Spoken interaction with MARRtino

Daniele Nardi,

Andrea Vanzo

Dipartimento di Ingegneria Informatica, Automatica e Gestionale

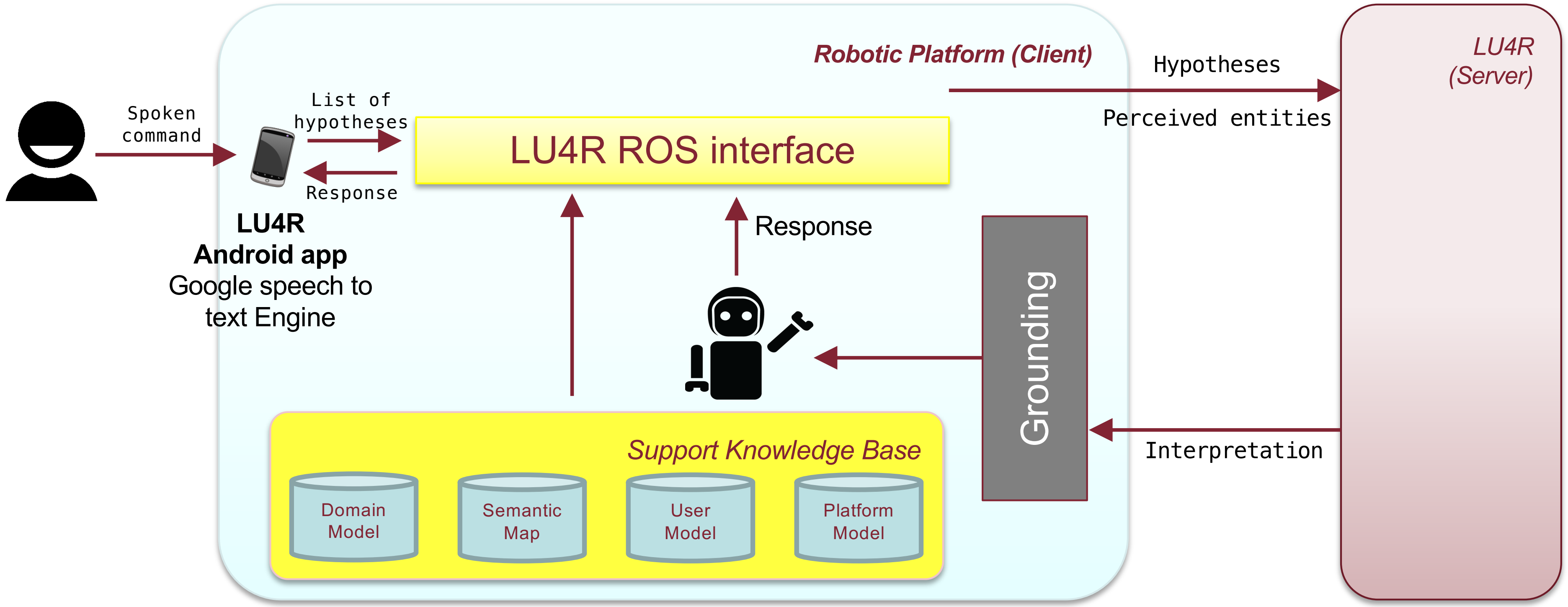
nardi@dis.uniroma1.it

http://rococo.dis.uniroma1.it

SAPIENZA UNIVERSITÀ DI ROMA

LU4R
The Architecture

- Client/Server architecture
  - Completely decoupled from the Robotic Platform
  - Can be invoked through simple HTTP POST requests



LU4R
adaptive spoken Language Understanding chain For Robots

- A Spoken Language understanding tool for robotic commands
- Based on
  - **Frame Semantics [Fillmore, 1985]**
    - psycho-linguistic theory about the lexicon
    - semantic frames (e.g. *Motion*, *Taking*, …) correspond to situation evoked by the lexicon
    - each frame involves different semantic arguments (e.g. GOAL, THEME)
- ..and relies on
  - **A cascade of data-driven processors**
  - **Perceptual evidence**
    - Interpretation depends on the environment configuration (e.g. semantic map)

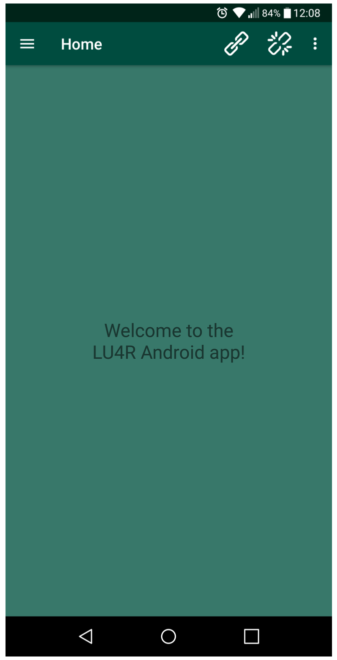
LU4R
adaptive spoken Language Understanding chain For Robots

- Input (JSONs):
  - Transcription of a spoken utterance
    - E.g. "*take the glass near the volume on the table*"
  - List of entities perceived in the environment (and their position)
    - E.g. `glass(1.0, 2.0), book(2.0, 2.0),…`
- Output (CFR):
  - Interpretation of the spoken command in terms of frames

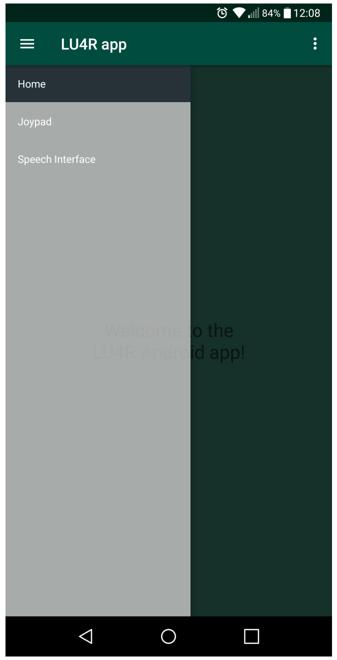
```
BRINGING(theme:"the glass near the volume",goal:"on the
table")
```

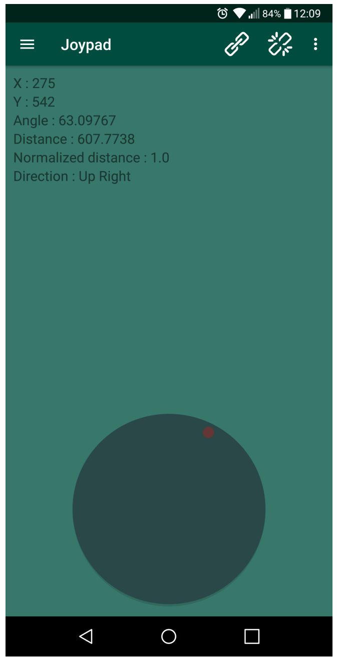
LU4R Android app

- Android application that provides two functionalities:
  - ASR for LU4R
  - Virtual joypad

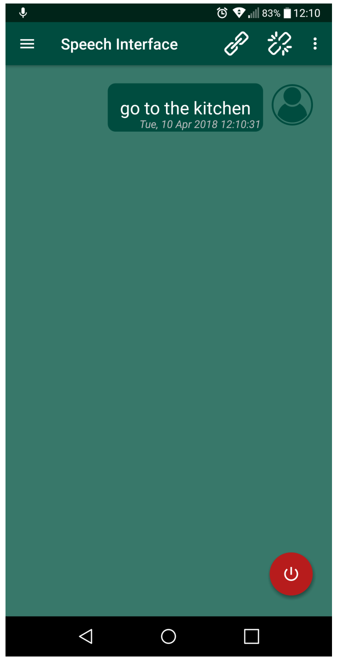


Welcome to the
LU4R Android app!





```
X : 275
Y : 542
Angle : 63.09767
Distance : 607.7738
Normalized distance : 1.0
Direction : Up Right
```



go to the kitchen
Tue, 10 Apr 2018 12:10:31

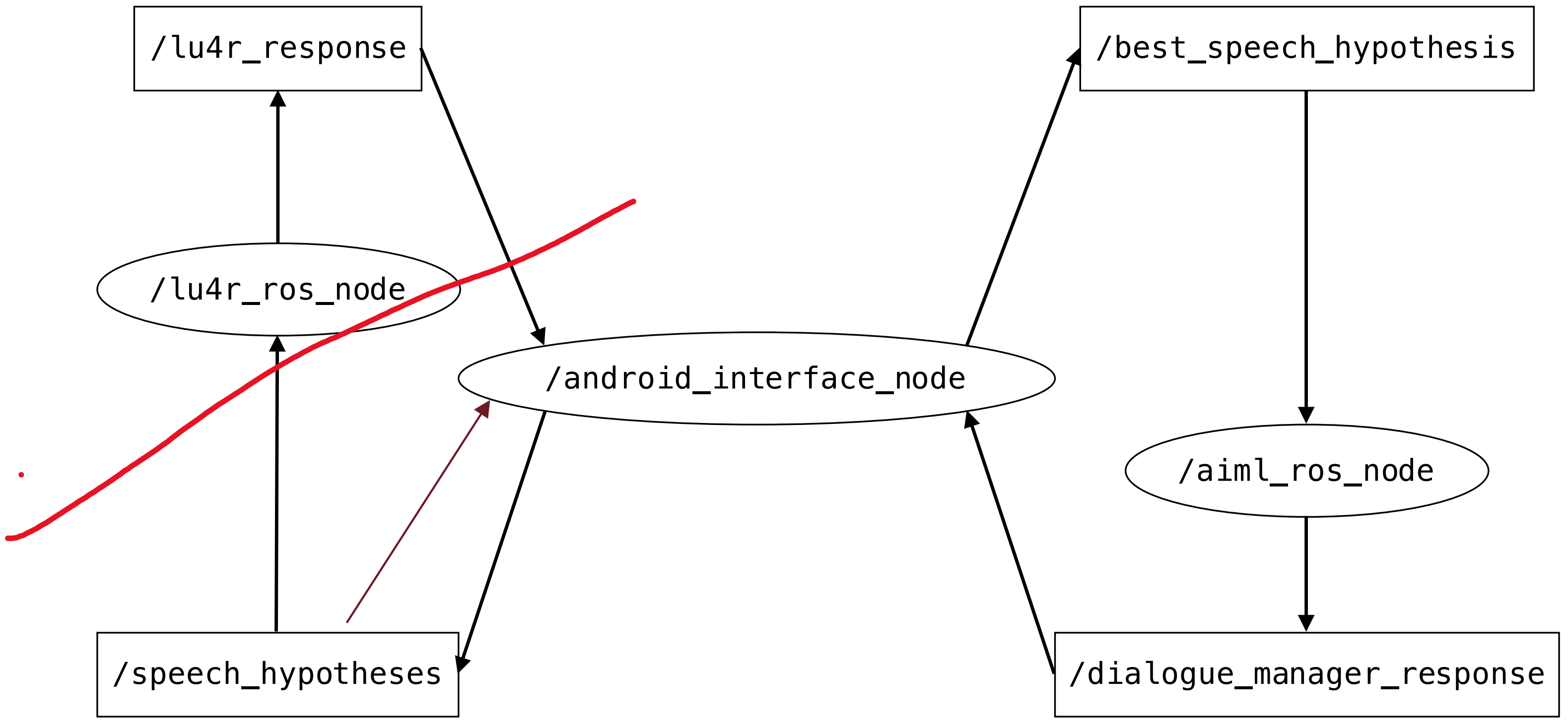
- More details at: https://www.marrtino.org/software/app-speech

LU4R ROS interface

- Collection of ROS nodes that enable a full integration of LU4R into the ROS environment
  - android_interface: main orchestrator of the system
    - https://gitlab.com/andreavanzo/android_interface
  - aiml_ros: interface to an AIML KB
    - https://gitlab.com/andreavanzo/aiml_ros
  - lu4r_ros: interface to LU4R server
    - https://gitlab.com/andreavanzo/lu4r_ros
  - framenet_ros_msgs: provides a mapping between ROS msgs and Semantic Frames
    - https://gitlab.com/andreavanzo/framenet_ros_msgs

LU4R ROS interface

- Communication between nodes through ROS publisher/subscriber protocol over topics



AIML

- Markup Language
- *"Stimulus/Response"* (S/R) pattern (used in common chatbots)
  - **Stimulus** represents what the user may say, and is the input of the Interpreter
  - **Response** represents what the user expects as answer, given the corresponding stimulus. It is the output, that can be:
    - A string
    - A system call

Richard Wallace. (2003). The Elements of AIML Style.

AIML
Basics

- `<aiml>` begins and ends an AIML document
- `<category>` defines a unit of knowledge (interaction)
- `<pattern>` matches what the user may say or type (stimulus)
- `<template>` defines the response to the user’s input (response)

```
<aiml>
  <category>
    <pattern>HELLO</pattern>
    <template>Hi human!</template>
  </category>
  <category>
    <pattern>GO TO THE KITCHEN</pattern>
    <template>action:GOTO_dest:kitchen</template>
  </category>
</aiml>
```

Richard Wallace. (2003). The Elements of AIML Style.

AIML

RegEx

- Supports the use of regular expressions inside the `<pattern>` tag

```
<aiml>
   <category>
    <pattern>HELLO I AM *</pattern>
    <template>Hi <star/>!</template>
  </category>
   <category>
     <pattern>GO TO THE *</pattern>
     <template>action:GOTO_dest:<star/></template>
   </category>
</aiml>
```

AIML
<that/> tag

- `<that>`: special type of pattern element used for context matching

```
<aiml>
   <category>
    <pattern>WHAT ABOUT MOVIES</pattern>
    <template>Do you like comedy movies?</template>
  </category>
  <category>
    <pattern>YES</pattern>
    <that>Do you like comedy movies</that>
    <template>Nice, I like comedy movies too.</template>
  </category>
  <category>
    <pattern>NO</pattern>
    <that>Do you like comedy movies</that>
    <template>Ok! But I like comedy movies.</template>
</aiml>
```

Richard Wallace. (2003). The Elements of AIML Style.

AIML
Variables

- `<get name=“var”/>` to **get** the value of a variable
- `<set name=“var”/>` to **set** the value of a given variable

```
<aiml>
   <category>
    <pattern>*</pattern><that>WHAT IS YOUR NAME</that>
    <template>Hi <set name=”user”><star/></set>!</template>
  </category>
  <category>
    <pattern>WHAT IS MY NAME</pattern>
    <template>You are <get name=“user”/></template>
  </category>
</aiml>
```

AIML
<topic/> tag

- `<topic name=“context”>` stores a context so that later conversation can be done based on that context

```
<aiml>
   <category>
    <pattern>LET DISCUSS MOVIES</pattern>
    <template>Yes <set name=“topic”>movies</set></template>
  </category>
  <topic name=“movies”>
     <category>
       <pattern>*</pattern>
       <template>Watching good movie refreshes our minds.</template>
     </category>
     …...
   </topic>
</aiml>
```

AIML: random responses

- `<random>` allows to select one of a list of responses randomly

```
<aiml>
 <category>
   <pattern>HELLO I AM *</pattern>
   <template>
     <random>
       <li>Hi <star/>!</li>
       <li>Hello <star/>, nice to meet you!</li>
     </random>
   </template>
  </category>
</aiml>
```

Richard Wallace. (2003). The Elements of AIML Style.

AIML: chatbot

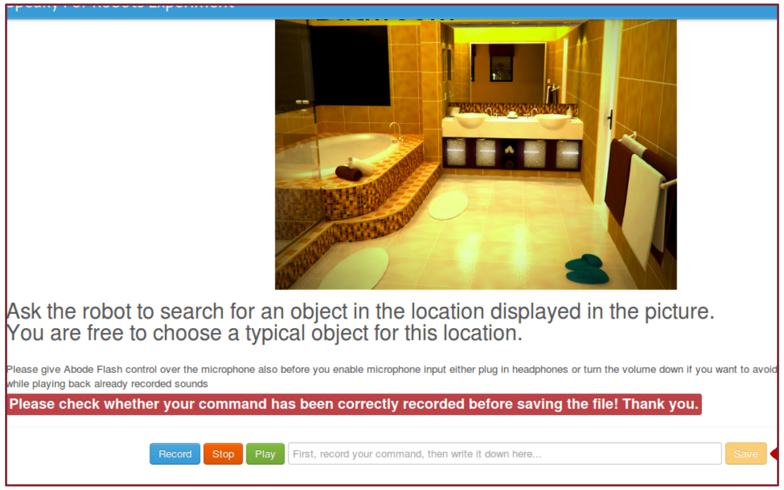
To create a simple dialogue manager:

- Create a set of files with aiml specifications (.aiml). They can be arranged in directories and subdirectories for a modular development
- Use a simple program in python to handle question answering (chatbot_tester. Py)

See aiml.zip for the tester program and some example files.

Our testing corpus: HuRIC

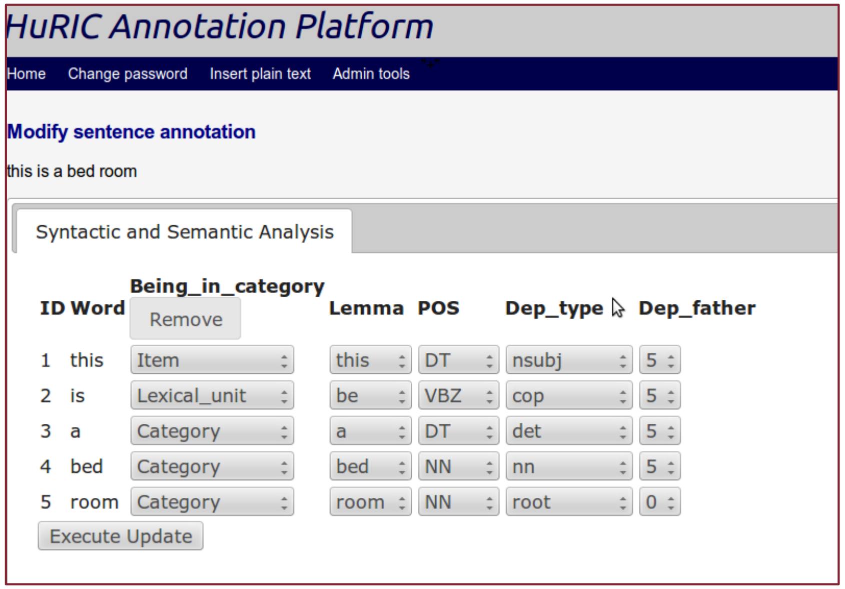
- Audio files, Transcriptions, POS-tagging, Dependency trees, Frame Semantics, Spatial Semantics, Abstract Meaning Representation
- English
  - 656 annotated spoken commands (1200 audio files)
- Italian
  - 241 annotated spoken commands (331 audio files)
- Presented at LREC 2014 (further improved at IJCoL)
- Still growing



Ask the robot to search for an object in the location displayed in the picture.
You are free to choose a typical object for this location.

Please give Abode Flash control over the microphone also before you enable microphone input either plug in headphones or turn the volume down if you want to avoid while playing back already recorded sounds

Please check whether your command has been correctly recorded before saving the file! Thank you.



HuRIC Annotation Platform

Modify sentence annotation

this is a bed room

Syntactic and Semantic Analysis

ID	Word	Being_in_category	Lemma	POS	Dep_type	Dep_father
		Remove				
1	this	Item	this	DT	nsubj	5
2	is	Lexical_unit	be	VBZ	cop	5
3	a	Category	a	DT	det	5
4	bed	Category	bed	NN	nn	5
5	room	Category	room	NN	root	0

Project

Design the Dialogue Manager (an AIML KB) to execute the set of commands that you have implemented in MARRtino. The dialogue manager must implement also some dialogue, when the command is not executable by the robot (Download the app from:

https://www.marrtino.org/software/app-speech).

- The KB should be a collection of AIML files, that will be interpreted by the aiml_ros node. This requires the use of the android app and of the AIML-ROS node (but not the "lu4r_ros" node and server).

Note :

1) to run the code without the LU4R node, uncomment the line:

self.best_hypo_publisher.publish(self.best_hypo)

2) you have to download the ROS package "framenet_ros_msgs" to avoid dependency issues.

3) If you run Python 3, then few modifications are needed.

Project (Optional)

You can run also LU4R, in which case you have to add the node lu4r_ros and install the server LU4R.

To get the server code write to vanzo@diag.uniroma1.it).

To handle the dialogue Modify the android_interface node (Python), whenever LU4R returns the message “NO FRAME(S) FOUND”