

Artificial Intelligence and Robotics

“La Sapienza” University of Rome
Faculty of Information Engineering, Information Technology and Statistics
Department of Informatics, Automation and Control Engineering
"ANTONIO RUBERTI"



SAPIENZA
UNIVERSITÀ DI ROMA

Task-oriented Spoken Dialogue System

NLP Homework

SANNI Oluwatoyin Yetunde [1871835]

Instructor: NARDI, Daniele

07-01-2020

1.0 Introduction

The task-oriented Spoken Dialogue System implemented in this homework is a restaurant waiter robot. The goal of this robot is to serve customers by taking their orders while giving them a list of the available types of menu based on Vegetarian and Non-Vegetarians, as the case may be), and provide them the requested meal in accordance to its availability.

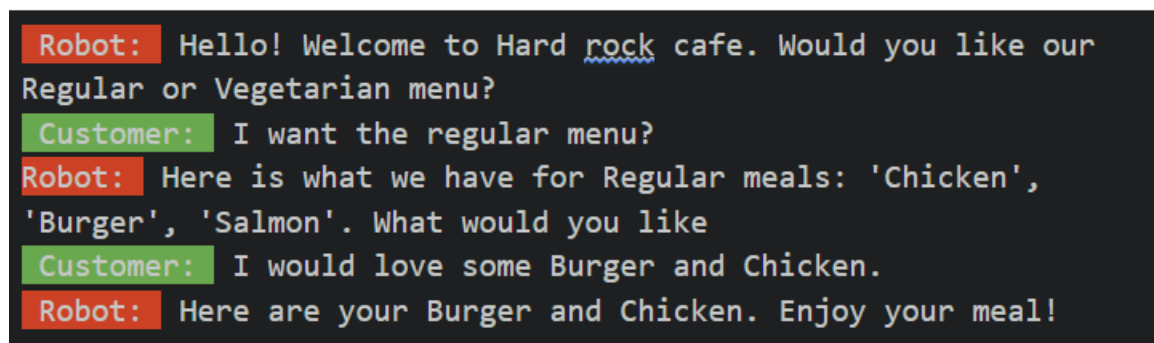
2.0 Implementation

2.1 Scenario

The scenario used is a waiter robot in a restaurant. The event starts with the robot welcoming the customer, then asks if the customer would want a Vegetarian or Regular menu(i.e Non-Vegetarian menu), servesthe order and wishes the customer a wonderful time with their meal.

The only input required by the system is the user's voice, right after showing the word "Speak" . Outputs are also via spoken words by the computer.

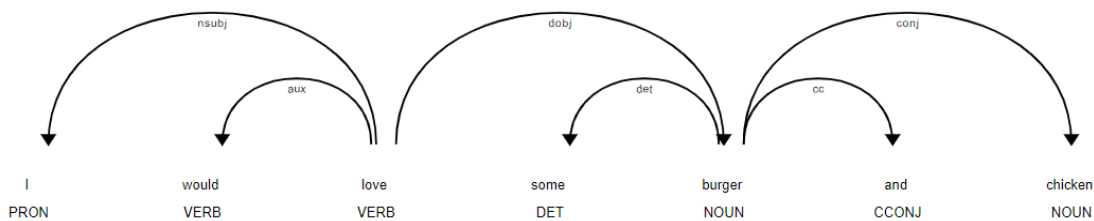
A sample flow conversation is given below:



```
Robot: Hello! Welcome to Hard rock cafe. Would you like our
Regular or Vegetarian menu?
Customer: I want the regular menu?
Robot: Here is what we have for Regular meals: 'Chicken',
'Burger', 'Salmon'. What would you like
Customer: I would love some Burger and Chicken.
Robot: Here are your Burger and Chicken. Enjoy your meal!
```

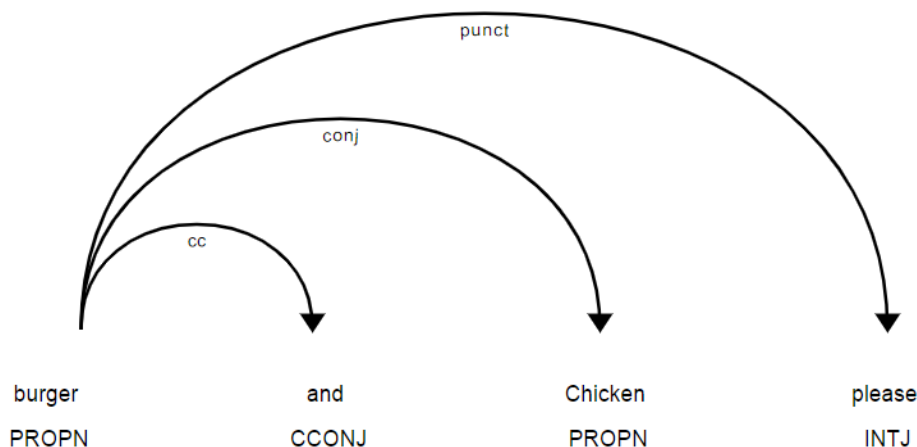
In the simple scenario above, it should be noted that Burger and Chicken are two main distinct meals in the regular meals. The robot sees this event as two orders which is why it uses the plural word "Here are your" instead of "Here is your" . In the event when it's a single order, the latter is used.

The dependency tree is as shown below:



The dependency tree above is the output of Spacy’s dependency parsing for the customer’s input "I would love some Burger and Chicken". From the dependency tree shown above, the parts of speech of each of the words are written below each word while the dependencies are marked as arcs above them; we see “burger” marked as being dependent on “some” and still “burger” being dependent on “chicken” . This is valid since “and” is a conjunction between “burger” , “chicken” .

Dependency parsing focuses on identifying phrases and their recursive structure, resolving ambiguities in the process. Spacy performs a typed dependency passing, which gives labels indicating relationship between words. It starts by tokenizing the input, then proceeds with parsing and tagging which it does using a statistical model. This enables it to make a prediction of which tag or label most likely applies in the context of the whole sentence.



In the dependency tree represented above, we will see that same “Burger” and “Chicken” represented as NOUNS in the previous tree is represented as PROPER NOUNS in another sentence context. This is why our code also checks if the meal that has been mentioned by the customer is either a NOUN or PROPEN.

2.2 Handling exceptions

The exceptions that are handled in this system are generally all about the human supplying an input that cannot be understood by the waiter robot – words that couldn't be correctly parsed or user requests outside the scope of the scenario. Examples include:

- 1) Making an order that doesn't belong to the available menu.
- 2) Requesting for a regular while selecting a meal in the vegetarian list or vice-versa.

We have two major scenarios handled;

- When a meal the customer wants is not in the menu list, Robot responds with; **"I'm sorry but we don't have that <meal>"** .
- In the case of point 2 mentioned above, Robot responds with; **"You have made the wrong order, please try again"** . After this, the Robot would again ask what the Customer would like to have.
- In the case the customer explicitly say they would want nothing, Robot responds with; **"Thanks for stopping by. We hope to see you again soon"** . This exception is caught before and after the menu list has been read out.

2.3 Tools

All parts of this homework was coded in python. The libraries used are highlighted below:

- **Spacy**: This is a python library that provides semantic analysis of text inputs. The text output from SpeechRecognition was supplied as input to this engine. It makes a total analysis of the supplied text from the user input, together with scenario parameters that we coded with it, then provides outputs, based on the scenario parameters. One of the outputs it provides is the dependency tree.
- **gTTS**: A python library and CLI tool to interface with Google Translate's text-to-speech API. Writes spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation. This library was used to convert every text output from Spacy to a voice output for the user.
- **SpeechRecognition**: This is a python library for performing speech recognition, with support for several engines and APIs, online and offline. The API used for this work is the Google Cloud Speech API. It was responsible for accepting speech input, and providing the text equivalent, as best as it can.

3.0 Conclusion

The combination of Spacy, gTTS and SpeechRecognition makes a very powerful tool for Human Robot Interaction projects, allowing for a full speech-based interaction between robots and humans. We have modelled a few possibilities of what could happen during the interaction between the robot and human. A lot more things would happen in a real world scenario, hence, a lot of other considerations and tests would be considered for a production level system.

<https://spacy.io/>

<https://pypi.org/project/gTTS/>

<https://pypi.org/project/playsound/>

<https://pypi.org/project/SpeechRecognition/>