Spoken Human Robot Interaction report

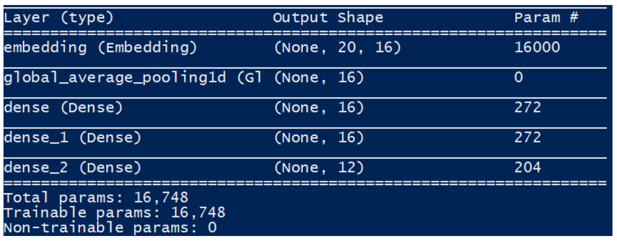
Edoardo Montecchiani 1969618

The scenario of this robot is a hotel receptionist. The robot is able to understand and respond to most of the phrases included in a discussion with a receptionist in a small hotel. The robot is named Poe.

All code is written in Python in a windows environment thanks to the use of some external libraries. The libraries used are these:

- **Google Speech Recognition**: This library is used in order to perform speech recognition. (speech to text). The microphone will detect any phrase you say in English and turn it into text;
- **Google Text-To-Speech**: This library is used to vocally synthesize the output of the robot (text-to-speech);
- **spaCy**: This library is the heart of text analysis, is used mainly as dependency parser;
- Other python auxiliary libraries such as numpy tensorflow and random.

The code consists of a main loop where you can ask the robot anything. A first analysis to understand what is being asked, is done thanks to a pre-trained neural network. The neural network is trained with a little dataset, a json file, where there are some sentences each associated with a tag. It is a very simple neural network with an embedding and 2 middle Dense layers with 16 units and a last Dense with the number of tags it is able to recognize.



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 20, 16)            16000
_________________________________________________________________
global_average_pooling1d (Gl (None, 16)                0
_________________________________________________________________
dense (Dense)                (None, 16)                272
_________________________________________________________________
dense_1 (Dense)              (None, 16)                272
_________________________________________________________________
dense_2 (Dense)              (None, 12)                204
=================================================================
Total params: 16,748
Trainable params: 16,748
Non-trainable params: 0
```

Figure 1Neural network for first analysis

The user can say any sentence, if the neural network can classify into one of its tags with at least 60% confidence then a response is processed based on that tag.

Tags are differentiated into two groups, those that represent sentences or questions to which a single sentence is sufficient as an answer and there is no need to analyze the sentence further. And those that instead need deeper analysis and if they haven't received enough data, they interact with the user and request more information related to the request.

The possible responses to requests classified in a tag in the first group are written within the same json file for training the neural network. As soon as a certain tag is recognized, a response phrase will be randomly chosen from those in this json file. Tags belonging to this group represent all the pleasantries and some hotel information. In particular they are:

- Greeting: When you greet Poe, Poe responds by greeting you in turn.
- Bye: this tag is very important because when a sentence is classifies in this tag, Poe says goodbye and the program ends. It is the only way to terminate the conversation. To be classified in this tag you can simply say goodbye or say some keywords like "cancel" or "quit".
- Thanksgiving: Operable if you thank Poe.
- Presentation: If you ask who he is or what his name is, Poe will tell you.
- Helping: Operable if you ask Poe if it can help you.
- Breakfast: If you ask for some information about breakfast.

All other tags are more elaborate and each one is associated with a python function that, depending on what it serves, performs different types of textual analysis and interacts with the user. The tags in question are:

- Booking: To book a room.
- Check-In: To do the check-in in a room.
- Alarm: To set an alarm.
- Key: To request your key room.
- Cost: To know the costs of the rooms.
- Taxi: To ask for a taxi.

Let's analyze each one individually.

Booking:

This is probably the most important tag. It allows you to book any of the available rooms and choose the length of stay. The available types of room are “single”, “double”, “quadruple” or “suite”. If you already say in the initial sentence the type of room you want and/or how long you want the room for then automatically they are recognized. If not, Poe will ask you first what kind of room you want. Room type detection is simply checking to see if there is a certain type of room within your sentence. In fact, the bot will explicitly ask you to choose one of the four available room types. After that, Poe will ask you how long your stay will be. SpaCy's names entity recognition is used to analyze this answer. You can say only a single number (entity “CARDINAL”) that correspond to the number or nights or you can do more complex sentence. It checks if there is an entity with tag “DATE”. So you can say for example 4 days or 2 weeks but also 2 week and 4 days (and it understands that it's 18 days) or something like “Up to Sunday” and it will calculate the number of days starting from today's real date until Sunday. During one of these two interactions you will still be able to ask for the cost of the room which will temporarily trigger the other action corresponding to the Cost tag (if he already knows the type of your room he will be able to tell you the price of that type, otherwise he will ask you what type you want to know the price of). After defining the type of room and the length of stay you will be asked for an explicit confirmation of the reservation with the total price. After an explicit confirmation we will pass to the Check-In by executing the function belonging to the Check-In tag.

Check-In

This feature is designed for those who already have a reservation or have just booked. So you can activate this tag by saying that you have a reservation or by expressly asking for the check-in or as mentioned above is invoked automatically after action “booking”. Poe will ask your first and last name to check in, which will always be detected thanks to the name entity recognition as entity "PERSON". He will then proceed to "hand over" the room keys to you.

Alarm

If you ask Poe to set an alarm or wake you up at a certain time it will activate this feature. If you don't already say in the first sentence a certain time, then Poe will ask you at what time you want to wake up.

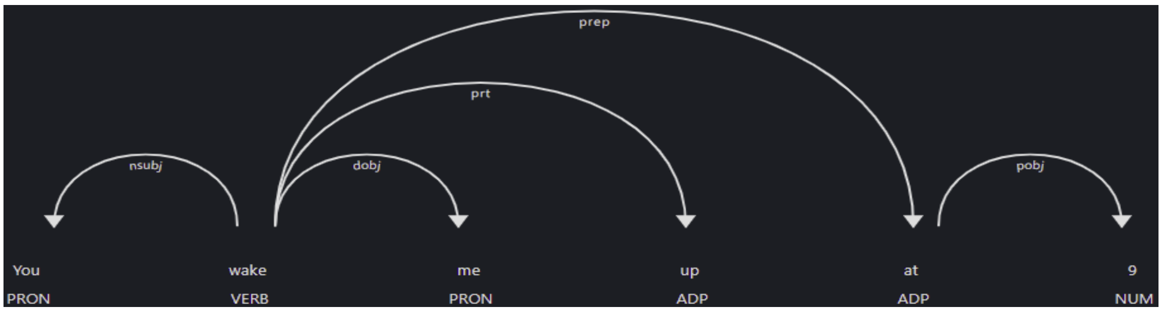


Figure 2 SpaCy display library. plot "you wake me up at 9" with dependencies

There are 3 different controls to check the time. The first, the most common, checks if you said an "at" which usually always accompanies a time in a complete sentence. Scrolling down the dependency tree (Figure 2) we notice that the child of “at” corresponds to the time. If you don't say the keyword "at" (for example if you respond directly only with a single number or a single hours) then check is done on the entities. If "Cardinal" or even better "TIME" entity is present the alarm is set to the given time.

Key

If you ask Poe for your room key he will ask you for the room number (if you haven't already said it in the first sentence) and he will "give" you those keys. Spacy will check if there are any numbers in your sentences and he will figure out the number of your room that you want the keys to.

Cost

Cost, as said before, you can also invoke it within booking. In the first sentence, thanks to a dependency analysis, it will be checked if the direct object (or a child of it) corresponds to a type of room, otherwise it will be asked explicitly of what type of room you want to know the cost per night and an appropriate answer will be elaborated.

Taxi

You can also ask Poe to call you a taxi. To do this Poe needs a destination. If you did not specify the destination in the initial sentence then it will be asked explicitly. In the main sentence, the destination is extracted by checking to see if there is a word "to" that is preposition (important because "to" is used very often as an auxiliary. "I need a taxi to go to the airport" has the first "to" as an auxiliary and the second as a preposition). The child in the dependency tree of this preposition will correspond with the destination. Otherwise, if not made explicit in the main sentence, just say the single name of the destination.

In every single moment, inside each function, it is possible to tell Poe to cancel and return to the main loop with some keywords like "cancel" or "quit".

In conclusion, I tried to create this robot with the idea that it could replace a human hotel receptionist. Obviously, the functionalities are not enough to really do this, but the data it can obtain could be easily managed with a database. I also tried to vary as much as possible the type of analysis of the text, both analyzing the dependencies and the entities depending on the situation, and you can very easily add new features that use these types of analysis.