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

GARBA Mariam Olajumoke [1871871]
SANNI Yetunde Oluwatoyin [1871835]
MAKINWA Sayo Michael [1858198]

16.02.2019

1.0 Introduction

The task-oriented Spoken Dialogue System implemented in this homework is a coffee shop waiter robot. The goal of this robot is to serve customers by taking their orders (or give them a list of available variants of coffee, as the case may be), and provide them the requested variant in accordance to its availability.

2.0 Implementation

2.1 Scenario

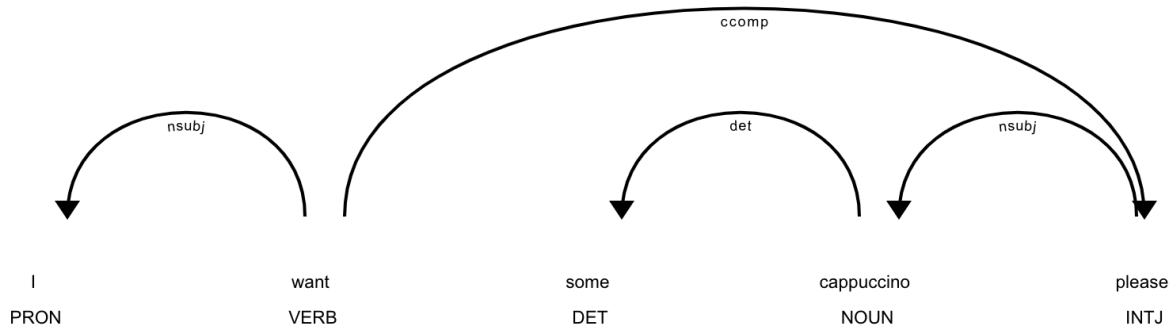
The scenario used is a waiter robot in a coffee shop. The event starts with the robot welcoming the customer, then asking what type of coffee the customer wants, taking the order, serving the order and wishing the customer a great day.

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 our coffee shop. What kind of coffee do you want?
Customer: What do you have?
Robot: We have Cappuccino, Espresso, Vienna
Customer: I want some Cappuccino, please.
Robot: Here's your Cappucino. Enjoy!

The dependency tree is as shown below:

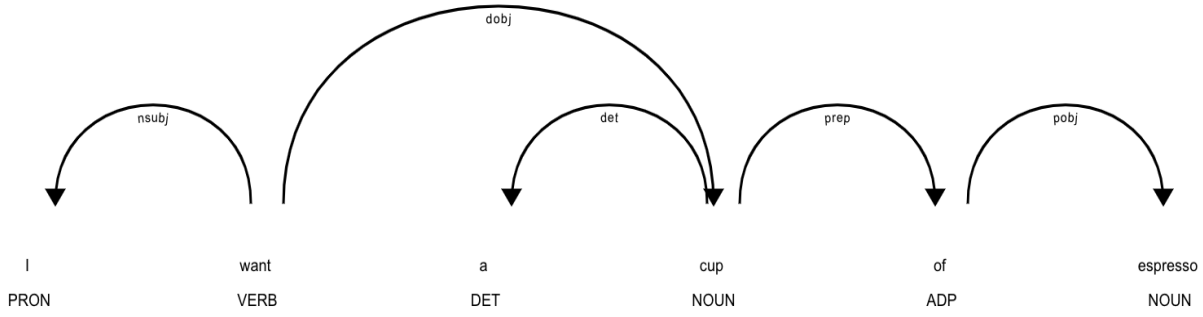


The dependency tree above is the output of Spacy's dependency parsing for the customer's input "I want some Cappuccino, please".

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. The statistical model consists of binary data and is produced by showing a system enough examples for it to make predictions that generalise across the language – for example, a word following "the" in English is most likely a noun.

From the dependency tree showed 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 "Cappuccino" marked as being dependent on "some" and "some" being dependent on "please" as used in the sentence.

Another dependency tree is provided below:



From the dependency tree above, each word in the noun phrase "a cup of espresso" are evidently dependent on each other exactly in the order that they follow, that is, "of" needs to come after "cup", and the noun "espresso" needs to come last.

We also see how the verb "want" depends on the noun "cup" which is the root of the noun phrase "a cup of espresso". Also pronoun "I" depends on the verb "want" as its subject.

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, for example, ordering beer instead of coffee or a different kind of coffee outside the scope.

When the system cannot parse the user's input, it responds with the words: "I am sorry, I don't understand, please come again", and when the user makes an order which is out of the scope of the scenario, it replies with the words:
"Sorry, we don't have [user request], here is what we have; …", it then goes on to list the available variants of coffee as coded in the scenario.

2.3 Tools

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

- **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.
- **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.
- **Pyttsx**: This is an offline text to speech conversion library in python, multiple tts-engine support. This library was used to convert every text output from Spacy to a voice output for the user.

3.0 Conclusion

The combination of SpeechRecognition, Spacy and Pyttsx 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.

3.0 References

1. https://spacy.io/
2. https://pypi.org/project/SpeechRecognition/
3. https://pyttsx.readthedocs.io/en/latest/