may soon replace websites-interface also.

If the nature of program is more of query-response type (i.e. Question-Answers) then bots are highly suitable. Searching for flight-hotels bookings, knowing admission procedures, enquiring home loan status, etc. are some of the potential applications. To know more about certain procedures, like how to apply for a passport, websites totally provide standard procedure information along with Frequently Asked Questions (FAQs). Going through them can be painstaking and plain boring. How about having a chat interface to FAQs? Such FAQ-bots could be highly effective in Customer Service kind of scenarios. The example shown here is a chat interface to GST (Goods and Service Tax) FAQs.

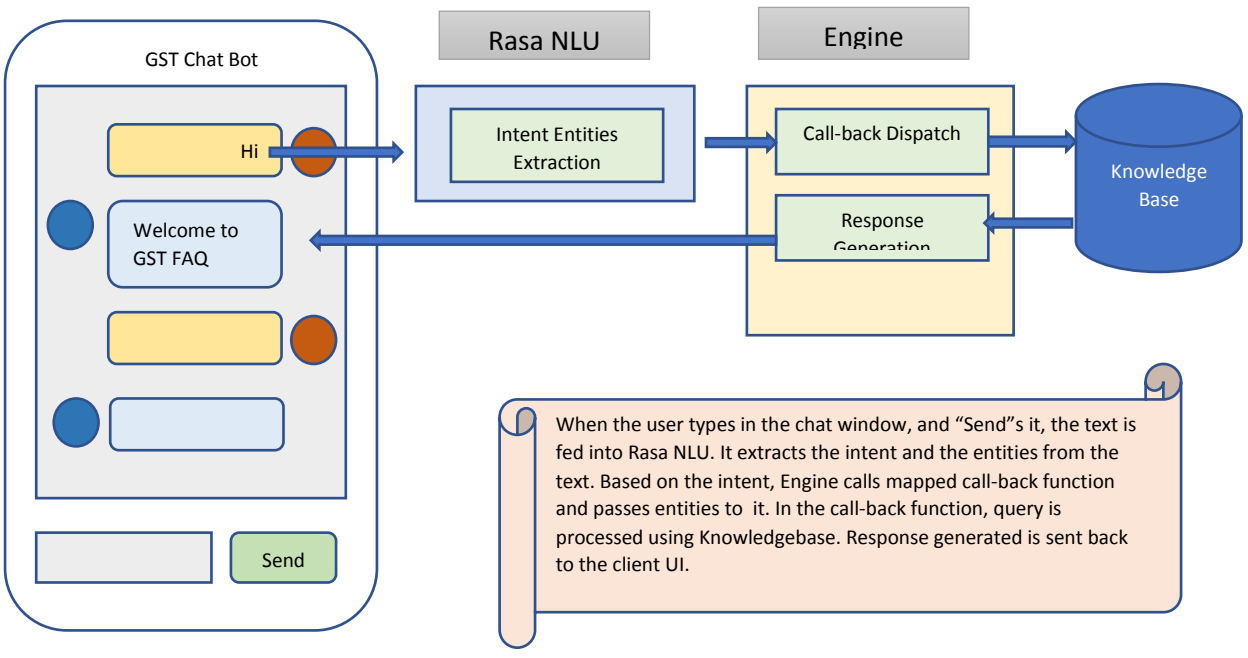
Many chatbot platforms are available, right from rudimentary rule-based (AIML -Artificial Intelligence Markup Language), right up to highly sophisticated Artificially Intelligent bots. Popular platforms are API.ai, Wit.ai, Facebook APIs, Microsoft LUIS, IBM Watson, etc. Most of these are Webservices, where you send conversations to their server, then in response, receive the intent and entities of it. If one cannot send these conversations over the web, due to security issues, then Rasa-NLU could be a good platform choice.

Rasa-NLU builds local NLU (Natural Language Understanding) model for extracting intent and entities from a conversation. It's open source, fully local and above all, free!! It is also compatible with wit.ai, LUIS, or api.ai, so you can migrate your chat application data into Rasa NLU model.

Below is a demonstrative procedure to install Rasa-NLU, and build a simple FAQ bot in Python.

Building GST FAQ bot architecture

Chat bot is a client-server application. In Rasa-NLU case, even server can be local. Client is nothing but Chat Bot UI. Interaction and architecture can be understood by the following diagram:



Installations

Rasa can be installed and configured on a standalone machine. Steps to follow:

1. Ready installation can be done by pip install rasa_nlu
2. If you need the latest, follow instructions at http://rasa-nlu.readthedocs.io/en/latest/installation.html

Rasa-NLU is made up of a few components, each doing some specific work, like one for intent detection, one for entity extraction, etc. Each component may have some specific dependencies and installations. Options like MITIE (NLP+ML), Spacy, Sklearn are available to choose from. Using Spacy-Sklearn here.

Client UI can be a Web page (using frameworks like Flask in Python) or a mobile app. Flask is simple to code and runs locally. Use pip install flask and follow this tutorial for its basic understanding.

Server

Rasa-NLU platforms need to be trained before we start using it. We need to supply it with a few sentences and mention which are the intents and entities in them. Intents are the actions/categories of the sentences and entities are the necessary variables needed to fulfil the actions. For example, "I wish to book flight from Mumbai to Pune on 27 March" has "flight-booking" as the intent and "Mumbai"," Pune", "27 March" as the entities. Similarly, many training examples can be supplied so that Rasa-NLU gets trained on ways of extracting intents/entities from your domain conversations. Such training data is stored in a json, sample of which can be seen at https://github.com/RasaHQ/rasa_nlu/blob/master/data/examples/rasa/demo-rasa.json

It contains many entries. One the sample entry is shown below:

```
    {
      "text": "show me a mexican place in the centre",
      "intent": "restaurant_search",
      "entities": [
        {
          "start": 31,
          "end": 37,
          "value": "centre",
          "entity": "location"
        },
        {
          "start": 10,
          "end": 17,
          "value": "mexican",
          "entity": "cuisine"
        }
      ]
},
```

Meaning of some of the fields shown above, are:

- text: the input sentence. During real running of chat bot, any sentence like the shown would also get treated like it.
- intent: action or category, "restaurant_search". This is typically the call-back function name.

- entities: array of entities. Here there are 2. One is of type ‘location’ with value as ‘centre’, whereas the other is of type ‘cuisine’ and value ‘mexican’. ‘start’, ‘end’ specify beginning and ending indices of the word in the sentence.

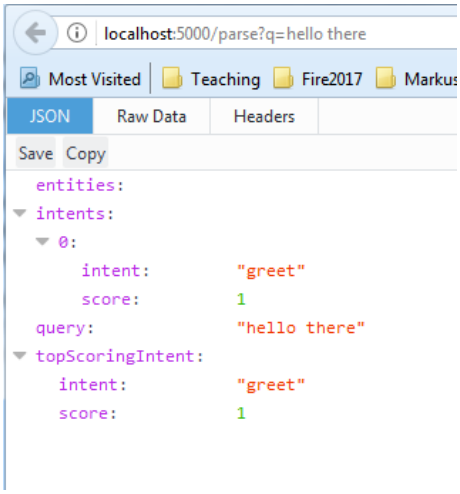
You can use online tool https://rasahq.github.io/rasa-nlu-trainer/ as well, to generate this json file.

Steps to build server side of the GST chat bot application:

1. Create a new directory, navigate into it.
2. Create following things in it
    a. data directory,
    b. data/demo_gst.json: This has GST FAQ training examples as shown before.
    c. config.json: This has settings for Rasa-NLU like shown below:

```
{
 "pipeline": "spacy_sklearn",
 "path" : "./",
 "data" : "./data/gstfaq-data.json"
}
```

3. Train the model by python -m rasa_nlu.train -c config.json
4. This will create a model_YYYYMMDD-HHMMSS folder.
5. Run python -m rasa_nlu.server -c config.json --server_model_dirs=./model_YYYYMMDD-HHMMSS. This starts rasa NLU server and lets us know the port, say 5000.
6. Once the server starts running you can GET/POST via curl to post or use it as HTTP server.
7. For HTTP server, add "-e luis, while running the server and then in the browser say http://localhost:5000/parse?q=hello%20there
8. Output shows as:



We are all set to start with remaining components of the GST FAQ bot.

Client

Chat bot client UI has been built using Flask framework. It uses html templates to render UI (ie Chat window). The template is divided into two parts. Outer UI is built with base.html as shown below:

```
<!DOCTYPE HTML>
<html lang="en">
<head>
…
</head>
  <body>
     <h1 align="center">GST FAQ Chat</h1>
    <div class="container">
      {% block content %}{% endblock %}
    </div>
</body>
  <footer>
    {% block other_footers %}{% endblock %}
  </footer>
</html>
```

The **content** and **other_footers** block are defined in home.html as below:

```
{% block content %}
<div class="row">
  <div class="col-sm-6">
    <div class="row">
      <div class="chat_window">
        <ul class="messages"></ul>
        <div class="bottom_wrapper clearfix">
          <div class="message_input_wrapper">
            <input id="msg_input" class="message_input" placeholder="Say Hi to begin chat..." />
          </div>
          <div class="send_message">
            <div class="text">send</div>
          </div>
        </div>
      </div>
      <div class="message_template">
        <li class="message">
          <div class="avatar"></div>
          <div class="text_wrapper">
            <div class="text"></div>
          </div>
        </li>
      </div>
    </div>
  </div>
</div>
{% endblock %}

{% block other_footers %}
<script src="{{ url_for('static', filename="js/bind.js") }}"></script>
{% endblock %}
```

Engine

This is the heart of the chatbot. Based on the intent received from Rasa-NLU, it dispatches the entities to the mapped call-back functions. The function in turn, depending on the entities, calls Knowledgebase to get the response. Once the response is received, it is sent back to the UI. Knowledgebase can be as simple as a dictionary of questions and answers or as sophisticated as one can imagine/require, like databases, internet sources, etc. This article, being minimalist for demonstration purpose, fetches pre-coded responses from the dictionary.

Sample dictionary can be like:

```
intent_response_dict = {
   "intro": ["This is a GST FAQ bot. One stop-shop to all your GST related queries"],
   "greet":["hey","hello","hi"],
   "goodbye":["bye","It was nice talking to you","see you","ttyl"],
   "affirm":["cool","I know you would like it"],
   "faq_link":['You can check all the events here <a
href="https://www.cbec.gov.in/resources//htdocs-cbec/deptt_offcr/faq-on-gst.pdf</a>']
}
```

Engine's use of Rasa-NLU for intent-entities extraction and dispatching call-backs can be seen below:

```
@app.route('/chat',methods=["POST"])
def chat():
   try:
      user_message = request.form["text"]
      response = requests.get("http://localhost:5000/parse",params={"q":user_message})
      response = response.json()
      response = response["topScoringIntent"]
      intent = response.get("intent")
      entities = response.get("entities")
      if intent == "gst-info":
         response_text = gst_info(entities
      elif intent == "gst-query":
         response_text = gst_query(entities)
      else:
         response_text = get_random_response(intent)
      return jsonify({"status":"success","response":response_text})
   except Exception as e:
      print(e)
      return jsonify({"status":"success","response":"Sorry I am not trained to do that yet..."})
```

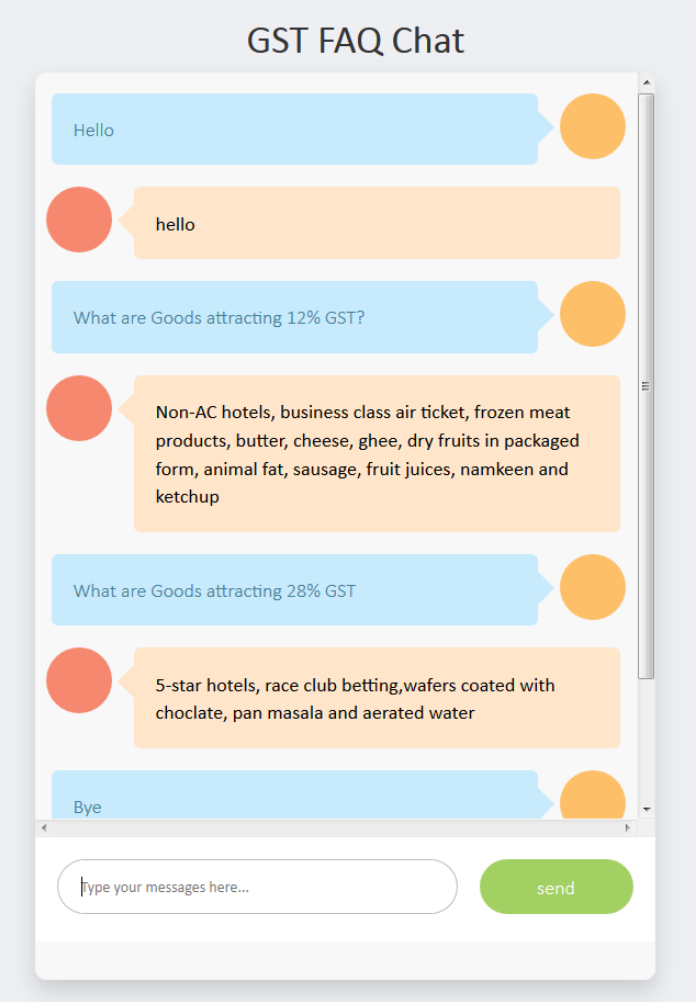
User text is sent to Rasa-NLU server using http://localhost:5000/parse. Its response contains the intent and the entities. Depending on the intent, functions like gst-info and gst-query are called. Their responses are then sent back to UI.

The source code for this app is available at github.

In Operation

Steps to operate:

1. Start the Rasa-NLU server by executing “run_server.bat” script. It loads the custom trained model and starts listening to port 5000
2. Execute the Flash app, by running the localhost at the given port, say 8000.
3. Start typing commands in the bottom chat window and click “Send”.
4. Typed messages and their responses appear in the window above, as seen in the adjoining picture.



Video link for its operations is here.

Future scope

This tutorial presents just a toy example, demonstrating potential to develop something full-fledged and practically useful. GST Q&A bot can be enhanced on various fronts such as expansion of knowledgebase, i.e. number of questions and answers, better training to find more intents and entities, Natural Language Generation of the query responses to have a human language feel, etc.

GST FAQ Bot is just one example of building intuitive frontend over government information. With availability of more APIs or open public data, one can build similar (better) bots for those databases. Imagine you can interact with government departments using a chatty bot!!!

References

- RASA-NLU setup, installation, configuration and sample example http://www.toptechpoint.com/mlai/rasa-nlu-installation-configuration
- Bhavani Ravi’s event-bot code , Youtube video.
- GST FAQs:
    - http://www.cbec.gov.in/resources//htdocs-cbec/deptt_offcr/faq-on-gst.pdf
    - http://www.gstindia.com/frequently-asked-questions-faqs-on-goods-and-services-tax-gst/