UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE ED ELETTRICA E MATEMATICA APPLICATA

Computer Engineering

Cognitive robotics

Pepper in a shopping mall

Final project

Professors:
Prof. Alessia Saggese

Students:
Carmine Aliberti
Giovanni Casella
Christian Salvatore De Angelis
Nunzio Del Gaudio

YEAR 2024/2025

Contents

Chapter 1

Introduction

Our project aims to program Pepper, the humanoid robot, to function as a social robot capable of interacting seamlessly with users. The programming will be conducted using the Robot Operating System (ROS), which provides a robust framework for robotic development.

The primary goal is to enable Pepper to respond to spontaneous, non-predefined questions within a known domain. Specifically, Pepper will access a database containing information about a shopping mall. It will be able to inform users about topics such as the number of customers currently present, the number of customers carrying a shopping bag, and other similar statistics.

This project emphasizes real-time interaction, effective data retrieval, and creating a user-friendly experience, making Pepper a valuable asset in environments requiring human-robot interaction.

Chapter 2

Ros Architecture

2.1 Intro

We will use the Robot Operating System (ROS) to develop Pepper's functionalities. The programming is carried out on a Linux machine running Ubuntu 20.04 LTS, a reliable environment for ROS-based projects.

The philosophy of the project is based on modularity and independence between nodes. For this reason, even though many nodes could have been integrated (without the use of topics and services), we decided to keep them separate in case of future improvements or feature additions.

For example, it would have been possible to integrate the animation client with the animation server in only one node, but by using the animation server, we were able to decouple from potential active waiting issues related to the ALAnimation module. Additionally, this approach allows the animation client to subscribe to multiple topics, making the animation independent of multiple sources of information (currently, it depends only on when the robot speaks, but in the future, it could be triggered by other events).

2.1.1 Graph Representation of the Architecture

In the architecture diagram, we adopted the following conventions for the graphical representation:

- **Transparent ellipses**: Represent nodes.
- **Gray ellipses**: Indicate server nodes.
- **Black ellipses**: Represent services provided by Pepper's APIs.
- **Green ellipses**: Indicate the connection with the chatbot.
- **Squares**: Represent topics.

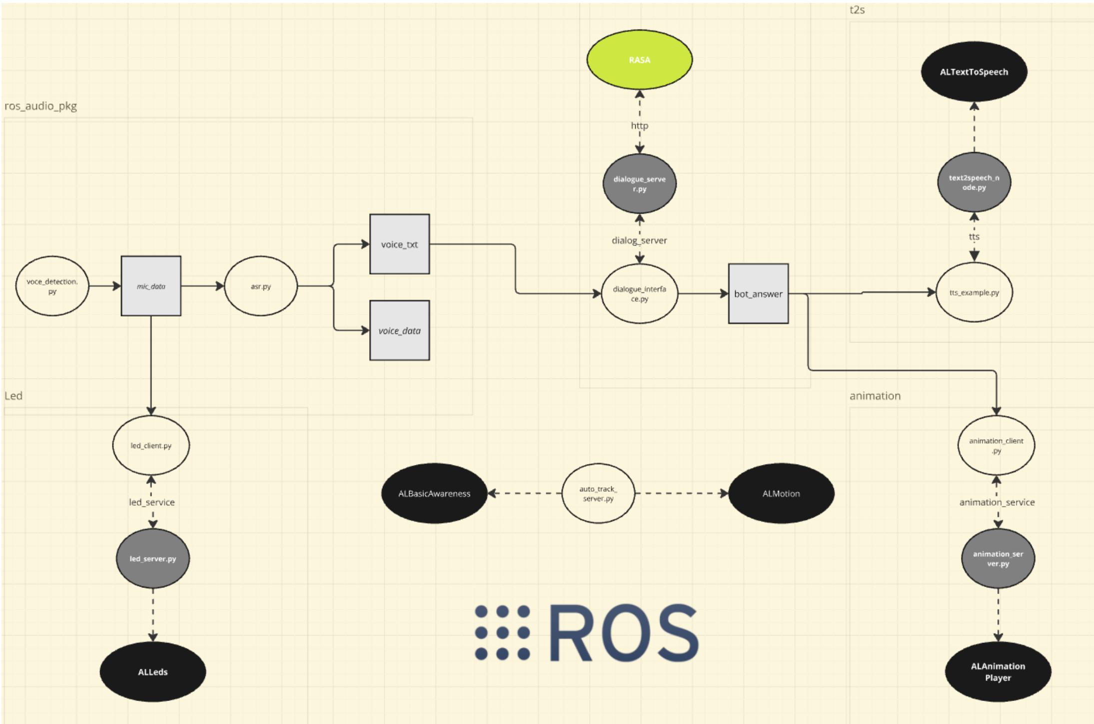

Figure 2.1: Architecture.

2.2 Input Audio Nodes

The separation of audio's functionalities into two distinct nodes ensures independence between the audio acquisition method and the audio processing method, providing modularity and flexibility in the system's architecture in case of change of the capture device or the audio processing method.

2.2.1 Voice_Detection

This module uses the microphone to acquire the audio track of the user's speech. The energy threshold, determined through tests with various values, is fundamental for the audio acquisition node to understand when a recording is considered complete (i.e., when a person has stopped speaking). The audio Pepper listens to is published in the **mic_data** topic.

2.2.2 ASR

This module subscribes to the **mic_data** topic and utilizes Google Speech Recognition to transcribe audio tracks into text. The transcription is then published to the **voice_txt** topic for further use. Additionally, the audio is published to the **voice_data** topic to facilitate future processing.

Word recognition is not perfect, both due to the limitations of the ASR module and the potentially imperfect pronunciation of the interlocutor. For this reason, a function named `_word_replacement()` was added to the ASR module. Its role is to replace similar words within the recognized phrase. Specifically, it searches for words in the sentence that are very similar to those expected in the conversation with the bot.

This process is performed in two ways:

- **Deterministic Mapping**: Fixed correspondences determined during testing are always applied (e.g., "Peppa" is always transformed into "pepper").
- **Similarity Matching**: For other words, a similarity check is performed using `get_close_matches()` with a confidence threshold of 0.75, set high to minimize false positives.

This approach was viable only because the robot is expected to handle specific types of queries. Without this context, such reasoning would have had more drawbacks than advantages

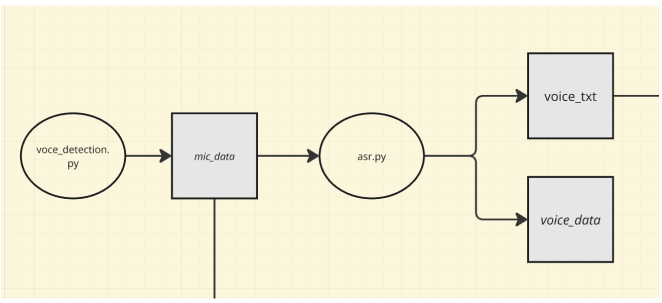

Figure 2.2: Input Audio nodes architecture.

2.3 Rasa Nodes

The RASA nodes are all nodes whose primary task is managing both the transmission of the user's spoken phrase and the handling of the RASA bot's response.

2.3.1 Dialogue_interface

The dialogue interface node is the beating heart of the entire project. This node acts as a bridge between the textual version of what the user says and the responses provided by RASA. Its primary role is to read from the `text_data channel`, interface with the dialogue server node, and write the received response to `bot_answer`.

2.3.2 Dialogue_server

This node provides an intermediary service that forwards user requests to a Rasa server, subsequently receiving and handling the corresponding responses. It was choose a service over a publisher/subscriber because we want the answer to the corresponding question. In case of publisher/subscriber could happen that the node could respond with an answer that does not refer to the last message.

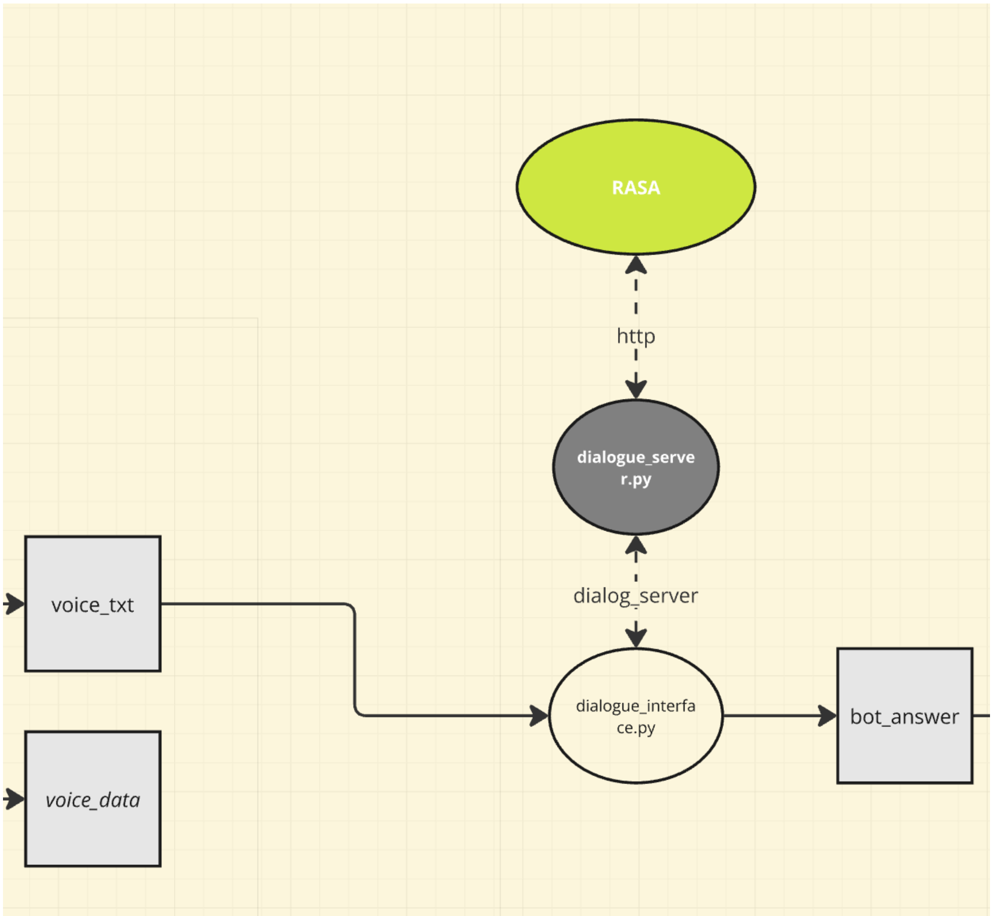

Figure 2.3: Rasa nodes architecture.

2.4 Animation Nodes

These nodes are designed to instruct Pepper's API on which animation to perform.

2.4.1 Animation_client

The `animation client` node is responsible for deciding which animation Pepper should perform. Specifically, it was designed to subscribe to the `bot_answer` topic, allowing Pepper to perform animations during speech. By reading Pepper's responses, it becomes possible to trigger different actions depending on what Pepper says.

By default, body talk animations were configured unless specific keywords, such as `Happy` or `hello`, are detected.

2.4.2 Animation_server

The `animation server` node expects a tag to be passed to Pepper's API. It was decided to use the `runTag()` API to introduce randomness into the animations. By passing the tag to the API, it will randomly select one of the animations associated with that tag.

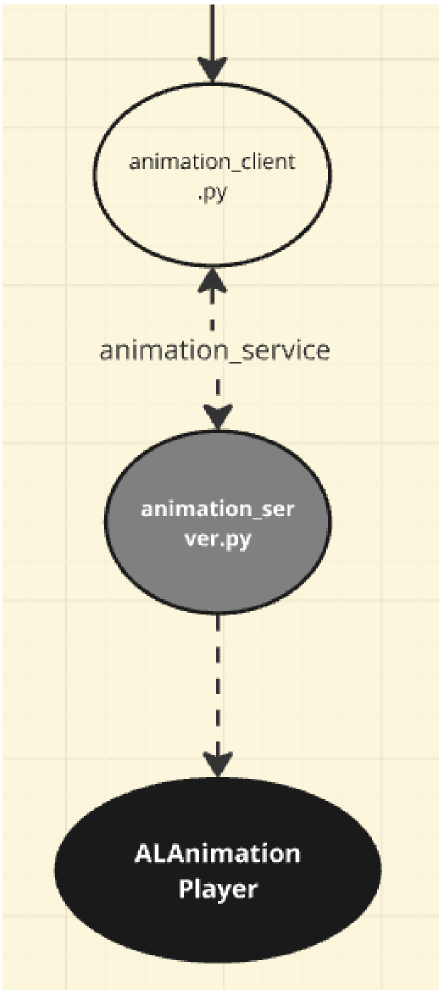

Figure 2.4: Animation nodes architecture.

2.5 Led Nodes

These nodes have the sole purpose of activating Pepper's eyes to provide a visual output that is not limited to animations. This serves to indicate to the user that their voice message has been successfully received.

2.5.1 Led_client

The main purpose of the `LED client` node was initially to subscribe to both the `mic_data` topic (to indicate that something was heard) and the `bot_answer` topic (to indicate that it was no longer necessary to keep the eyes on). However, at a later stage, it was decided to activate the eyes for a predetermined amount of time, making the sole task of this node to request the server node to activate the eye LEDs.

Although the node is no longer essential for the current architecture, it was kept in case it is needed in the future to make the LED logic more complex.

2.5.2 Led_server

The `LED server` node simply calls the `randomEyes()` API for a predefined amount of time.

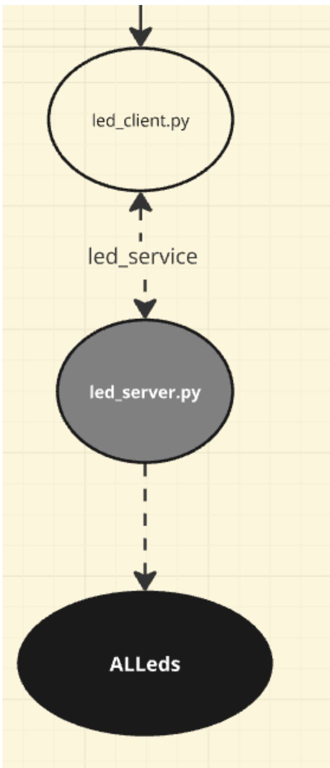

Figure 2.5: Led nodes architecture.

2.6 Text to speech Nodes

These nodes are essential for enabling Pepper to speak. Their task is to have Pepper say what was output by the RASA bot.

2.6.1 tts_client

The `TTS client` is subscribed to the `bot_answer` topic and its sole task is to request the intervention of the `text2speech` server to make Pepper speak what it has read.

2.6.2 text2speech_node

The `text2speech` node handles all the logic related to Pepper's speech, calling the appropriate APIs and setting the correct language.

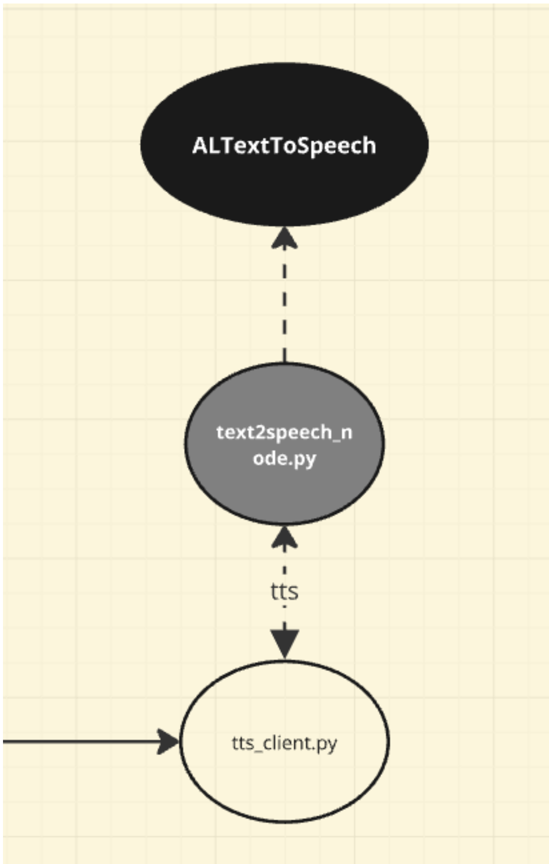

Figure 2.6: Speech nodes architecture.

2.7 Auto_track Node

This node is responsible for preparing Pepper for the dialogue. It calls the necessary APIs to position Pepper correctly (i.e., movement) and requests Pepper to follow people with its gaze, prioritizing facial tracking first and then sound. This node is not subscribed to any topic and can be considered as a setup node for Pepper.

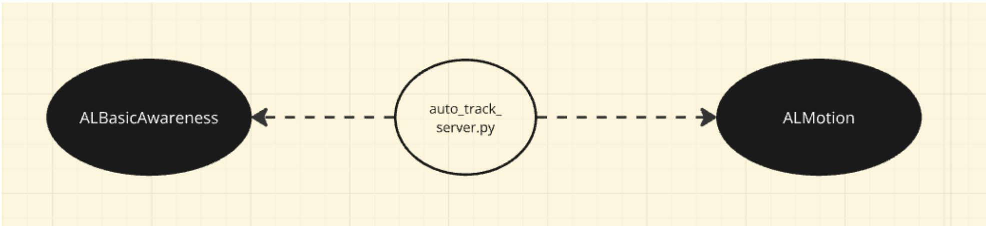

Figure 2.7: Set-Up node architecture.

2.8 Pepper API

This section will present the APIs used by Pepper to implement its functionalities. Among them:

- **AlBasicAwareness**: This service is designed to enhance Pepper's ability to interact naturally with its environment. It enables the robot to perceive and track human presence, paying attention to visual and sound.
- **AlAnimationPlayer**: This service enables Pepper to perform predefined animations based on a passed tag, enhancing its expressiveness and interactivity.
- **AlLeds**: This service provides control over Pepper's LED lights.
- **AlMotion**: This service enables Pepper to move its body and perform dynamic actions.
- **AlTextToSpeech**: This service allows Pepper to convert text into spoken language.

Chapter 3

Choices for Video analytics module and Input-Output Audio modules

3.1 Video Analytics

Initially, we decided to use the **efficientdet_d1_coco17_tpu-32** person detection model because it provided good results in both detection accuracy and system performance. The plan was to use this model with a custom detector as our final solution, and then focus on actual face tracking. However, this approach encountered complications, especially when trying to use the detector with Pepper's webcam. We aimed to create bounding boxes, but face tracking proved to be imprecise, additionally, the tracking obtained, in terms of the robot's movements, was not smooth: the robot's head movements appeared forced. Also, implementing sound-based localization from scratch was challenging. These considerations led us to the final outcome.
As a result, the final solution involved using a service called **AlBasicAwareness**, which made person tracking much simpler. It also incorporated sound localization and automatic head repositioning, resolving the issues we faced with the detector. Another alternative we considered was the OpenCV library. Nonetheless, this design decision provided robust and reliable results, especially in scenarios with varying distances between Pepper and the individuals being tracked. The final tracking results in smooth movements that do not appear forced. Moreover, the robot is able to keep track of the person it is talking to at all times and can turn around when called.

3.2 Speech-to-text module

In this case, the decision to use Google Speech Recognition was made because OpenAI's Whisper offered a paid service that was unknown to us. Given our familiarity with Google Speech and its efficiency in utilizing this resource, we decided it would be the best solution for our needs. By leveraging Google Speech Recognition, we achieved good results, particularly in terms of its reliability and speed in converting spoken audio into text. The system demonstrated good accuracy in transcribing speech, ensuring that the robot could effectively understand and process spoken input. Once the audio was transcribed into text, it was seamlessly processed by Pepper, enabling the robot to generate contextually relevant and coherent responses. This integration of speech recognition allowed for natural, real-time interactions with Pepper, enhancing user experience and communication efficiency.

3.3 Pepper self-listening problem

Auto-listening on Pepper was one of the challenges we faced only after testing the bot on the robot itself. Initially, it did not occur to us that Pepper might listen to itself. Once we became aware of the issue, we considered several approaches to solve it:

1. **Subscribe the voice detection node to the `bot_answer` topic.** This would allow the node to know when to avoid listening to the microphone. While simple and effective, this solution was discarded due to the project philosophy of keeping the robot's different functionalities as separate as possible (the microphone listening node is conceptually distinct from RASA's responses).

2. **Disable the voice detection node for a predefined amount of time.** This solution is not very adaptable, as it assumes that all RASA responses are roughly the same length, which is not necessarily true.

3. **Implement a filter in the RASA interface node (`dialogue interface`).** In this solution, any input text sent to the bot that closely resembles a response previously given by the bot is considered auto-listening and is not forwarded to RASA. This appeared to be the most suitable solution but comes with a series of challenges:

    - **False positives:** If the user says something very similar to what Pepper has just said, the input might be incorrectly flagged as auto-listening. However, we considered this scenario extremely rare (and it never occurred during testing).
    - **False negatives:** Auto-listening detection might fail if the ASR does not correctly recognize the words, leading to the re-listened phrase being treated as a new one.

Despite these challenges, this solution seemed the most appropriate to adopt. However, we did not have the opportunity to test it on the real robot. For this reason, we decided to leave the option to disable this functionality within the node. If this feature is disabled, the only way to avoid auto-listening would be to turn off the microphone when Pepper is speaking.

Chapter 4

Dialogue management module

For the dialogue management module, RASA was chosen to implement the chatbot's functionalities. The option of using a pre-trained NLP model, such as GPT-4 or other versions, was also considered. However, the choice fell on RASA because the robot's context is limited to two domains: the contest and the mall. Consequently, it did not seem appropriate to add a heavy NLP model like GPT, which could potentially worsen the robot's response times and add an additional computational load to the existing audio and video tracking models. Furthermore, the use of the GPT model is not free, making it less convenient for testing purposes, even though the performance in terms of response accuracy would undoubtedly have been superior. The conversation design was created based on three main assumptions: Who is the user I am interacting with? What is the purpose (to define intents and entities)? What are the typical conversations? It was decided to build a Task-Oriented Dialogue (TOD) chatbot, which does not aim to answer all kinds of questions but rather focuses on having fewer conversation turns to solve a specific problem within a given domain.
The RASA module can be divided into two main areas:

- **Contest**
- **Shopping Center Guardian**

Both these areas share a common component, which is Pepper's interaction with people. This includes responses such as **utter_greet**, **utter_happy**, or **utter_goodbye**, which are common to both domains.

4.1 Contest

The main entities of the contest are **Group** and **Ranking**. The Group entity is used to identify the group number being referred to, while the Ranking entity identifies the position in the ranking.
For Group, a slot has been defined, which is used in actions to answer sequences of questions where the group is explicitly mentioned in the first part and implied afterward. For example:
Q: “Who is in group 3?”
A: ”The group 3 consists of x, x, x.”
Q: “And what was the afs of this group?”
A: ”The afs was x.”
An example of implemented actions is:

- **action_groups_number**: Provides the total number of groups in the competition, answering intents like **info_quantity** (e.g., ”What is the number of groups in the competition?”).
- **action_groups_composition**: Provides details about the members of a specified group, answering intents like **info_group** (e.g., ”What are the components of group one?”).
- **action_groups_ranking**: Provides the current overall ranking of the competition, answering intents like **info_ranking** (e.g., ”What is the current overall ranking of the competition?”).
- **action_group_rank**: Provides the rank of a specified group, answering intents like **info_group_rank** (e.g., ”What is the rank of group six?”).
- **General Information Action** : Responds to general queries about the contest, such as ”Tell me about the competition” or ”What is the afs?”.

4.2 Shopping Center Guardian

The main entities in the context are **Attribute** and **Person ID** and **Shops**. The Attribute entity aims to recognize characteristics of the person being searched for, such as male/female, hat, or bag, or a combination of these three. The Person entity is used to recognize an ID to facilitate the search for a person. For Person, an additional slot is maintained, allowing it to be queried when neither attributes nor the ID are provided in an intent. An example of conversation:
Q: “Have you seen a person with a hat and a bag?”
A: “There is one person matching your criteria. And it is ’7’”.
Q: “Do you know the position of this person?”
A: “Let me check in my database. Person 7 should have passed close to store PEPPER’S CAFE”.

For the **attribute**, we use a slot to complete the form required by the intent **ask for someone**. The form requires the attribute slot to be filled with values such as male/female, hat, and bag. Based on the data provided, it will return the people who match the specified criteria.

The **shops** entity is used to identify which store the user wants information about and can be combined with the attribute entity to identify a person in a specific store or the set of stores they have visited. The line are mapped to shops thanks to a dict, this is the mapping: **1:"PINCO PALLINO"**, **2:"UNISA"**, **3:"COGNITIVE STORE"**, **4:"PEPPER'S CAFE"**.

The implemented actions are as follows:

- **ActionCount**: This action counts the number of people who match specific criteria based on attributes like "hat," "bag," "male," or "female." When asked through the **ask_count** intent, it returns the number of people who meet the criteria and communicates this to the user.
- **ActionGetPersonInfo**: This action retrieves detailed information about a specific person identified by their ID. When the user inquires using the **ask_person_info** intent, the system provides all available information regarding the person.
- **ActionAttribute**: This action filters individuals based on the given attributes such as "hat," "bag," "male," or "female," and returns a list of IDs of the matched persons. The Ask_attribute intent is used to ask which people have a specific attribute, and the system responds by providing the matching IDs.
- **ActionGetTrajectories**: This action retrieves trajectory information (movements) of people based on the specified attributes. It indicates the most recent store where the person was seen. The **Ask_trajectory** intent requests the trajectories of people with certain attributes, and the system provides the relevant movement details.
- **ActionGetOtherTrajectories**: This action is used to track the movements of a specific person using their ID. It provides details about the person's trajectory and recent stores they visited. When the **Ask_trajectory** intent is used with a specific ID, the system will return the corresponding trajectory information.
- **ActionComparison**: This action is used to perform a quantitative comparison between the number of people with a certain attribute and the number of people with another attribute. It uses the slots "attribute" and "shops" to define the groups of people to count. It responds to the intent **Ask_comparison**, returning the number of people associated with the highest attribute among those requested and the attribute itself.

Finally, the robot's empathetic capability was implemented by adding intents that recognize whether a person is feeling happy or sad and respond accordingly. For example:
mood_unhappy: "I'm so worried"
mood_great: "I am happy"
In response to these intents, the bot will show concern or interest with questions such as "What happened? Can I help you?" for sadness or "Great! Tell me why you're happy, if you want" for happiness. This allows the user to inquire about a person they are searching for in the shopping mall or express happiness about their ranking, creating a more emotionally responsive interaction.

4.3 Issues

Considering the Contest area, the bot is unable to answer overly specific questions such as “How many groups are above group 14”. For simplicity, it was decided that, in such cases, the bot would respond by simply displaying the overall leaderboard. In rare instances, the bot confuses rank with group number. In most cases, the action manages to identify the issue and responds by asking the user to repeat the question. Another issue is the incapability to provide the trajectory prior to the one already given, and to fix this, it has been decided to return all available trajectories. Additionally, it does not respond dynamically based on attributes; for example, it does not reply using the feminine form when asked to find a woman, or vice versa. Instead, it uses the impersonal phrase ”people matching your criteria” rather than specifying gender. A final issue is that, when searching for a person, it is necessary to specify the attributes they possess, not those they do not possess. For example, ”he is a male with no bag.” In such cases, the chatbot considers the slot filled for ”bag” and will perform a search that treats ”bag” as valid.

4.4 Final Remarks on RASA

The main challenges in using RASA were encountered in defining intents and managing entities. The two domains, although very different, share some similar questions, such as “What is the total number of groups?” and “What is the total number of people in the shopping center?” This similarity can lead to confusion for the bot. A better approach could have been managing actions based on entities within a shared intent. The bot successfully handles nearly all the tasks it is designed for, even if its responses are often general when dealing with questions too complex to manage using entities or intents. Additionally, as the number of intents and entities increases, the complexity of the bot grows, and so does the likelihood of errors, as managing these aspects becomes increasingly challenging. Even with the use of synonyms and an expanded set of phrases for each intent, the bot often struggles to capture the meaning of all terms, even when the correct intent is defined. As a result, the bot may provide a vague response or ask the user to repeat their question. Nevertheless, the model, while not as sophisticated as GPT-4, effectively follows the flow of the conversation and provides appropriate responses. It successfully acts as a competition referee or a shopping center guardian, achieving its goal of assisting any kind of user with very short response times.

Chapter 5

Assessing Qualitative Examples and Quantitative Results

To evaluate the performance of the implemented ROS modules, live tests were conducted with Pepper to assess the robot's interaction capabilities. The tests were performed at a distance of approximately 1 meter, with slight background noise and one person interacting with Pepper. The conversation was carried out in English.

5.1 Rasa Test

The Rasa module, after an initial common draft, was gradually expanded by conducting tests on conversations that the robot could normally perform. This approach allowed us to identify possible shortcomings in terms of intents or stories, as well as actual errors, which mostly concerned the bot's difficulty in recognizing specific intents.

The conversations were conducted with a group member who did not work on the Rasa module to make them as realistic as possible and to add new potential interactions each time. A key part of the tests was understanding entity recognition, experimenting with new synonyms or more complex sentence structures.

Once checkpoints were reached (completion of the contest module, completion of the shopping mall module), the model was automatically tested using Rasa's TEST module, which allows the definition of test stories and an NLU test file.

A RASA TEST was conducted on 21 stories to evaluate the model's ability to recognize intents and entities.

It was observed that the chatbot recognizes most of the entities, meaning there are no entity mismatches.

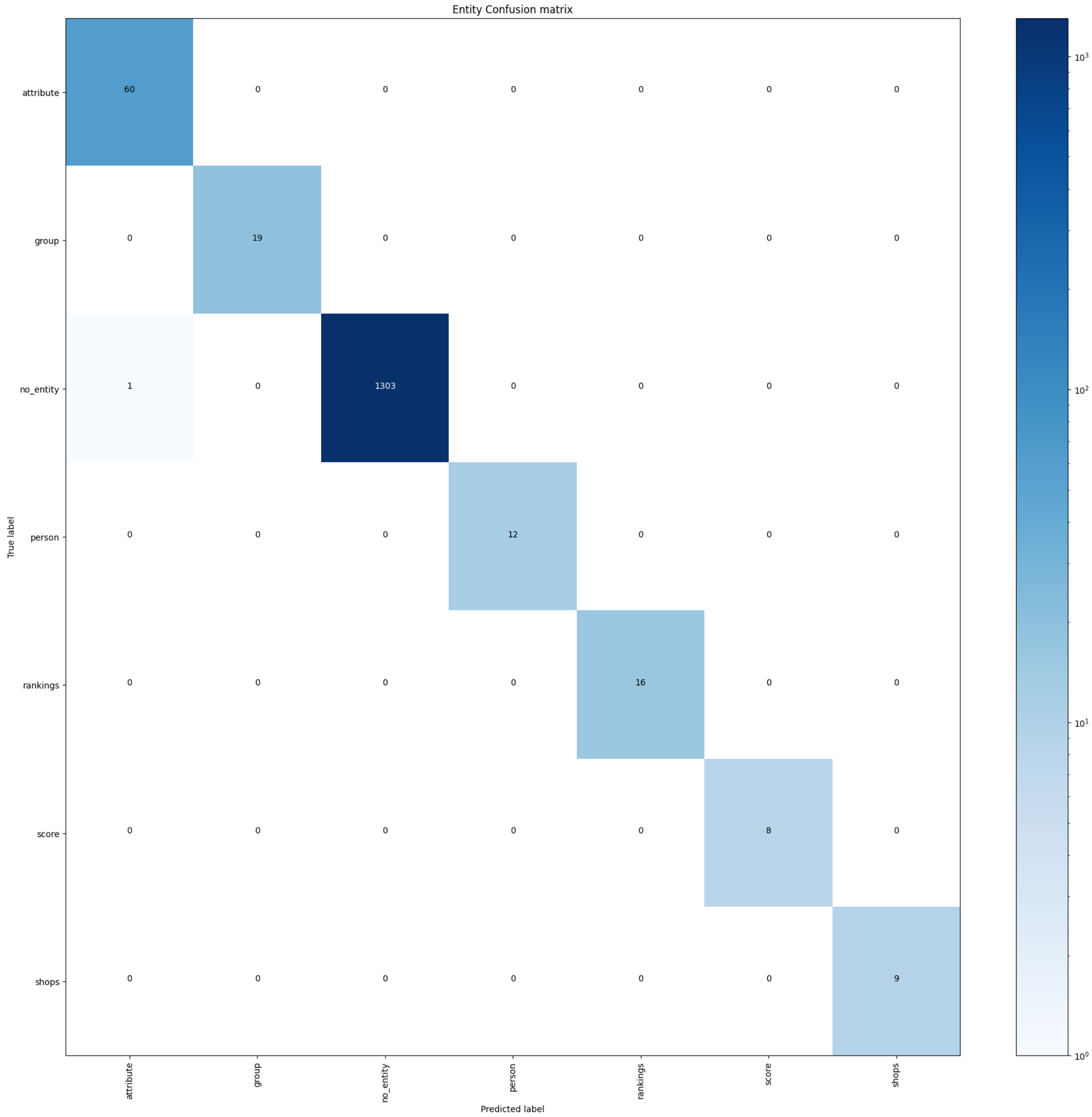

Figure 5.1: Final Confusion Entity Matrix

The same result is achieved for intents. The model is able to recognize all intents, with very few errors.

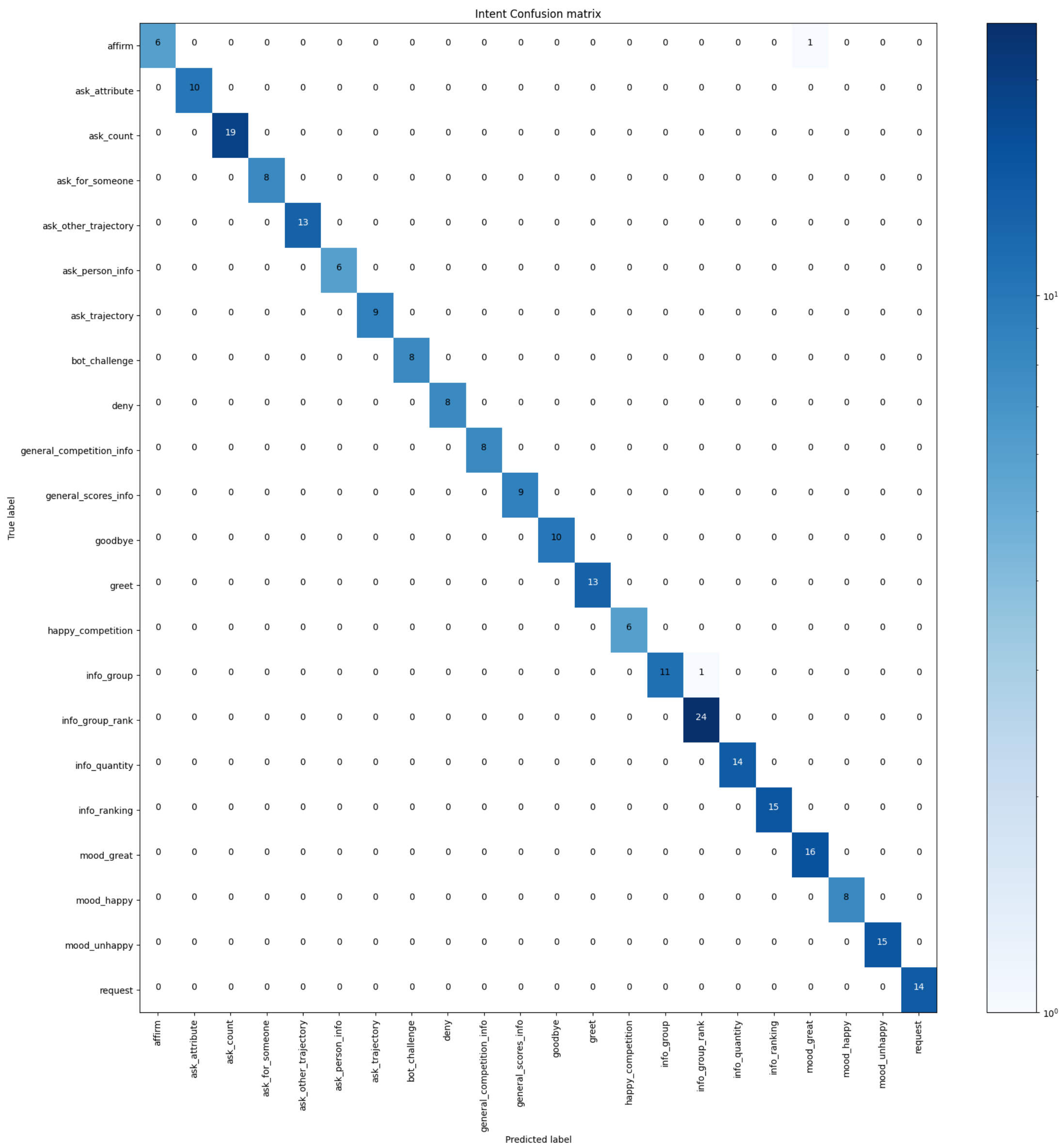

Figure 5.2: Final Confusion Intent Matrix

A final test was conducted on a laptop. Out of 25 questions, only 4 were answered incorrectly (this conversation can be found in the final chapter). The main issue lies in the confusion between intents regarding the number of people in the mall (answering with the total number of groups instead). It might have been better to create a specific entity instead of two different intents to minimize errors.

However, the results from practical tests with Pepper are slightly different: the speech recognition module often misinterprets words, and as a result, Rasa fails to recognize entities or intents. In the NLU file, some intents with grammatical errors were added, but such cases still occur.

5.2 General Test

- **Interaction Test**: Verified whether the robot accurately followed the face of the tracked person.
- **Listening Test for the Tracked Person** Assessed Pepper's ability to listen specifically to the individual being tracked.
- **Listening Test for Another Person**: Evaluated Pepper's response to a different person attempting to distract its focus from the currently tracked entity.
- **Arms movement test**: Checked whether Pepper moved its arms appropriately when responding to a person.
- **Head Reset**: Determined whether Pepper returned its head to the original position when no one was being tracked.

Chapter 6

Final evaluation and possible improvements

The final result is satisfactory: the robot performs the tasks it was designed for and even demonstrates empathy in some cases through its responses or arm movements.

Regarding empathy, an attempt was made to implement a module for emotion recognition using a face detector. Based on the detected emotion, a specific action would then be determined. However, this solution was not fully implemented as it was not sufficiently tested. The attempts made showed promising but unsatisfactory results, mainly because the detector often confused emotions. However, the module could be easily integrated as it is simply a subscriber node to the detection topic.

An additional area for improvement is the chatbot itself. While it gets most responses right, certain situations emerge during longer conversations where it fails to respond, as these cases were not considered in the Rasa module. It was considered to use a neural network and the Rasa module together to handle similar issues, such as grammatical errors in audio input, overly complex sentences that do not match any intent, and sentences with similar entities that the system fails to distinguish. However, the solution achieved with Rasa is satisfactory, as the robot can still hold a complete conversation with the user.

The robot does not engage a person proactively; it requires the person to speak first to initiate the conversation. When there are two people in front of the robot, it struggles to switch tracking between them. However, tests showed that in many cases, it successfully tracks the person speaking, thanks to audio recognition.

The response speed is not instantaneous due to the processing required by the audio module and the Rasa request. However, it is fast enough to allow a normal conversation.

The tests were conducted using an external microphone; however, once the module to avoid the feedback issue was implemented, there was no time to conduct consistent tests. In any case, the robot works even when using the built-in microphone (by specifying the appropriate index in the audio file). In case of malfunction, the original solution can be easily restored by setting the index of the external microphone.

6.1 Final bot test conversation

Your input: Hello
bot: Hey! How are you?
Your input: what can you do?
bot: I am Pepper, the guardian of the mall and the referee of this competition, ask me something about the ranking or a person in the mall to find.
Your input: what is the global ranking?
bot: The ranking is:
Position: 1, Group: 1, AFS: 0.895
Position: 2, Group: 6, AFS: 0.882
Position: 3, Group: 8, AFS: 0.862
Position: 4, Group: 5, AFS: 0.8
Position: 5, Group: 3, AFS: 0.787
Position: 6, Group: 10, AFS: 0.751
Position: 7, Group: 12, AFS: 0.729
Position: 8, Group: 7, AFS: 0.725
Position: 9, Group: 9, AFS: 0.667
Position: 10, Group: 14, AFS: 0.662
Position: 11, Group: 2, AFS: 0.594
Position: 12, Group: 13, AFS: 0.562
Position: 13, Group: 4, AFS: 0.513
Position: 14, Group: 15, AFS: 0.488
Your input: who are the members of the first group?
bot: The components of group 1, which classified first are: Sessa Antonio, Molinario Angelo, Martano Pietro, Ranauro Massimiliano.
Your input: and what is the afs of this group?
bot: The group:1 AFS is 0.895. An impressive result!
Your input: tell me about the competition
bot: The contest is based on building a detector for pedestrian attribute recognition task in a video, aiming to identify three different attributes: gender, hat, bag, and the trajectory, which refers to the number of lines crossed. Ask if you want more information about the groups or the ranking of competition.
Your input: what is the af score?
bot: The PFS is the Person F-Score, the LFS is the Line F-Score, the GFS is the Gender F-Score, the BFS is the Bag F-Score, the HFS is the Hat F-Score.
Your input: how many people are in the mall?
bot: There are 21 people in the mall matching your criteria Do you want information about a specific ID?
Your input: no, are there more people with an hat or with a bag?
bot: There are more bag, they are 7
Your input: i'm searching for people with an hat
bot: There are 3 people matching your criteria. And they are : [7, 10, 18], if you want information by one, specify the ID
Your input: Give me information about 18 **WRONG!!!**
bot: I didn't understand the ID. Can you please repeat it?
Your input: Give me information about person 18
bot: Information about the person with 18 id: { 'id': 18, 'gender': 'Male', 'bag': 'No', 'hat': 'Yes', 'trajectory': [2, 3] }. You can ask me the positions of this ID to know the

shops that he visited!
Your input: i am worried
bot: Maybe i can help you
Your input: i lost my friend
bot: It might be helpful to search in the mall. Please provide details such as gender, whether they are carrying a bag, wearing a hat, or both.
Your input: he is a man with an hat
bot: There are 3 people matching your criteria. And they are : [7, 10, 18], if you want information by one, specify the ID
Your input: can you give me the position? **WRONG!!!**
bot: The group 18 wasn't in the contest
Your input: give me the position of id 7
bot: Let me check in my database. He should have passed closed to store ['UNISA', 'COGNITIVE STORE', "PEPPER'S CAFE"] some minutes ago.
Your input: i've lost my hat
bot: It might be helpful to search in the mall. Please provide details such as gender, whether they are carrying a bag, wearing a hat, or both.
There are 3 people matching your criteria. And they are : [7, 10, 18], if you want information by one, specify the ID
Your input: how many people passed through UNISA **WRONG!!!**
bot: The groups are 14, with 3 or 4 people per group. Do you want know the global ranking or who is the winner?
Your input: how many people passed through PEPPER'S CAFE
bot: There are 12 people in the mall matching your criteria in PEPPER'S CAFE.Do you want information about a specific ID?
Your input: i'm so happy about the competition **WRONG!!!**
bot: The contest is based on building a detector for pedestrian attribute recognition task in a video, aiming to identify three different attributes: gender, hat, bag, and the trajectory, which refers to the number of lines crossed. Ask if you want more information about the groups or the ranking of competition.
Your input: i'm so happy
bot: Great! It is about the competition? Tell me why if you want
Your input: yes, we arrived fourth at the contest
bot: Congratulation! Let me remember your interesting project!
The group 5 ended in position 4. This is a summary for you. The components of group 5 are: Del Gaudio Nunzio, De Angelis Christian Salvatore, Aliberti Carmine, Casella Giovanni. the AFS is: 0.8
Your input: yes thats me, thanks
Your input: bye
bot: It was a pleasure, see you!