EmGiBot: the World’s first

Voice Assistant that sees you!

Emanuele Alessi (1486470)

Gianmarco Forcella (1725967)

Sommario

[1. Introduction 3](#_Toc28255794)

[2. Theoretical overview 4](#_Toc28255795)

[2.1 Speech Recognition 4](#_Toc28255796)

[2.2 Speech Interaction 5](#_Toc28255797)

[2.3 OpenCV 5](#_Toc28255798)

[2.4 RASA NLU 6](#_Toc28255799)

[References 7](#_Toc28255800)

# Introduction

The last decade saw the introduction in the market of the *Intelligent Personal Assistant*, which rollout slowly began in 2010 with Apple’s **Siri** and, for four years after that date, these types of *bots* were relegated only to smartphones, with no luck of using them for domotic stuff. This changed in 2014, when Amazon launched its new device, **Amazon Echo**, that was equipped with **Alexa**: this was literally a game changer for the *I.P.A.* sector, as it opened the doors also for **Google Home** and Apple’s **HomePod.**

None of these companies, though, ever thought of making an *Intelligent Personal Assistant* for computers that combines the power of Semaphoric Gestures to stop a song or skip to the previous/next one.

For this reason, we thought of trying to make an experiment to see whether such a combination could be possible. And that successful experiment is called **EmGiBot**, which we are going to present in this report.

# Theoretical overview

Before going deeper with describing our solution, let us first describe all the possible *Multimodal* Interaction problems that we will face and, also, a brief description of the tools that we will use.

## Speech Recognition

**Speech Recognition** (or **Automatic Speech Recognition** (ASR), or **computer speech recognition**) is the process of converting a speech signal to a sequence of words, by means of an algorithm implemented as a computer program.[1]

This is the core and, naturally, the most important part of the entire project.

Building a Speech Recognizer relies, mainly, in N variables:

1. The **Environment** itself: if the place in which the user is using the IPA is too *noisy,* this will be troublesome as the program won’t be able to understand very well when an **utterance** **actually ends or not**;
2. The **Speaker**, also, can make a huge impact on the performance of the Speech Recognition: there can be a tremendous difference whether the user uses an *external microphone* or the computer’s *internal microphone;*
3. Last but not least, the way the user **speaks** also makes a huge difference: the speed of the utterance can, in fact, impact what the algorithm understands and, then, “translate” to written language.

Though there are 3 different ways in doing a possible Speech Recognition, we decided to rely on the **Artificial Neural Networks** methodology.

## Speech Interaction

**Speech Interaction** doesn’t have any peculiar definition: it’s just what it is. The way with which we humans communicate to each other.

In terms of abstraction, the **Speech Interaction** has 3 layers[2], as it can be seen from the below image.

Immagine che contiene testo

Descrizione generata automaticamente

Figure 1 The Stack of abstraction of the Speech Interaction. From bottom to top: the phonetics area; the linguistic area; the meaning area

A Multimodal system that wants to try to imitate a human’s speech, needs to perfectly be able to work with all of these layers, since it has been proved that **Speech Interaction** is probably the best way to interact with a human being.

## OpenCV

**OpenCV** (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. **OpenCV** was built to provide a common infrastructure for computer vision applications.  
The library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. These algorithms can be used to detect and recognize faces, identify objects, classify human actions in videos, track camera movements, track moving objects, etc.

**OpenCV** is a cross-platform library using which we can develop real-time computer vision applications. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection and it has more than 47 thousand people of user community and estimated number of downloads exceeding 18 milion.  
The library is used extensively in companies, research groups and by governmental bodies.

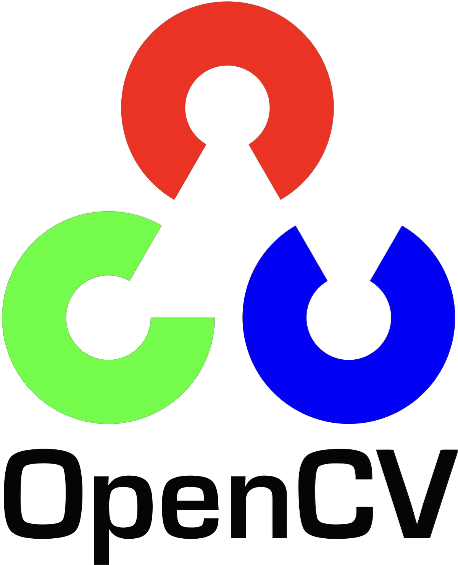


Figure 2 OpenCV’s logo

## RASA NLU

**Rasa NLU** is an open-source natural language processing tool for intent classification, response retrieval and entity extraction in chatbots. Probably, it’s the best opensource library available in the market.

Based on the **Long Short Term Memory** (**LSTM**) **Neural Network** **Model**, **RASA** takes in input a dictionary on <*intent>* - <*possible sentences*> and, then, the Machine Learning model does the magic.

Once it has been trained, **RASA** is ready to perform all of its logic to a new input sentence and try to understand the intent of this phrase. This usually has an accuracy of at least **80%** even if that intent only has 3 associated sentences.

It relies on SpiCy’s corporas.



Figure 3 RASA's logo

# Project setup

Let us know enlist the technologies that we used while working on this project.

* **Python 3.x**
* **PyCharm as our IDE**
* **NumPy**
* **OpenCV**
* **RASA**
* **Selenium + PhantomJS** for allowing songs to play from **YouTube**;
* **NewsAPI.org** for fetching the latest news;
* [**Speech\_Recognition**](https://github.com/Uberi/speech_recognition) for the **Speech Recognition** part;
* [**Pyttsx3**](https://github.com/nateshmbhat/pyttsx3)for the **Speech Interaction** part;

To check out the full project, please refer to [this](https://github.com/TheGlobalist/MultiUseAssistant) GitHub repository, which also contains all the papers we took inspiration from for our work.

## Typical Workflow

The following picture represents a typical workflow for **EmGiBot.**

**Immagine che contiene oggetto

Descrizione generata automaticamente**

Figure 4 EmGiBot's workflow

* From a device of any kind (for now, just a desktop computer. In the future, perhaps, more!), the user performs a request through **Speech Interaction.** This is called the **Event Phase;**
* The server then receives the request. The **Compute Phase** starts;
* The **Language Phase** is all dedicated to the core part of the application. It is divided into:
  + **Speech Recognition Part**: through *Google’s Cloud Speech API*, we make a call to convert from voice to text;
  + **Natural Language Understanding Part**: after converting the voice to text, we take that to *RASA* in order to understand the user’s intent
* Depending on which intent was found, the **Data Gathering Phase** starts:
  + **News** will be gathered if and only if *RASA*labels the request for that;
  + **Music** will be played if and only if *RASA* labels the request for that;
    - Should music be requested, then we also enter in the **Tracking Phase**, through which we understand (thanks to *OpenCV*) whether to go to next / previous song or stop the playback;
  + **Help** on how the *IPA* works will be dispatched if and only if *RASA*labels the request for that

## Technical Limitations

In order to develop **EmGiBot**, we wanted to build / use four different Neural Networks:

1. One that could perform the NLU part, something for which we decided from the very beginning to use *RASA;*
2. One that could be able to perform the **Speech Recognition;**
3. One that could perform **Speech Interaction;**
4. And one that could possibly be able to track and recognize the user’s movement.

But since all the development was carried out in two machines with similar computational power (both equipped with a *3,1 GHz Intel Core i5,* 8GB of RAM and an NviDia GeForce 130MX graphic card), we knew from the very beginning that **EmGiBot** would have never gotten too far because of the high calculation power that it required.

For this reason:

1. The only Neural Network that we decided to keep was *RASA’s*, because of its power and precision even with just a small dataset;
2. We decided to give the **Speech Recognition** part to *Google’s Speech API*, because of its tremendous power on getting right the sentence;
3. Leave the **Speech Interaction** part to Python’s *pyttsx* library, which seemed perfect for the task;
4. And use *OpenCV* to detectthe gestures.

Doing so granted **EmGiBot** to perfectly work on the computers used, even on a *RaspBerry Pi* 4 equipped with a camera and a microphone.

# References

[1] Course’s Slide #06, Speech Recognition

[2] Course’s Slide #07, Speech Interaction