FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Leveraging Context-awareness in IoT Systems: An Approach with Voice Assistants

Daniel da Silva Gonçalves

Mestrado em Engenharia Informática e Computação

Supervisor: João Pedro Dias

Second Supervisor: André Restivo

July 27, 2023

Leveraging Context-awareness in IoT Systems: An Approach with Voice Assistants

Daniel da Silva Gonçalves

Mestrado em Engenharia Informática e Computação

July 27, 2023

Abstract

The Internet of Things (IoT) represents the network of physical devices – "things" – equipped with sensors and actuators, with low-computing power that allows one to communicate and share data with other devices over many types of networks, including the Internet. Lately, the increasing capacity to link everyday items has evolved to a degree where seamless communication and monitoring of our homes and gadgets is becoming more widespread, vitalizing smart spaces.

Several companies have been advertising their smart assistants, which aim to make these gadgets more user-friendly by streamlining human interaction inside smart spaces. These same technologies already exist and may be found all around us, such as Cortana, Siri, and Google Assistant. The last two, for example, allow the user to define their name, recognize voice input, and detect if the user is talking to them based on particular summoning instructions such as 'Hey, Siri!'.

However, despite being handy for everyday tasks, they are not meant to be wholly integrated into smart environments, *i.e.*, interact with IoT devices. The reason lies in their restricted support for limited and straightforward instructions and their inability to identify and retain the context of the interaction. Siri, for example, will only be able to link the context of two separate requests if the assistant asks to elaborate.

Existing literature presents several approaches to improve smart assistants, *i.e.*, one capable of discerning the context of natural commands and acting on them. Nonetheless, these works also present several limitations and pending issues, namely the handling of context resolution, user behavior analysis, and the development of self-correction machine-learning algorithms.

As such, we developed this work with the aim of addressing the challenges present in current IoT ecosystems by enhancing contextual awareness and studying its impact on user experience. The resulting system we have developed, allows users to interact with a simulated smartspace and a conversational assistant through an interface. Users, regardless of their level of expertise, can modify the environment, establish rules and preferences, and observe how the assistant responds to their requests.

The thesis successfully demonstrates the feasibility of developing a conversational assistant that understands the context of IoT queries. It encompasses a wide range of interactions, including resolving contextual ambiguity, utilizing user preferences, and detecting the user's position when specific information is missing.

To assess the system's effectiveness, a group of 10 users, including both experts and newcomers in the IoT and the field of technology, were selected to test our system. These users were asked to evaluate the system's usability, effectiveness, and their likelihood of using it compared to existing technologies. Additionally, the system's feasibility was compared to an existing dataset of requirements and practical query samples, as well as with the performance of WIT.ai.

Keywords: Contextual Awareness, IoT, Internet of Things, Voice Assistants, Virtual Assistant, Voice-Controlled Assistant, Automation

Resumo

Internet of Things (IoT) representa a rede de dispositivos físicos – "*things*" – com sensores e atuadores, e baixo poder computacional que permitem a comunicação e a partilha de dados com outros dispositivos pela Internet. Ultimamente, a crescente capacidade de unir dispositivos do dia a dia evoluiu para um ponto em que a comunicação e a monitorização contínua das nossas habitações estão a tornar-se mais difundidas, vitalizando os nossos *smart spaces*.

Várias empresas têm vindo a anunciar os seus avanços com *smart assistants*, que visam tornar esses gadgets mais fáceis de usar, simplificando a interação humana dentro de espaços inteligentes. Essas mesmas tecnologias já existem e podem ser encontradas ao nosso redor, como Cortana, Siri e Google Assistant. Os dois últimos, por exemplo, permitem que o usuário defina seu nome, reconheça a entrada de voz e detecte se o usuário está a falar com estes mesmos com base em instruções de convocação específicas, como 'Olá, Siri!'.

No entanto, apesar de serem úteis para tarefas diárias, não conseguem ser totalmente integrados em ambientes inteligentes. A razão está no suporte restrito a instruções limitadas e diretas, e na incapacidade de identificar e reter o contexto da interação. A Siri, por exemplo, só consegue unit o contexto de dois pedidos distintos se ela pedir para elaborar.

A literatura existente apresenta várias abordagens para melhorar os *smart assistants*. No entanto, estes trabalhos também apresentam várias limitações e questões pendentes, nomeadamente o tratamento da resolução de contexto, a análise do comportamento do utilizador e o desenvolvimento de algoritmos de aprendizagem automática de auto-correção.

Como tal, desenvolvemos este trabalho com o objetivo de enfrentar os desafios presentes nos ecossistemas atuais da Internet das Coisas (IoT) ao melhorar a contextual awareness e ao estudar o impacto na experiência do usuário. O sistema final permite que os utilizadores interajam com uma simulação de um smart-space e um assistente através de uma interface. Os utilizadores, independentemente de seu nível de experiência, podem modificar o ambiente, estabelecer regras e preferências e observar como o assistente responde aos seus pedidos.

A tese demonstra com sucesso a viabilidade de desenvolver um assistente de conversação que compreenda o contexto de consultas na IoT. Ela engloba uma ampla gama de interações, incluindo a resolução de ambiguidades contextuais, a utilização de preferências do usuário e a detecção da posição do usuário quando informações específicas estão ausentes.

Para avaliar a eficácia do sistema, um grupo de 10 utilizadores, com diferentes níveis de experiência na área de IoT e técnologia, foram selecionados para testar o sistema. A esses utilizadores foi-lhes pedido para avaliar a usabilidade, a eficácia e a probabilidade de uso do sistema em comparação com as tecnologias existentes. Além disso, a viabilidade foi comparada com um conjunto de dados existente de requisitos e exemplos práticos, bem com o desempenho do WIT.ai.

Keywords: Contextual Awareness, IoT, Internet of Things, Voice Assistants, Virtual Assistant, Voice-Controlled Assistant, Automation

Acknowledgements

I would like to extend my heartfelt gratitude to the individuals who have supported and guided me throughout the completion of this thesis.

First and foremost, I am deeply thankful to my thesis supervisor, João Pedro Dias. Your expertise, patience, and continuous support have been invaluable throughout this research journey. Your insightful feedback, constructive criticism, and unwavering commitment to excellence have greatly shaped the direction and quality of this thesis.

I would also like to express my appreciation to my co-supervisor, André Restivo, for his valuable contributions and guidance. Your extensive knowledge, attention to detail, and commitment to academic rigor have been instrumental in enhancing the depth and breadth of this research. Your willingness to challenge my ideas and encourage creative thinking has been truly inspiring.

To my parents, whose unconditional love and unwavering belief in me have been a constant source of strength and motivation, I am forever grateful. Your endless support, encouragement, and sacrifices have made it possible for me to pursue my academic goals. I cannot thank you enough for the countless sacrifices you have made and the countless hours you have dedicated to nurturing my intellectual growth.

I would also like to acknowledge the friendship and support of Sérgio Dias and Diogo Vicente. Your presence and unwavering support during the ups and downs of this thesis journey have been truly invaluable. Your encouraging words, insightful discussions, and short breaks gave me a sense of balance and perspective.

Lastly, I would like to extend my gratitude to all the individuals who have played a part in my academic journey, directly or indirectly. To my professors, colleagues, and friends who have provided advice, shared their knowledge, or simply offered a listening ear, thank you for your contributions and for being a part of my growth as a student and as a professional.

Daniel Gonçalves

"The only way to do great work is to love what you do."

Steven Paul Jobs

Contents

List of Figures

Abbreviations e Symbols

IoT	Internet of Things
VA	Voice Assistant
API	Application Programming Interface
Py	Python (Programming Language)
SA	Smart Assistant
AI	Artificial Intelligence
WWW	*World Wide Web*
SLR	Systematic Literature Review
SUS	System Usability Scale
NLP	Natural Language Processing
STT	Speech to Text
ML	Machine Learning

Chapter 1

Introduction

Contents

In this chapter, we lay out the background of the fields focused on our research and the objectives this thesis aims to accomplish. Both the context of this work and some example scenarios that drive the motivation for this effort are presented in Section 1.1 and Section 1.2, respectively. We discuss the problems of existing solutions in Section 1.3 and list our goals in Section 1.4. Finally, the structure of this paper is laid out in Section 1.5.

1.1 Context

The Internet of Things (IoT) ecosystem is defined as the connectivity of devices over the Internet. Standalone devices with Internet connectivity that allow themselves to be monitored or controlled remotely can be considered an IoT device. Due to the wide availability of devices with varying processing and communication capabilities, the IoT represents a highly heterogeneous environment. In turn, this paves the way for the development of smart spaces across various fields, from innovative agriculture to the reshaping of hospitality and smarter surroundings that include whole cities. The heterogeneity of data provided by so many devices is the glue that holds disparate sources together to enable coherent communication and inference from the distinct sensor and visual data. T. Lu and W. Neng. [37] define IoT in this way:

> "Things have identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts." [37]

In addition to that, and the context of communication between devices and systems, D. Buhalis and R. Leung. [11] define another relevant piece of the system (interoperability) in the following statement:

> "Interoperability is the key requirement in smart hospitality, as disparate systems can interconnect and exchange information, among public and private organisations." [11]

In 2020, the global Internet of Things industry was valued at 182 billion US dollars, and by 2030, that number is expected to triple, with the correct figure being around 621 billion in US currency. Parallel to this, the number of linked devices is also predicted to triple, reaching 29 billion IoT devices in 2030 from the 9.7 billion registered in 2020. Such statistics lead to an approximation of 15 connected devices per person, representing the increasing IoT field trend. Figure 1.1 graphically represents the growth of active IoT connections. Nevertheless, despite the sheer number of IoT devices, the complexity of these systems is also on the rise. Proper operation of the network depends on a plethora of disparate communications and devices interacting seamlessly with one another.

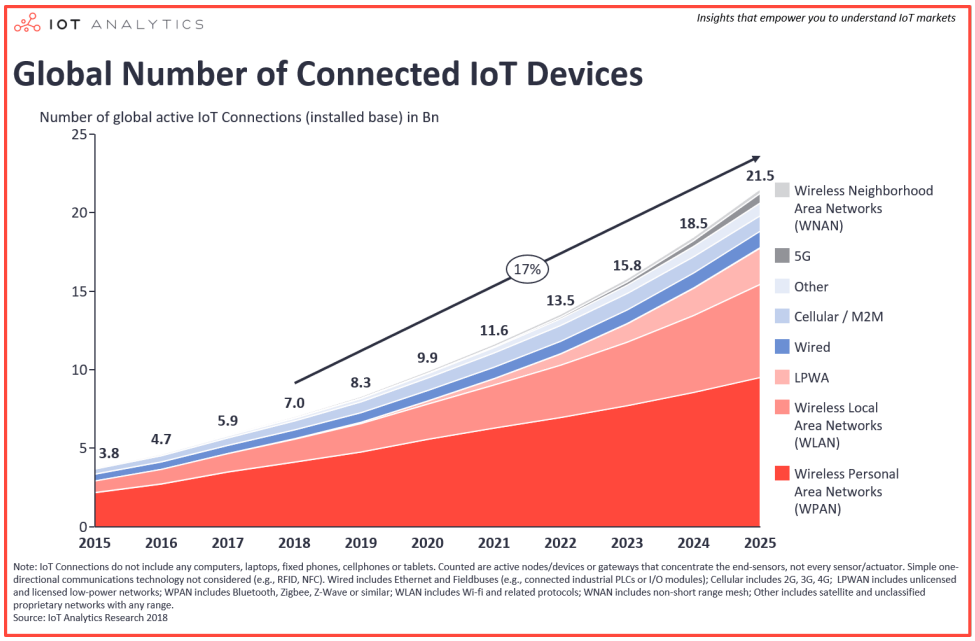

Figure 1.1: Growth of active IoT connections [38]

Within the same vein as the IoT, the usage of voice assistants (VAs) is on the rise, with a result of 4.2 billion voice assistants in use throughout the globe in 2021 alone [57]. According to the Adobe research cited in the same page [3], 94% of the 1000 users polled found speech technology efficient and beneficial in their daily lives. Nonetheless, this encouraging discovery has a downside: around half of them found speech technology confusing and challenging.

The distinction between Voice Assistants and the Internet of Things is becoming more challenging to recognize as the two spheres evolve. Throughout 2021, the European Commission

worked on a report that would help it again a better understanding of the consumer IoT Sector [13]. On point 9, the commission recognizes that there is a growing trend towards using voice assistants as a user interface that enables interaction with smart devices and consumer IoT services. Additionally, on point 17, the commission states that the full potential of the IoT ecosystem can only be realized with interoperability across smart devices, voice assistants, and consumer IoT services.

Contextual awareness is also relevant to Voice Assistants and Internet of Things. Understanding the context of user queries and using the sensors intelligently to predict the context of the query is pivotal towards a more cohesive and seamless user experience within a smart space environment. Raoul Wijgergangs, EnOcean's CEO, in an article [64] states that data is required for the systems in the smart home if they are to achieve the goals of becoming context-aware and more intelligent. A smart assistant (SA) or Artificial Intelligence (AI) platform might then utilize this data to adjust the home's environment based on factors like the temperature or the status of connected equipment.

1.2 Motivation

To introduce the context that was discussed earlier to our case in point, this study outlines two hypothetical scenarios in how VAs equipped with contextual awareness and the IoT work together to create an environment that is simple and enjoyable for the end user. These hypothetical visualizations will also play an important role throughout this whole work as the primary illustration of our goals.

First Scenario: As a variation of the *Execution* requirement mentioned by Perera *et al.* [12], we picture a user driving home from his office. During their travel, they connect with their VA at home and ask it to prepare everything for when they get home. In this case, the VA should relay the information to the IoT space for the user's house to be at a pleasant temperature when they arrive, the coffee machine to be ready, and the water heater to be ready. However, let us assume we want the user to be more particular and establish that a few stages are performed throughout this usually automatic procedure. In this specific situation, the VA would need to contextualize the request depending on the user's distance to their house as well as what is more convenient or probable. There is also the possibility of more complex occurrences, such as the following complication: Suppose the user exceptionally requests the VA to prepare their coffee machine, but three of the same type are in the residence. To which one should the VA transmit this request? Naturally, we may concentrate on the device that the user uses the most, yet, he could choose the one that is closest to their house's front door. Before making a judgment about this matter, the VA should probably request more feedback. However, this may not be necessary if two coffee machines are empty or haven't been used in a very long time. As such, we want the VA to inspect the state of the IoT space (and its history) to make a contextualized decision or, at the very least, the most convenient one.

Second Scenario: Let us assume a user who's using an interface that allows them to visualize an entire smart space. In it, the user can click on any position inside the house and ask the VA through the text box or by speaking, to take any specific action. If, by arbitration, the user decided to ask the VA to "Turn on the television"", the VA would be missing crucial information. For example, the VA doesn't know what television the user is referring to. Under such circumstances, the most reasonable assumption for the VA would be to infer that the user is referring to the television in close proximity to their current position. Nonetheless, the execution of this seemingly simple directive necessitates a series of preliminary tasks, including the examination of sensory data to detect the exact position of the user. Another more complex request (query) could arise if the user told the VA to turn on two televisions. Assuming there are more than two televisions in the house, and without further input, the VA could first prioritize the closest television to the user and then select the other television based on the usual steps taken by the user at around that time of the day. This could be done by studying the data stored by the IoT system. Alternatively, the VA might choose to prioritize the televisions in closest proximity to the user.

While we recognize that the preceding scenarios are hypothetical and that the work to be presented in the following sections does not aim to cover every possible situation and subtlety, we decided to use the second scenario as a solid foundation for our research efforts, taking advantage of the benefits provided by the simulation interface. This approach provided evidence that was instrumental in order to better manage our time, as it kept us from needing to deploy the entire system within a real smart environment. In Chapter 5, we provide a more detailed explanation of our technique through an in-depth discussion of the systemic application of this approach.

1.3 Problem

Many different voice-activated assistants are now available. Whether or not they are directly applicable to the present sectors of IoT, we have enough information to classify them into distinct subsets that are optimal for their intended purposes and levels of implementation. Figure 1.2 illustrates an example of a few existing solutions, broken into categories.

General Purpose Voice Assistants are best suited for everyday tasks like making phone calls, sending texts while we're not near our devices, and other simplistic tasks. The biggest downside lies on the low specification for IoT spaces. On the contrary, Domain Specific Standalone Assistants and In-App Voice Assistants are more restricted to specific applications, environments and not specifically targeted towards a general-case scenario which would be the aim of a smart home assistant.

The best current alternatives are Smart Speakers, which are targeted towards smart spaces and general-case use, but existing solutions are too reliant on propriety services, generate privacy concerns [58], and most of the time is imprecise [44]. The biggest loss comes from their extremely simplistic behavior and lack of contextual awareness. Contextual awareness, as seen in

Section 1.2, is essential for any smart space assistant because it cuts down on the amount of back-and-forth communication between the end user and the IoT service, making the experience more streamlined, natural, and pleasant. Currently, voice assistants like Google Home, Alexa, Sonos Speaker fail to achieve this while posing many other concerns from regulators.

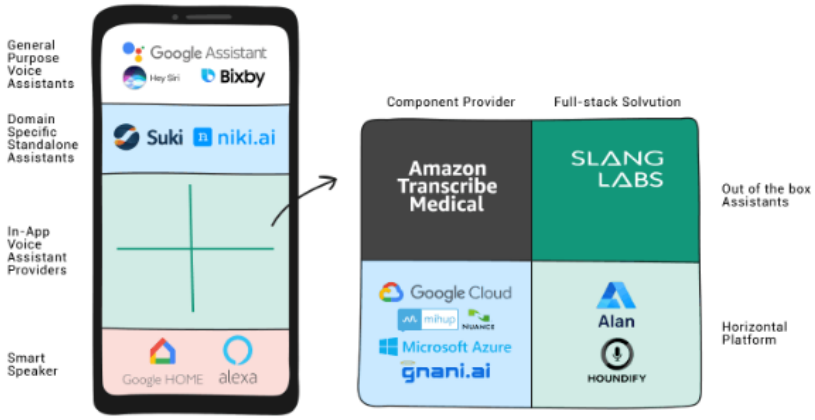

Figure 1.2: Types of Voice Assistants [33]

1.4 Goals

The work presented here aims to prove and demonstrate that a functional voice assistant can be developed; that is, one that can effectively control and manage a smart-space and deal with questions from users of varying complexity levels while being contextually aware.

We develop the scenarios from Section 1.2 as a starting point for our development and as a guide for how the voice assistant should operate. Understanding of the IoT environment, contextual study based on sensor data and user position, and application of rules and preferences are the primary focuses of this research; as such, in-depth analysis of rule collision and other occurrences of a similar kind are not within the scope of this study.

With these foundations in mind, our voice assistant has to be able to (1) understand the context of a user inquiry, (2) identify the action that constitutes the query in question, and (3) interpret which IoT devices are relevant for the request based on patterns of behavior, preferences and active sensor data.

1.5 Structure of this Document

The structure of this thesis is broken down into seven chapters. In the present chapter, we have introduced the work's concepts and the goals it aims to accomplish.

In chapter 2 the required context for this work and the notions that we believe to be important to comprehend most of the work that was carried out are presented. These concepts are relevant for studying the state of the art and contextualizing the work done.

Chapter 3 describes the research done on existing literature, ranging from research questions to implementing a Systematic Literature Review (SLR) whose results are later categorized and detailed. A conclusion is extracted at the end, describing the issues and improvements that could be made.

The work and its concepts are defined in Chapter 4. In it, we use the information from Chapter 3 to describe the issues with existing solutions and define the hypothesis of this work, along with the methodology, the goals and aims for the research questions we plan to answer.

We elaborate on the implementation of our work in Chapter 5, discussing the implementations we have made, the architecture of the system, and assumptions that we made to simplify our development process.

After that, we evaluate the system's feasibility and usability in Chapter 6, where we use a group of 10 users to test our system and compare our system with an existing dataset of requirements and query samples. We make a brief conclusion of these results, which are then mimicked in Chapter 7 where we also introduce challenges faced and future work.

Chapter 2

Background

Contents

This section provides background information on the primary topics that will be covered throughout this work.

2.1 Internet of Things

The Internet of Things (IoT) term was coined in 1999 by Kevin Ashton. It was used describe a complete system where the physical word is connected through the Internet, while also including radio-frequency identification (RFID). Ever since that time, the field has grown exponentially, accommodating more technologies and interacting with different fields to embrace a more comfortable and diverse ecosystem. According to Keyur K Patel *et al*. [48], they define IoT in the following sentence:

> "Internet of Things (IoT) is a concept and a paradigm that considers pervasive presence in the environment of a variety of things/objects that through wireless and wired connections and unique addressing schemes are able to interact with each other and cooperate with other things/objects to create new applications/services and reach common goals." [48]

We embrace these "things/objects" as IoT Devices. In the mentioned research article [48], the authors also consider the IoT field as an internet that can be broken into three aspects, namely: (1) People to people, (2) People to things, and (3) Machine to Machine, Interacting through the Internet.

Aspect (1) relates to the usage of the IoT field to communicate with other people. Living Labs [39] are an example of this, where it integrates research and innovation processes in order to co-create innovative and technological products. This is done by sharing knowledge between stakeholders, which leads to an internet of people to people. This is usually the higher layer, and aimed at the needs of the users when they engage with an IoT ecosystem.

Aspect (2) is related to the interaction between the end user and the IoT devices. Voice assistants and monitoring dashboards such as Grafana [41] are two examples of user interfaces that act as a link between these two entities, displaying the history of the data handed out by each device.

Aspect (3) represents the ecosystem itself and the communication that occurs between devices in order to share date, complementing certain weak spots offered by other sensors or even diagnosing possible issues with an intelligent tool responsible for monitoring the data.

The Scenarios listed in Section 1.2 make use of the types listed in Point 2. and Point 3. where the main goal is for the IoT ecosystem to interact with the user through a user interface known as the Voice Assistant while also executing Machine to Machine interactions based on the study of data to make inferences. These three pillars provide the ecosystem's fundamental core, enabling IoT to branch out into a wide variety of areas, such as intelligent automation in factories, airports, hotels, cities, and transportation. These applications are represented in Figure 2.1 and can communicate both ways, thus allowing devices from one side of the globe to interact and react to data changes on the opposite side.

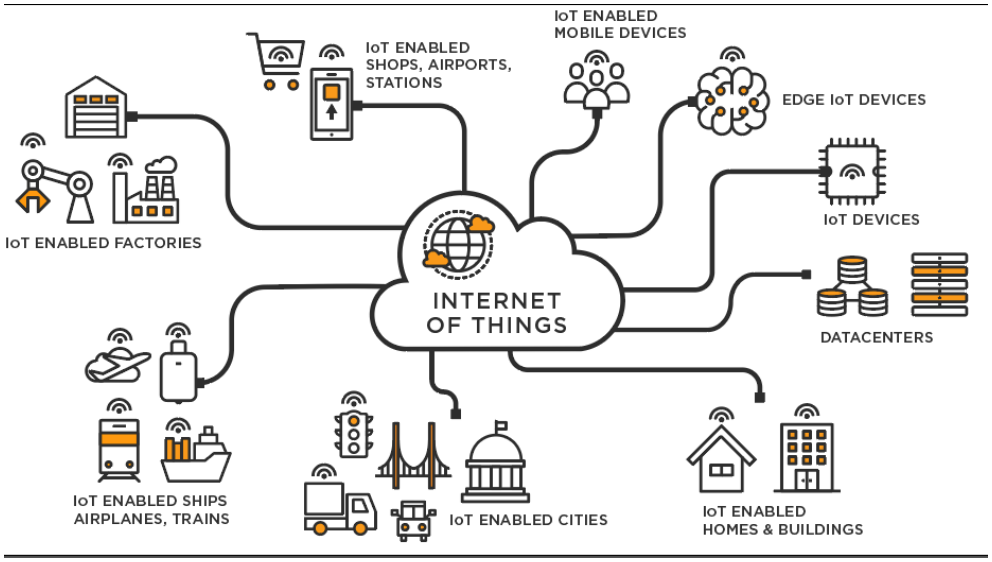

Figure 2.1: Applications of Internet of Things [61]

2.2 Voice Assistants

Voice Assistants trace back to the beginning of voice recognition technology that was installed in personal computers back in 1990s [29]. Ever since that age, improvements have been made in this field, with Siri being the first modern digital voice assistant that was installed on a smartphone.

A voice assistant works through voice recognition and machine learning to understand user input. Voice recognition is done through pattern recognition that is based on analog signals that are first converted to digital signals and then matched against a pre-existing database.

There already exist many technologies to sustain this type of technology, ranging from speech to text libraries, to dedicated libraries like DialogFlow which already incorporate natural language processing capabilities that facilitate the creation of conversational user interfaces. Figure 2.2 lists a representation of Dialogflow's workings. The input given by the user is processed and sent to Dialog, which is then fulfilled based on specific API calls and services.

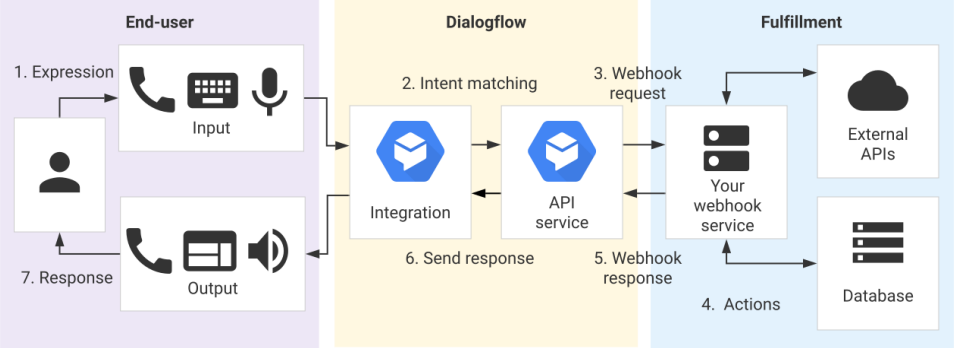

Figure 2.2: Representation of Dialogflow's pipeline [28]

The benefit of voice assistants is that they can handle instructions quickly and efficiently. Voice-based techniques are often quicker than text-based alternatives, although they are less accurate. As a consequence, despite their convenience and accessibility, they fail to provide more exact results, which may lead to misunderstandings or incorrect instructions being handled. To protect against this, several voice assistants include confirmation prompts, but this frequently leads to tasks being delayed.

2.3 End-User Programming

End-user programming (EUP) or End-user development (EUD) is a term that encompasses the tools which allow users who do not possess particular skills or experience in programming to write/assign instructions and executions in computers. As noted by Dias *et al.* [20, 19], there exists a vast amount of tools available in order to help users handling the increasingly complex ecosystem. As more devices and sensors are added, it becomes necessary to find a way to easily handle the final result in a fast and accurate manner. Voice Assistants, as noted by *et al.* Austerjost [8] are an incredibly useful alternative for home automation, next to visual programming (VP) methods.

As noted in Section 2.2, p. 8, Voice Assistants are very quick and efficient at handling tasks but existing solutions like Siri, Alexa, Cortana and others do not focus on IoT-features, making their implementation in smart spaces more challenging [20, 19]. In addition to this it is difficult

for end-users to follow the topic of the conversation if the interaction between the user and the VA lasts too long (shown by the work done by Lago *et al.* in Section 3.2.1, p. 18), introducing additional challenges to this approach.

A way to avoid these problems is through Visual Programming (VP), which encompasses the act of programming using a visual programming language (VPL). VPL contemplates any programming language that allows us to create programs and define sequences of instructions graphically rather than textually [32, 15, 50]. This strategy is common and widely used in circumstances when coding is particularly difficult or where it might be very difficult to debug. One prominent example may be seen in the programming of shaders for 3D engines. In this kind of programming, instructions are divided up into distinct blocks in order to limit the likelihood of errors occurring.

The choice for VP comes from its simplistic design, abstracting the need to learn programming languages for the common user. Such abstraction allows the user to easily understand the entirety of their program in a more direct and straightforward way, and makes the development of software much easier and quicker, with a small overhead to learn the tool in question. In fact, it would seem that approaches for program visualization are more efficient when used to brand new situations for which the programmer has no prior experience [45]. Some disadvantages, however, have been noted, such as limited platform scalability, limited potential and how the execution of complex programming functions is a tedious process. [7].

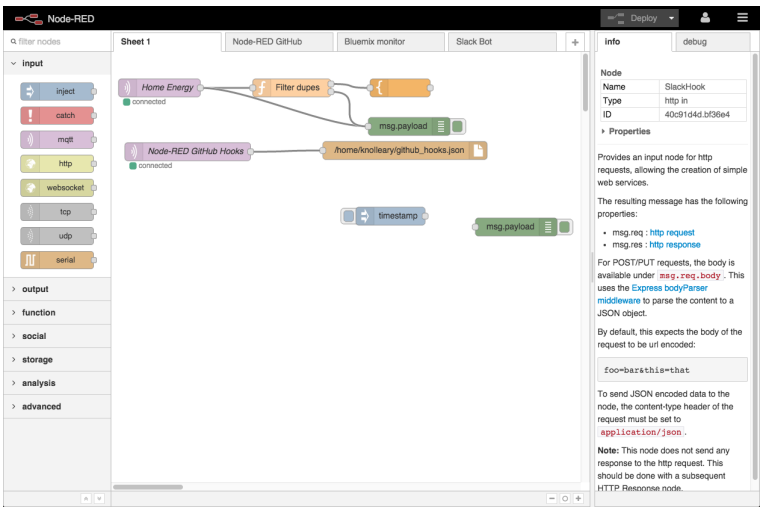

Figure 2.3: Example of Node-Red [46]

Node-RED [17, 62, 18], is a visual tool (check Figure 2.3 for an example) based on the flow-based programming paradigm and was built on NodeJS which runs locally on the browser. The programming tool is open-source and is part of the JS foundation, although it was initially developed by IBM. The development of applications involves the use of nodes and wires, which, when combined, form a flow. A node is generally understood to be the most fundamental component of a flow; moreover, nodes are able to communicate with one another via the use of wires that are connected to the ports on each of their endpoints. Nevertheless, nodes may only be activated in very specific circumstances, such as when they receive information from another node that is

linked to them, when they are triggered by a timer, or by an incoming HTTP request. In addition, a node is only permitted to have a single input port, although the number of output ports it might have is unrestricted. The user is able to introduce their own Javascript code in addition to the predefined set of nodes, which enables further customization of the tool's internal operations.

Because it can be used to a wide variety of use cases, Node-RED enables rapid prototyping and may be of great assistance to users who have only a basic understanding of programming. It also provides users with a high degree of flexibility, while being a "little code" solution.

2.4 Contextual Awareness

Contextual awareness is defined as the capacity to use *context* in order to understand and characterize a situation or entity as to make a *contextualized* inference [2].

The word *context* refers to any amount of *information* that allows us to characterize people, entities and the environment around us. Within the framework of this work, context-awareness in Voice Assistants in Internet of Things (IoT) consists of using sensor data and the knowledge (*context*) about the smart space in order to predict and better serve the user without the request of more input.

A more formalized definition of contextual awareness in the field of computer science is:

> *"Context-aware systems are concerned with the acquisition of context (e.g. using sensors to perceive a situation), the abstraction and understanding of context (e.g. matching a perceived sensory stimulus to a context), and application behaviour based on the recognized context (e.g. triggering actions based on context)."* — Archived from [54] [53].

The most common example of contextual data is *localization* [53]. Two possible exemplifications were introduced in the motivation of this work in Section 1.2, p. 3, where the location of the user is used in order to predict which devices the user is referring to or even predicting future events, allowing the system to better serve the user in the future. Context awareness is also relevant for *Activity recognition*, where researchers believe that system monitoring of a user's activity can allow the system to eventually act on the user's behalf and simplify their lifestyle [63].

Besides location, other forms of contextualization exist, such as the current time of day, weather forecast stations and identity of the user. These individual components can also be merged in order to increase the accuracy of the prediction and identification of clusters. For example, if a user wakes up at 8 AM every day, and opts to stay in bed every time it's raining outside, then when he asks the system: *"Turn on the TV."* the most logical decision would be to select the TV in the bedroom rather than the ones in other rooms. A possibly different decision would be made if it happened to be sunny. With this, location, time and weather are three variables being used as contextual data in order to better analyse and understand the situation. The method used to convey this awareness for this work lies the entity prediction implied by the initially vague context of the query and its user's surrounding context.

2.5 Summary

In this Chapter we have discussed a variety of factors that are essential to understand the composition of our work. In Chapter 2.1 we defined the concept of Internet of Things. In Chapter 2.2 we talked about Visual Assistants and discussed how they are made while introducing Dialogflow as an essential component for the introduction of a VA.

In Chapter 2.3 we covered End-User Programming based on Voice Assistants and Visual Interfaces. We covered Node-RED superficially and offered an overview of the tool's functionalities. Finally, in Chapter 2.4 we discuss the concept of context-awareness and relate it to the topic of IoT, introducing a brief example on how context-awareness can be employed in a smart space environment.

Chapter 3

State of the Art

Contents

This chapter provides a comprehensive analysis of the current State of the Art in regards to voice assistants and context-awareness in the field of IoT. Chapter 3.1 refers to the methodology used, where the research questions, search queries and filtering criteria are laid out.

3.1 Methodology

In order to retrieve papers and projects that are relevant to the topics and areas related to this work, we performed a *Systematic Literature Review* (SLR). In it the work and conclusions of other writers are investigated in relation to predetermined research questions. It is a search that is exhaustive and carried out across a variety of databases and grey literature, and one where other people have the ability to recreate. As a result, the first thing that we do in this multi-stage procedure is establish the research questions whose relevant literature we wish to focus on. The next step in the process involves the formulation of specialized search queries based on relevant keywords and concepts to our work. These queries are then executed in a small collection of academic research databases, where they are then matched to thousands of papers. The returned papers, which may be of a variety of formats, are subjected to a manual filtering process based on the contents of their introduction, abstract, and conclusion sections. In addition to these results, we may include grey literature in our work. Grey literature refers to a wide variety of content that is available outside of conventional publication and distribution models, and it is not easily accessible through the results of search engines.

3.1.1 Research Questions

As was previously indicated, SLR relies on a set of concise and answerable foundational Survey Research Questions (SRQ). These constitute the basis of our research and help reduce selection bias. Each study selection criterion is based on a research question, which specifies the kind of documents or papers can best answer that issue. In most cases, these questions help us define the eligibility or inclusion requirements for the papers relevant to our work.

SRQ1 *What solutions exist for voice assistants?* The first question has a very wide scope, enabling us to establish potential solutions and narrowing down our search to a more manageable subset of all possible documents in the database. The question does not take into consideration whether or not the findings are connected to smart spaces that are powered by the Internet of Things. In a broader sense, the purpose of this initial question is to provide an overview of the potential solutions rather than accurate and precise ones targeted at our problem at hand.

SRQ2 *Based on the results of **SRQ1**, which of those are relevant to IoT powered smart spaces?* This question narrows our search area to publications that are relevant to Internet of Things and smart spaces, hence reducing the size of the subset that was originally acquired. This allows us to exclude findings that would otherwise concentrate on voice assistants for general use, such as support assistants, chat bots, and other instances that do not fall within the purview of our study.

SRQ3 *Based on the results of **SRQ2**, which of those involve context awareness in decision-making?* This question further narrows our search down, to a more specific group of results that better serves the main goal of our research. After processing, we are left with papers that include Voice Assistants as a User Interface with IoT-powered smart environments and whose primary emphasis lies on context awareness. These findings are essential to our work and come quite near to accurately representing the solution that we want to achieve. It also serves to assert which solutions are closer to our main hypothesis.

3.1.2 Databases

This systematic literature review (SLR) for our State of the Art study was applied to three distinct databases: (1) Scopus, (2) ACM Digital Library and finally (3) IEEE Xplore. The findings from each of these several sources were combined, and any duplicate findings were omitted from the merged results. This work settles on these three databases due to the reliability of the articles they include as well as the breadth of the ground they cover in terms of academic literature about information technology. According to what is shown in Section 3.1.3, the queries that are used will be matched against the metadata of each document, namely the title, the abstract and the keywords. In addition to these findings, this work included a few more papers that were investigated in a separate part of the report devoted to grey literature. These documents assisted this work in locating pertinent materials that may not be included in official academic research databases.

3.1.3 Search Queries

To extract the relevant papers for this work, a search query was formulated and used on the database systems referenced in Subsection 3.1.2. This query was made in line with the research questions mentioned in Subsection 3.1.1 and adjusted over repeated iterations. As such, in order to obtain relevant results, the query was initially made with combinations of the following keywords: IoT, Internet-of-Things, Voice Assistants, Assistant, Context Awareness, Smart Space. Multiple iterations were carried out in the process taking into consideration already known papers that were certain to contribute to this work positively and help us improving the quality of the findings. The query is made of disjunction of possible terms in order to broaden the scope of our search by attempting to cover the greatest number of cases while simultaneously trying to match these results with our initial research questions.Given these statements, we have the following query, which was applied on all three previously mentioned databases.

```
("iot" OR "Internet-of-Things") AND ("End-User Development" OR
"Smart home" OR "Home Automation") AND ("context awareness" OR
"decision-making" OR "conversational assistants" OR "chatbot")
```

The query is applied to the entire metadata of every publication (Title, Keywords and abstract) and aims to filter any work that is related to the Internet of Things when Voice Assistants are involved. The query introduces terms such as "natural language processing", "artificial intelligence" and "machine learning" in order to cover a wider scope of possibilities. Overall, the query is far from restrictive, offering many alternatives for each term and it helps us get a foundation of articles that may employ unexpected phrases to describe a study that is pertinent to this thesis.

Another discovery that prompted this level of openness and diversity was the realization that contextual awareness is notably rare when paired with the Internet of Things and Voice Assistants, making it difficult for us to identify relevant content. Using searches with more specificity would result in just three items being returned. Consequently, the query was expanded to include other phrases like "End-User Development" since many relevant articles made use of concepts that aren't immediately obvious. Additional pertinent domains in which voice assistants are employed and given as alternatives were also taken into consideration, which enabled us to cover other keywords that may still be important to our work despite the fact that they are not organically tied to the premise that our study is based on.

3.1.4 Result Filtering

The results obtained from the queries listed in Subsection 3.1.3 are subject to the inclusion and exclusion criteria defined in Table 3.1.

The filtering procedure was carried out through multiple iterations, first on the primary findings and then on the succeeding sets of outputs after each elimination step. There was no need for extreme rigor in determining when a publication qualified for inclusion. In fact, while it was true

Table 3.1: Inclusion and Exclusion criteria used to filter the results

	ID	Description
Inclusion	IC1	The work is involves Voice Assistants
	IC2	It is related to IoT devices and its ecosystem
	IC3	The publication is applied on a smart space environment (*e.g.*, smart home)
	IC4	It takes into consideration contextual-awareness on user queries
Exclusion	EC1	Duplicated Work
	EC2	Publications older than 2010
	EC3	Publications not written in English
	EC4	The work considers Voice Assistants but is mostly focuses on AI research
	EC5	The work focuses on security about Voice Assistants and
	EC6	Research work focused on definitions and representations

that articles needed to meet the more relaxed inclusive requirements, the exclusive criteria were only relaxed in cases when there were no better alternatives.

Given these conjectures, the results were analysed and filtered according to the previously set criterion. *IC1* and *IC2* define the basic inclusion requirements, restricting existing publications to those focused on Voice Assistants and IoT devices, which represent the basis of this work and the main areas of knowledge. Since these criteria ensure that documents are related to the areas of interest, they also reflect the research questions on Subsection 3.1.1, with *IC1* being the equivalent of *SRQ1* and *IC2* being the equivalent of *SQR2*.

IC3 and *IC4* place an even greater restriction on the resulting publications by mandating that results about IoT and Voice Assistants be applied to a smart space scenario and that contextual awareness be applied in the IoT ecosystem and Voice Assistant in order to better serve the needs of the user when a query is given.

The other remaining points of criteria are exclusive, with all five elements (*EC1*, *EC2*, *EC3*, *EC4* and *EC5*) defining the publications that won't be considered. One of these points removes publications older than 2010 (*EC2*) as to concentrate on more recent studies and avoid incorporating older works into our investigation; due to the rapid development of both technology and the IoT field, it is necessary to implement this restriction because newer solutions are typically more effective and more in line with the goals of this thesis. The *EC4* and *EC5* criteria serve to prohibit works that are part of the main set but have no relevant use for our work. In spite of the fact that these publications typically satisfy the *IC1* and *IC2* criteria, we cannot include them in our research because the primary focus of our efforts is not on AI or security research; rather, we focus on End-User Development and contextual awareness of the surrounding environment. Finally, *EC1* and *EC3* are intuitive, as the goal is to focus on English works and avoid having duplicate publications in our final set.

Keeping in mind the considerations mentioned up above, the following is an outline of the filtering pipeline procedure that was applied to the results of our database searches in order to filter the contents:

1. **Query Search:** The queries described before were run against each of the three academic

research databases mentioned in Subsection 3.1.2. After obtaining each result from each database, the contents were merged together into one collection.

2. **Duplicate Removal:** Duplicate publications were removed from the resulting set, resulting in the application of *EC1*.

3. **Language Filtering:** Queries that didn't match *EC3* were also removed, outputting a set of publications written only in English.

4. **Metadata analysis:** The metadata of the publications (Title, Keywords, and Abstract) were used to filter out the relevant publications, which complied with the inclusive criteria *IC1* and *IC2* and weren't discarded by the exclusive criteria *EC4* and *EC5*. The output was a set of publications guaranteed to be applied on the main areas of knowledge of this thesis.

5. **Introduction and Conclusion analysis:** The same analysis done on Step 4 was also done on the Introduction and Conclusion sections of each publication, being matched against criteria *IC1*, *IC2*, *EC4* and *EC5*.

6. **Overview analysis:** Based on the few publications remaining, the same criteria used in Steps 3 and 4 were reapplied in this filtering step. This was a non thorough full-text analysis and was used to filter out any final publications that may include information not useful for the study of the State of the Art.

A diagram of the steps mentioned above is available in Figure 3.1:

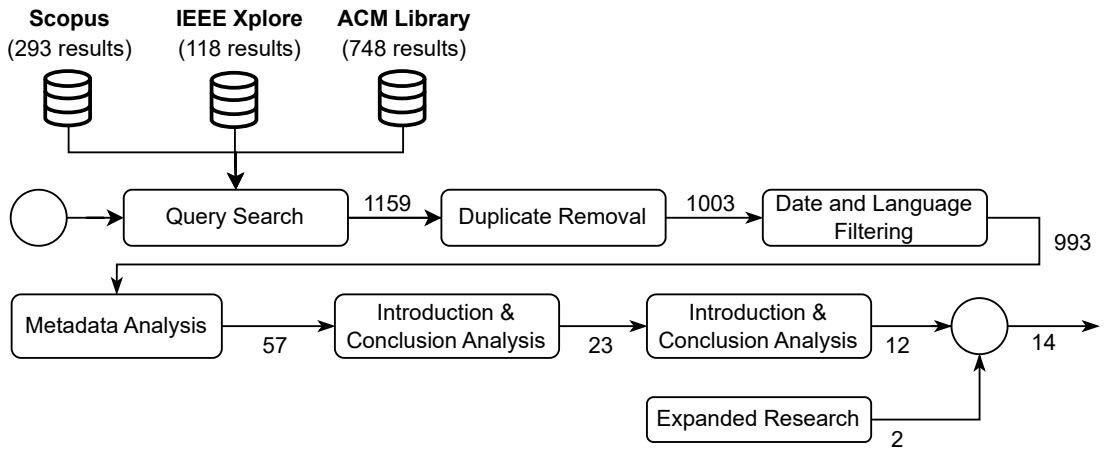

Figure 3.1: Diagram of the applied Systematic Literature Review

3.1.5 Grey Literature

The outputting set from Subsection 3.1.4 included additional tools that were not documented in scientific papers and publications. In order to correct this, the resulting set was augmented with additional research made outside of academic research databases, through Google. The possibility

for this lack of association, can be explained by the fact that some tools may not have academic research, or the papers detailing these contents possess keyword variations from the ones used in the search queries.

3.2 Results

The pipeline presented in Subsection 3.1.4 led to the acquisition of 15 results and 2 grey literature works. The main publications are briefly described in Subsection 3.2.1, with the results being presented in a descending order of relevance to the goals of this thesis. Supplementary studies, such as grey literature, are presented in Subsection 3.2.3.

3.2.1 Main Findings

The main findings for the state of the art analysis are detailed individually, based on their introduction, work done and conclusions. 2 of 15 results represent a survey and a review and are presented in Subsection 3.2.2 for a more individualized review.

André Sousa Lago *et al.* [16] present Jarvis, a conversational assistant serving as a user-interface medium to manage IoT ecosystems. They emphasize the need of a voice assistant in order to manage the repercussions of a more complex ecosystem, which comprises a rising number of household members and an increasing number of gadgets. They observe that, in addition to the complexity of the system itself, identifying the origin of particular interactions becomes more challenging. To tackle the issues presented by this complexity they suggest a voice assistant as the best way to tackle these issues. The assistant makes an effort to resolve these challenges by providing the user with development capabilities, such as the declaration of time-based rules, contextual awareness based on natural interactions, causality inquiries, and a few other features that are relevant to simulate a non-visual interface with the ecosystem.

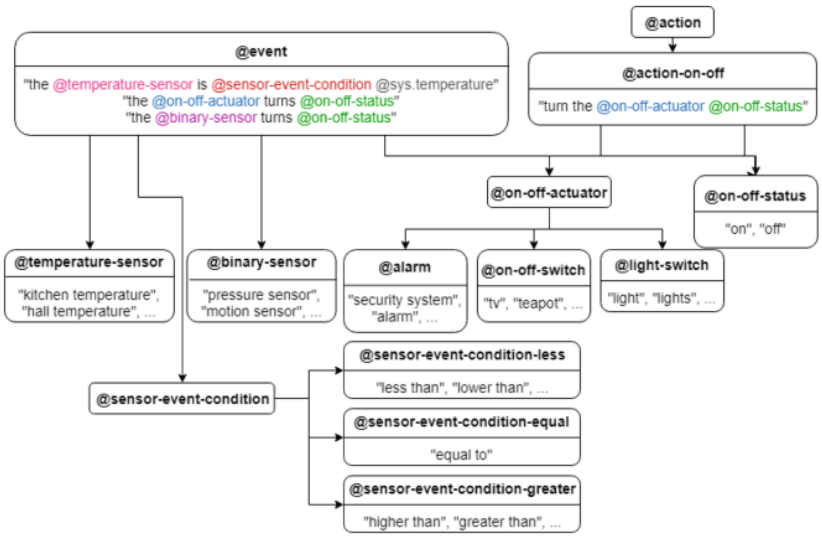

Figure 3.2: The Dialogflow of Jarvis. Taken from [16]

The capabilities of the conversational assistant are defined later on Section 3 of the paper, when the authors refer to conversational awareness throughout communications with the user, giving a use-case as an example. In it, a user queries the VA about why the toaster turned on, to which the VA replies that the user had programmed to turn it on at 8 AM, and then without referring the toaster again, they ask the VA to turn it on 10 minutes earlier.

The work was applied to the Slack group chat platform and was developed using Dialogflow, a Natural Language understanding platform. It works with both text and voice input, the latter of which is supported by Google Assistant. In Figure 3.2, the authors illustrate the representation of their Dialogflow entities and elaborate on the concepts of Entity, Intent and Context, which are used to define the features offered by Dialogflow. The output contents are then sent to a customized backend that processes them according to a predefined pipeline designed to decipher detected entities and definitions in the requests and provide the appropriate response. Based on the clarity of the contents, the backend may query the user for clarification or proceed with the request if it finds that all elements are valid and there are no existing conflicts.

The achievements of this work are evaluated using simulated situations in competition with a conversational interface such as Google Assistant, and a visual programming interface such as Node-RED. According to the findings, Jarvis was superior to these two tools in a number of ways, including the variety and different amount of capabilities it provided. In addition to that, the authors go on to conduct a quasi-experiment consisting of five distinct tasks on a total of 17 individuals, the majority of whom (14) did not have any formal technical abilities. The conclusive research shows that the success rate of Jarvis was quite high (88%) and that users subjectively reported that it was simple to use and enjoyable.

The experimental portion of this study revealed a number of issues, one of which was the fact that users would have a very hard time comprehending the causality inquiries if they were too lengthy because of the information being sent. Other difficulties include the possibility that inquiries such as "*Turn on the lights*" are not that prevalent. Finally, a relevant threat to the validity of the results noted the reduced size of the target population, which could translate to a coverage error.

André Sousa Lago *et al.* [34] extend the work done in [16], incrementing some of its functionalities and displaying a more elaborated sequence diagram of how queries are parsed and handled. It serves as a more elaborated version of Jarvis, with an in-depth description of its definitions, along with more illustrations of the work previously done. In addition to that, more elaborate information is offered, such as elaborated descriptions of how Jarvis handles Period Actions, External Events and Causality queries. Period Actions are handled through the *command pattern* of Dialogflow, by deleting the active command and creating a new one whenever a change is needed, and by using a state-machine to keep track of the current state of the command. Jarvis divides this state-machine into three possible states: a) Where nothing has executed, b) Where the first action has been executed, c) Where the second action has been executed. Other commands, such as commands based on external events are handled with a publish-scribe approach, where message queues are associated to one or more devices and are used for bidirectional communication between each other and Jarvis' backend. The authors define an Event as a form of communication where an IoT device sends a message to the queue with a sensor's reading, and the backend listens to it and checks if an active command depends on such reading. Finally, causality is more elaborated in this paper, with the implementation dictating that when a user wants to know about the cause of a certain event it shall, in most cases, return the immediate possible cause - this is usually the latest action, but it may not be relevant if there are multiple commands that can originate such event. Alternatively, the authors mention a heuristic meant to determine the relevance of commands when acquiring the cause of a given event, which they claim to be a non-trivial solution. Such solution can be handled by replying to the user with the entire chain of events that could have caused such event or even just telling the user the latest possible cause.

Sanket Salvi *et al.* [52] develop a do-it-yourself (DIY) Smart Home assistant, nicknamed "Jamura". One of its strengths, noted by the writers, is the ability to assist the user in decision-making and predictive analysis. Similarly to [16] and [34], this work applies Dialogflow as a text processing tool, and applies it on Telegram and ThingSpeak as "communicational" interfaces with the end-user.

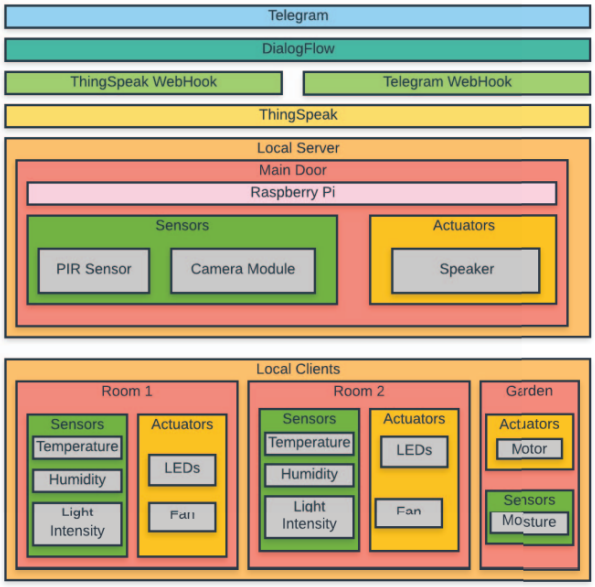

Figure 3.3: The system architecture of Jamura. Taken from [52]

An interesting feature offered by the assistant is its Intrusion Detector/Alert system, introduced with a passive infrared (PIR) motion sensor which is implemented at the lower level of the Local Server section. It is visible in the framework that is made available in the paper (Figure 3.3). Whenever a person appears in front of the camera, the PIR Sensor triggers the Camera Module, which will connect with the Google Vision Service for identification of a human in it. In this situation, and if it is detected, the image is sent to the homeowner on Telegram, requesting further action with a 30-second time-out. In both cases, it will use the speaker to notify the person outside of the given reply. Overall, the system has two chatbots, one for the LocalServer, where it manages the front door, and another chatbot used to manage the home space (Local Clients).

In spite of the interesting approach, the analysis of the results note some particular weak spots. Namely, the amount of limited queries (intents, which represent question-answer pairs) and how the training is extremely limited to a specific set of questions made by the end-user. Additionally, the Chatbot used in the SmartDoor only answered to a specific user, due to a assigned hard-coded user-id.

Long Huang *et al.* [30] develop a Voice Assistant that can contextualize the user based on the sounds of their steps when walking. The introduction elaborates on the low quality of existing voice assistants, where the authors note that users need to speak directly (and sometimes loudly) at the voice assistant in an intelligible and calm manner in order to have their commands understood clearly. Along with other citations, they mention acoustic attacks as possible ways to invade privacy of common user. With that in mind, the authors propose a Voice Assistant that detects the user based on walking sounds, including factors such as

body weight, leg traits and their walking style. They also argue that the sounds can establish security given that an attacker would have to mimic both voice and walking sounds to gain access to the ecosystem and VA's commands. This research is pertinent to the work done in this thesis since it demonstrates how the sounds of footsteps may be utilized to contextualize the user and so trigger a chain of instructions based on the personality retrieved from those steps, when speech alone is not sufficient to identify the individual.

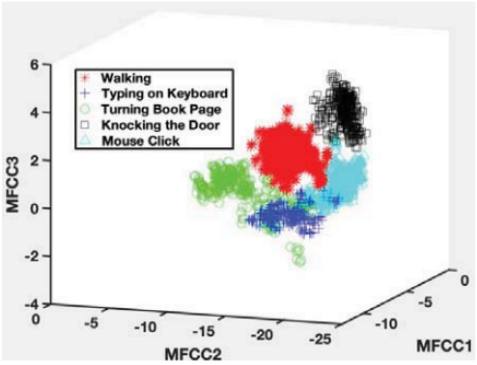

Figure 3.4: Clustering of Sounds using MFCC coefficients. Taken from [30]

The authors note relevant challenges when developing the system by stating that many daily events produce sounds very similar to the sounds of steps when walking. In addition to that, the corresponding sound is affected by the walker's pace, such as when the user is in a hurry. Finally, the authors point out that microphones can only pick up brief audio snippets given that sounds produced by walking steps have a very short duration — as little as 200ms. To fix these, they used the microphone array available in voice assistants, like in Amazon Echo, and used Mel-frequency cepstral coefficients (MFCC) as features in their machine learning algorithm, which were later used to cluster sounds based on their source (Figure 3.4). MFCCs are frequently used in speech recognition systems in order to recognize specific words, terms and even numbers in an audio. Using a majority vote scheme, that is mixed with various classifiers, the resulting work can help identify the user based on the walking steps alone.

The final result analysis done in this work evaluates and contrasts the performance of the classifiers, finding that the k-NN classifier achieves 92% accuracy while the SVM classifier and Random Forest obtain 60% and 80%, respectively. The authors additionally study the effect of the size of the dataset, finding that just 15-20 steps are enough to obtain 82%-92% accuracy. In light of these encouraging findings, the authors finish their paper proposing the addition of new functionalities to the assistant, such as training it to recognize the sounds of when the user walks to exit or enter the home. This would allow the assistant to tailor the environment to the specific person it has recognized, ushering in a more personalized experience.

George Alexakis *et al.* [6] develop an application used to control an IoT ecosystem. They call this application an IoT agent, and it is responsible for handling commands and communicating with the respective micro-controllers available in the ecosystem. In addition to that, the innovation offered by the work of the authors lies on the various third-party APIs and open-source solutions working seamlessly together without the need for intensive development. It is a *proof of concept that describes how an IoT application may be constructed by using a multitier architecture that is wholly dependent on out-of-the-box solutions, hence enabling a shorter and simpler development process.*

For the development of this work, the authors use Dialogflow for the NLP segment of the pipeline, Web Speech for voice recognition when interacting with the system, MQTT for the messaging protocol between devices and Firebase to store real-time data. The architecture of the system is divided into three distinct sectors: User, Services & APIs and Home Automation. The Home Automation sector consists of two microcontrolers, one responsible for controlling the lights and the other responsible for reading the temperature/humidity sensors in the room. This information is then sent to the Services & API sector, where the temperature and humidity data are transferred to the Firebase database. However, the light controller only connects to the database during the initial power-up, with its remaining communication being handled by the MQTT Broker, which builds a direct bridge with the User Sector. This sector consists of a Mobile Device which allows the User to interact with the ecosystem. The device has a web interface, and gives the user the ability to interact with the system through voice, text or even the dashboard, which also displays information available in the Firebase database and therefore including data from the temperature and humidity. Voice commands are converted to text commands using the Web Speech API, and afterwards they go through Dialogflow for processing.

After acquiring the response, the application decides the appropriate action based on specific assumptions of what the type of questions and responses will be which are available on Tables 1 and 2 of the same paper. This leads to one of the major limitations given by this approach which is the simplistic nature of the queries and restrictive amount of capabilities offered by the tool. Most questions are variations of one another e.g.*, "What is the humidity of kitchen?" and "Give me the humidity of kitchen." and happen to be excessively unrealistic, since it is not guaranteed that a user will speak like this in a real environment.*

The performance of the system was evaluated based on response-time and was tested under 10 different types of queries. Each was measured under three distinct types of network bandwidths, leading to the conclusion that the response time is severely affected by the available network bandwidth. Nevertheless, the authors note that even in a worse-case scenario, and under a throughput point of view, the total time is not significant, which makes their work a viable solution for low-cost developments. The users finally note some additions that could be added to the system, such as a sign up feature, additional sensors and rooms, among other things.

***Sakib Ahmed* et al.** *[4] develop a Facebook Messenger Chatbot able to handle an IoT ecosystem with the facilitation of home automation in mind. The work uses Raspberry PI as a processor and includes an application that offers features like face recognition, leakage and temperature detection, smart door locking system and among other features.*

A flowchart representing the main features of the system is illustrated in Section III of the paper. The sensors, such as the Camera, Door Lock and Temperature sensor, connect and communicate with Raspberry PI 3B+ which sends the information to a FLASK Python Server hosted in pythonanywhere. The authors also use the Graph API from Facebook in order to create the chatbot and enabling the user to interact with it using Messenger.

The authors note that the work was successfully implemented, with the Server connecting to the Graph API and the Raspberry PI, enabling smooth device interaction. Nonetheless, the resulting chatbot lacks contextual awareness and seems to be confined to a limited number of instructions such as "LIGHT STATUS" and "Show Temp" which decreases automation and forces the user to keep track of accessible commands. In fact, these commands are unrealistic and are not aimed towards common scenarios unless the user plans to give up on text processing in exchange for command-accuracy, system integrity and simplicity. The strongest point in favor of this work, however, lies on its cheap cost in exchange for high connectivity and flexibility given that it was initially planned with a developing country like Bangladesh in mind.

***Cyril Joe Baby* et al.** *[9] establish a work that is similar to the prior references in that it focuses on a Chatbot that employs NLP methods for receiving queries, controlling other devices connected to the house's LAN, and security features such as intrusion detection and user authorization. Similarly to the work of Sakib Ahmed* et al. *[4], the authors use Raspberry Pi and Python as to handle the sensors and the logic of the Chatbot, respectively.*

Nonetheless, the authors' approach is distinct in that it does not rely on a comprehensive and efficient out-of-the-box solution like Dialogflow, but instead on the NLTK kit provided as a package of the Python language. The authors follow a specific pipeline, starting with text tokenization, moving to stop-word removal, then keyword and action lists detection, and finally execution. Based on the tests that were executed inside the linux terminal, the system allows the user to formulate queries differently with simplistic variations of input, so long the entities and keywords are used. However, this type of communication still lacks context awareness, unlike approaches [16] and [34]. As the writers point out, the work may be enhanced by additional devices such as windows, cabinets, and other equipment. They also emphasize the need of using machine learning algorithms and neural networks, which would undoubtedly help the bot to predict user behavior. Even with these enhancements, the

most important would be to leverage the user's location to streamline inquiries and contextualize requests, making it more innovative and distinctive with previous works.

***Yi, Taeha* et al.** *[65] develop a Case-Based Reasoning (CBR) system that is able to recommend services to Users based on the data acquired from IoT devices. The work does not focus on contextual awareness; nonetheless, it serves as an important basis because of its capacity to anticipate user preferences and choices and act upon them without additional input. As a result of this, the system may be easily incorporated in a smart assistant in order to anticipate actions, orders, and choices before the user chooses to request them, essentially leading to a context-aware solution.*

The authors begin the work by stating that it's important to analyse the IoT ecosystem as a whole and not from a device-to-device point of view given that user actions affect multiple devices simultaneously. Therefore, their CBR system studies all the data from every device and then uses previous occurrences to solve new incurring ones. To achieve this, the authors first describe the user experience inside an IoT space based on the Function-Behavior-Structure (FBS) ontology and then later use this information to define the Case template used in the CBR system. In it, the Case template is defined and structured into a problem, solution and an outcome. The authors define the problem as the existing data (user data, device status data and environment data) and the solution is defined as the suggestion which leads to a feedback which is the outcome.

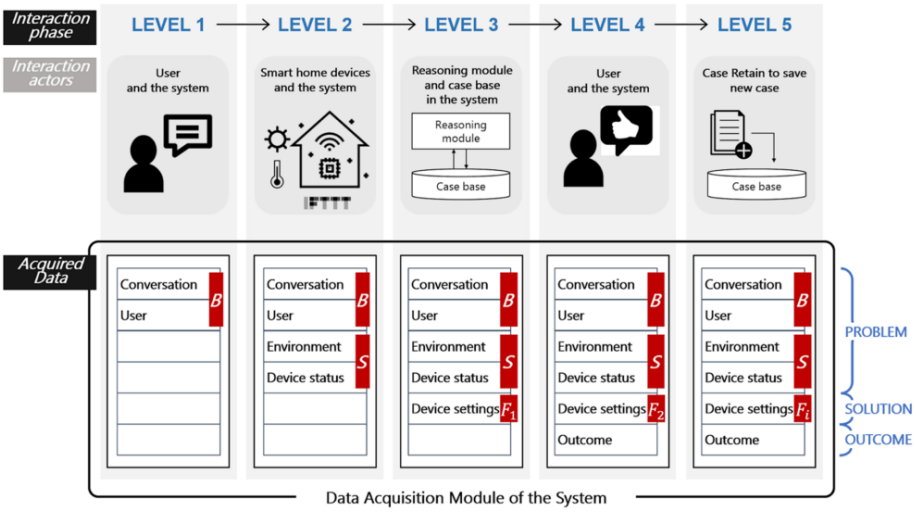

Figure 3.5: Flow of data acquisition of the CBR system. Taken from [65]

Instead of collecting a large amount of data, the authors' suggested CBR system collects data gradually, building up cases as it interacts with the user. The processes for acquiring instances are detailed in Figure 3.5, which is also accessible in the linked publication. Throughout the process, the system collects data from the user through direct contact, then

from the surroundings and the IoT devices in the home. Following this, the system will proceed to evaluate the data and provide a response based on past situations, resulting in a potential solution ***Fi****. If the system receives negative input from the user, it may seek another solution, and when the user ultimately accepts it, it is recorded as a new case in the case base. This approach follows the prototype of CBR, by applying the Retrieve, Reuse, Revise and Retain sequence.*

The final system was assessed by seven field specialists using 30 simulated cases. The experiment was conducted manually, with one case (the query case) being compared to five additional instances (options), and with participants being asked to choose the option that most closely resembled the query case. The study of results was done comparing the answers from the specialists with the answers given by the CBR system. Results showed that 54% of the CBR system's responses were similar to the experts' responses, but a specific query gave a 17% similarity, indicating the need for a more sophisticated approach between Levels 3 and Level 5 (Figure 3.5); this laid the groundwork for possible improvements, while also emphasizing that the CBR flow is beneficial in order to acquire data from the environment.

***Simone Gallo* et al.** *[23] develop a RuleBot that uses Machine Learning (ML) and NLP in order to allow users to set up automation rules. The work is driven by the abundance of commercial and research VA tools which do not take into account conversational capabilities. The previously mentioned approach on Jarvis [16] is mentioned in this paper although targeted with weaknesses such as the inability to set delayed actions conditioned to certain requirements. The authors also stress the ability for RuleBot to define triggers with negation.*

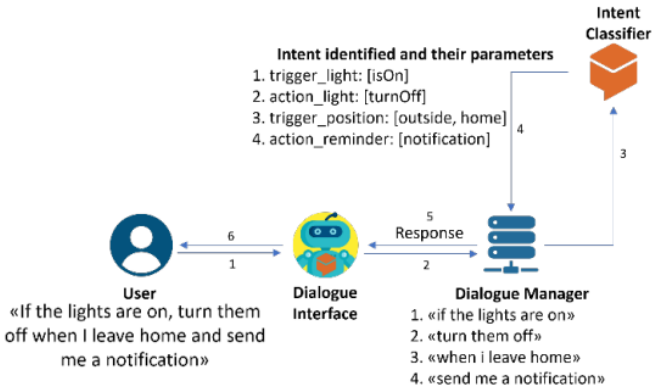

Figure 3.6: Implementation of RuleBot. Taken from [23]

In order to develop this tool, the authors use a language capable of handling boolean operators, such as AND / OR, for multiple-trigger application along with the definition of sequential actions upon a trigger application. Triggers can also be organized into three different categories which depend on the User (emotions and physical state), Environment

(light, temperature) and Technology (TVs and smartphones). The actions offered cover the ones of Jarvis, while extending it with the possibility of setting reminders, alarms and changing the state of home appliances.

The RuleBot's implementation went through many revisions. Initially, the authors began with a basic version (V1) in which intentions were centered on a single trigger and action and were separately referred to a particular device. This presented issues since it limited the option to one trigger-action combination, making the dialog unrealistic. The enhanced version (V2) incorporated a multi-type approach, enabling interactions with up to two triggers and two actions. However, the authors identified several flaws in this method, including that creating distinct intentions for each scenario based on more than one-to-one connections would present scalability challenges (in specific cases, combinations could reach up to 1653 possible intents for two triggers). To address this, the authors devised yet another enhancement, which they labeled V3. The method, seen in Figure 3.6, employs a Intent Classifier to categorize the intents and a Dialog Manager to divide complicated requests into smaller subsets that are then forwarded to the Intent Classifier one by one. This enables the identification of relevant intents in a user inquiry and their associated parameters, which are also validated before being returned to the Dialog Manager and being defined into a rule.

The most recent version of the bot was tested with ten participants, eight of whom were new to programming and two of whom only understood basic web-programming languages. The test was run against TAREME, a user interface that allowed the construction of trigger-action rules. Each user was put through three distinct scenarios to test this, one with one trigger-action combination, another with two triggers and one action, another with one trigger with negation and two actions, and lastly with two triggers and two actions. Ultimately, the RuleBot was quicker than TAREME, but it produced less precision in certain cases; in fact, error analysis revealed that accuracy was greater in the visual environment than in the conversational one. Nonetheless, RuleBot was much simpler to deal with, even on the NOT operator application, where just one participant reported some issues in getting the bot to comprehend. The papers's conclusion underlines the strengths and weaknesses of the work against a visual interface, with the authors noting the easiness of the conversational assistant and expressing intention to enhance the tool with causality queries.

***Varsha K Patil* et al.** *[49] develop a virtual-assistant based system that controls the ambient light in the room based on the user's emotions and managing temperature and humidity. The system is also able to determine a user's identity and respond to them differently based on the rights that have been assigned. For example, the VA will open the door to a registered user, but it may not regulate the ambient's light or appliances based on that user's emotions if they don't have permissions for such. In order to implement the face recognition the authors use Haar Cascade, an object detection algorithm, whereas they detect emotions using a Convolutional Neural Network (CNN) and SVMs. In addition to this, the VA is also capable*

to determine the number of people in a room using 2 PIR (motion sensors) that the authors name "entry" and "exit sensors". This number changes whenever one of these sensors is triggered, updating the count and therefore registering whenever a user leaves or enters the home. Based on their own tests, the authors note that their system has a 56% accuracy with CNN whereas SVM leads to a higher accuracy value of 58.9% in user emotion detection. They finish the paper by declaring the work as a cognitive result and an emotion-linked senso+visual product.

***Konstantinos Michalakis* et al.** *[42] introduce the idea of a device with Augmented Reality (AR) capabilities that allows users to identify sensors and their functionalities all around them. In order to achieve this, the work augments an existing IoT architecture [24] by introducing a context aware module and a smart object management module to it. The contextual awareness is based on two possible context sources: sensor data (such as lights and sounds) and the user profile in order to handle privileged resources (such as camera access).*

The authors note relevant challenges when trying to track smart devices, as they use trackers, which offer limitations for many situations. As examples, the authors indicate that devices may change positions over time, users may want to access the devices without being around them and other situations which may introduce challenges specific of non-dynamic markers. In order to solve this, the authors introduce the possibility of using Radio-frequency identification (RFID) tags and Bluetooth Low Energy (BLE) beacons. This last strategy, however, would need structural adjustments, since it would no longer be the user who established communication with the gadget, but rather the device itself.

No result assessment is done, with the authors leaving those tasks as future work.

***Maksym Ketsmur* et al.** *[31] provide information about a system that can assist users through text and speech as input and output modalities. The authors begin by claiming that these modalities are sometimes the only options available for specific circumstances, using as an example the value of speech output for people who are visually impaired. In addition to that, they stress the utility of speech based interactions, describing them as "fast, intuitive and hands-free". Additional points are made on the current simplicity of approaches and the very challenges of existing solutions, ranging from the different structure of smart spaces, the various inhabitants (which may or may not have impairments) and even the language of these inhabitants. In fact, most proposals only support English as its main language without considering the specificity of different and future situations.*

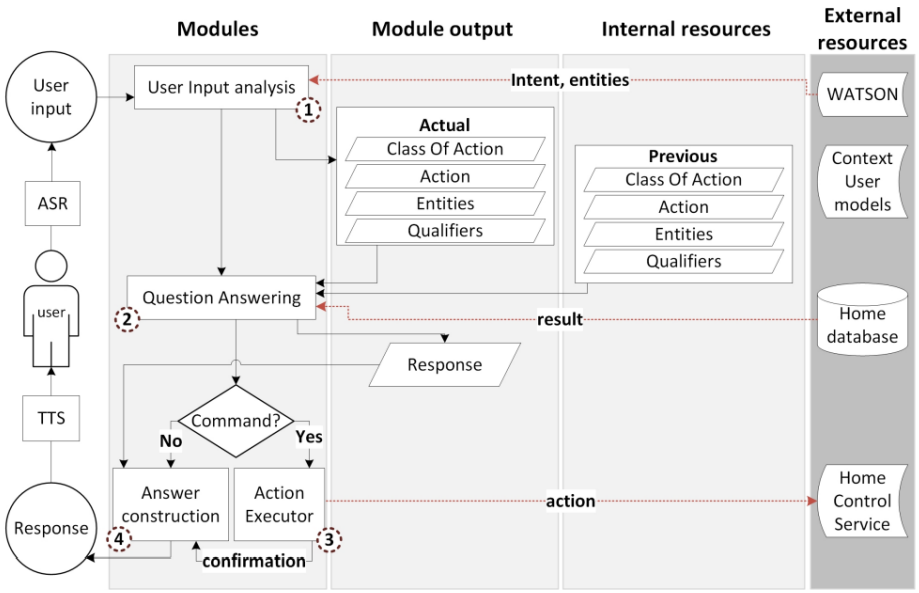

Figure 3.7: Architecture of the smart home assistant. Taken from [31].

The voice assistant was implemented using a user-centered design that was focused on a particular family. Two members of the household had relevant disabilities, such as eyesight issues and temporarily decreased mobility. One member of the family is four years old and cannot read. Given these conditions, the authors presented three scenarios that described the accessibility issues caused by disabilities. The first focuses on accessibility for minors, with a scenario involving the 4-year-old household member. The second focuses on accessibility for those with limited eyesight, while the third focuses on accessibility for home information everywhere, which is important for people with limited mobility. In addition to this, and unlike previous implementations, the authors chose the IBM WATSON Assistant due to its innumerable advantages such as its off-the-shelf identification of dates and numbers.

The architecture defined for the system is available in the paper and is illustrated here in Figure 3.7. The authors describe this system as a sequence of steps, starting from the user input analysis which identifies the intent, actions and entities relevant for the system from the input. After that, the output is relayed to the next module, named Question Answering which is responsible for handling the response. If required, it will create a query to the Home database to get the information needed to provide a response, or it will relay the command to the Action Executor module, which will then formulate a command to the Home Control Service. Regardless of the situation, the Answer Construction module will always generate a textual response based on the steps taken on previous modules which is later converted to voice using text-to-speech (TTS) techniques.

The authors study the results of this work by simulating the previously mentioned scenarios. Based on the illustrated results, the assistant is capable of retaining the context or

topic (mostly the object in question) of the conversation without having to refer to the object again. It is also capable of handling complex queries, such as: "What was the electricity consumption in the bedroom this month", followed by "And how much does this cost?". This leads to a richer experience, even though it does not take advantage of the environment's context. This conclusion is based on one of the simulated scenarios, where the child asks to turn on the heater and the system asks which one, rather than using the ecosystem's context. In fact, this last point is set out as possible future work along with evaluation of users, especially users with other needs.

***Mahda Noura* et al.** *[47] present VOISMA, an approach to identifying the location of the user based only on their voice. The work is motivated by the existing inability of smart home interfaces like Amazon Alexa and Google Assistant to possess indoor detection methods. As such, the authors focus on the detection of a user's location based on their voice, and implement a prototype while assessing its results.*

A physical space map must be created during the setup phase, and the microphones inside the smart environment must be deliberately placed and their coordinates marked, as described in the method's specifics. User input is required for both the mapping and coordinates of each microphone, which together give the system a "spatial notion" of the home space. After the setup, the toolchain starts anytime a speech signal is detected, filtering out background noise, analyzing the signal, and finally determining the user's location based on the measured distance between the microphone and the speaker. The authors set-up three different cases in order to evaluate the results of the system: In the first case, the end user is within the range of one microphone; In the second case, the user is within the range of two microphones; In the third case, they are within the range of three microphones. In all cases they register low latency values (2.7ms). The authors also evaluated the system's accuracy in the presence of background-noise and discovered that their method is effective only with moderate to medium levels of noise. Another limitation of the system is that it can only serve a single user at a time (mixed sounds, different voices, and other conflicting scenarios), having been set for "Future Work".

3.2.2 Surveys and Reviews

In our previous 15 findings, 2 of them were labeled as Surveys and Reviews, with no specific effort on creating new methods and tools for the IoT industry. As a result, we discuss those 2 publications independently from the previous study.

Arindam Giri *et al.* [27] conduct a thorough assessment and survey of the IoT field. They begin by examining the Internet of Things Architecture, noting that despite several ideas for a flexible layered architecture capable of dealing with heterogeneous devices, none has yet been chosen as a standard. TCP/IP, an internet architecture, is noted as being incapable

of managing the ever-increasing size of IoT, and they cite scalability, safety, and quality of service (QoS) as three of the primary reasons why current internet designs cannot be used to IoT. Nonetheless, the authors examine four potential architectural designs, contrasting their strengths and disadvantages.

The authors explore current technologies for identification, sensing, and communication. They cover methods for assigning a unique identification to IoT devices (Electronic Product Code and ubiquitous code) as well as various communication modes including Wireless Sensor Network (WSN), Wi-Fi, and Bluetooth. Certain protocols, such as MQTT, REST and XMPP are referred in the same section, referring to these protocols as "open, secure and spam-free communication between users". The authors also state that DNS is necessary to handle scalability challenges inside networks.

The study also examines Middleware Technology, which includes the most recent state of the art in middleware systems. Interface protocols, device abstraction, contextual awareness as part of middleware, and application abstraction are all mentioned. The work then mentions the applications and societal implications of IoT, with a focus on smart settings such as homes, workplaces, and plants. Another use mentioned is healthcare, which refers to the monitoring of a patient's data and the aid of a doctor. The authors also concentrate on other applications such as Smart Cities, Transportation, Agriculture, the Pharmaceutical business, and society. In addition, key topics such as Massive Scaling, Security, and BigData are explored around the end, attempting to provide solutions to challenges and elaborating about the research direction.

Sumathi Balakrishnan *et al.* [10] execute a superficial review on smart home technologies. The purpose of the work, as detailed by the authors, is to discuss the definition and purposes of a smart home along with the study of challenges and proposal of potential solutions. As such, the work starts by defining the concept of Internet of Things and the purpose of a smart home. At each step, multiple references are made to existing works as a way to index the current state of the art. The work follows by targeting the benefits of a smart space, focusing on elements such as Quality of Life, Energy efficiency, remote accessibility and sensor network monitoring.

After the introduction of the concepts and utility of smart homes, the authors exemplify existing technologies with smart homes in smart cities, and relating them to IoT and Cloud Computing. This leads to the introduction of challenges, which the authors elaborate in detail, focusing on critical issues such as User acceptability (whether users would accept these technologies), interoperability, relevancy of data extraction, commercialization (delays and bureaucracy which make it harder to put the technology on the market), lack of self-healing systems, security and privacy, cost of the technology and battery life, which will need to be extended in order to accommodate the growing pace of IoT. The authors finish this work

with a brief elaboration on possible solutions to these challenges, namely the problem of inserting new devices in an already existing and running space.

3.2.3 Grey Literature

In order to understand specific concepts that were introduced in the details of each paper in Subsection 3.2.1, this subsection expands the results with grey literature search aimed at the concepts of **Case-Based Reasoning** and **Function-Behavior-Structure** which were pivotal concepts for the work of Yi, Taeha *et al.* [65].

Case-Based Reasoning [1] is known as an approach where previous similar situations are used in order to solve a new problem. As such, knowledge and information obtained from previous events and choices are taken in order to analyse and predict future events. This process is widely used in Philosophy but is also applied in the fields of Artificial Intelligence. Unknown to us, our minds apply this method every day and it is even argued that all reasoning is based on previous experiences. An example would be a car mechanic who usually fixes cars based on previous experiences he had with car engine problems or electrical issues and other cases. Another example is used in Justice systems, where precedents are used as examples for new cases. The concept became a powerful method for computer reasoning and was later formalized into a four-step process: (1) *Retrieve*, where the system retrieves the cases relevant to the problem we are trying to solve; (2) *Reuse* where it maps the solution of the relevant case to the problem in question; (3) *Revise* where the mapped solution is tested and if it doesn't match the desired cased, it's revised usually based on feedback; (4) *Retain* where after successful feedback, the solution is saved and a new case is created, possibly being used in the future to solve new problems related to it.

Function-Behavior-Structure [25] is a representation of design objects, where it can be applied to any engineering discipline. It conceptualizes objects based on three descriptive categories: (1) *Function*, (2) *Behavior* and (3) *Structure*. In (1) the purpose or function of the object in question is defined. For example, the function of a sensor in IoT is to detect physical properties and convert them into binary data capable to be read and transferred throughout the internet. In (2) the attributes related to the design's structure (3) are defined. In the case of an IoT sensor, the attributes can be voltage, accuracy, minimum detection range for LiDAR sensors and others. Finally, in (3) we defined the structure of the component and the respective connections. In the case of an IoT sensor, this can change a lot based on the sensor, but a LiDAR sensor would have a laser source, a receiver, a tilting mirror and other components [22] that make the execution of its function possible.

3.2.4 Results Categorization

Based on the analysis done to each work referenced and described in Subsection 3.2.1, we categorize each of the results based on six parameters. A listing of each is available below, with

Table 3.2.4 referencing the characterization of each.

	Time-based	**Location-based**	**User ID**	**Previous Context**	**Command-flow**
Lago *et al.* [16, 34]	✓				✓
Huang *et al.* [30]		✓			
Baby *et al.* [9]			✓		✓
Taeha *et al.* [65]				✓	✓
Patil *et al.* [49]			✓		✓
Michalakis *et al.* [42]			✓		
Noura *et al.* [47]		✓			

Table 3.2: Categorization of the works by the type of context-awareness used.

Main scope: Some solutions were focused on general-purpose assistants, capable of letting the user define rules and instructions, whereas others targeted sensor detection techniques through data without any user interference; in this last case, users did not have a real impact and the focus was to augment assistants with more autonomy. For this feature, the possible values are: Rule Management, Conversational Interface, Detection, Prediction.

Interface: A portion of the solutions used voice recognition, while others were either text-based or didn't have an interface at all. For this feature we only consider conversational interfaces and it can take three possible values: Chat, Voice, Dashboard, AR or N/A.

Context Aware: Works may use sensor data to better serve the needs or requests of the users. If the context of the situation is taken into consideration, then this feature is checked.

Dialogflow: Dialogflow was occasionally used, given its position as one of the best natural language processing systems to this date. This feature may be ticked or unchecked to indicate whether or not this tool was utilized in the task.

Machine Learning: Some of the results made use of Machine Learning techniques, with data input and classifier testing.

User Identification: Some works augmented the VA with user identification, which allowed the voice assistant to react differently based on their assigned permissions.

In addition to this categorization made, the **Context Aware** attribute can be further elaborated into individual categories, which is available in Table 3.2.

Table 3.3: Categorization of the work results

Work	Main scope	Interface	Context Aware	Dialogflow	A.I - ML[1]	User Recognition
André Sousa Lago *et al.* [16] [34] [2]	Rule Management	Both	✓	✓	-	-
Sanket Salvi *et al.* [52]	Rule Management	Text	-	✓	-	✓
Long Huang *et al.* [30]	Detection	N/A	✓	-	✓	✓
George Alexakis *et al.* [6]	Conversational Interface	Both	-	✓	-	-
Sakib Ahmed *et al.* [4]	Conversational Interface	Text	-	-	-	✓
Cyril Joe Baby *et al.* [9]	Conversational Interface	Text	✓	-	✓	✓
Yi, Taeha *et al.* [65]	Prediction	Voice	✓	-	✓	✓
Simone Gallo *et al.* [23]	Rule Management	Both	-	✓	✓	-
Varsha K Patil *et al.* [49]	Detection	Dashboard	✓	-	✓	✓
Konstantinos Michalakis *et al.*	Visualization	AR	✓	-	-	✓
Maksym Ketsmur *et al.*	Conversational Interface	Voice	-	-	-	-
Mahda Noura *et al.*	Detection	Voice	✓	-	-	-

1 - Solutions that used AI or ML through third-party tools were not considered. AI solutions, such as NLP were considered.
2 - Both works from André Sousa Lago *et al.* were merged due to the focus on the same tool and given its work-of-extension nature.

3.2.5 Publication History

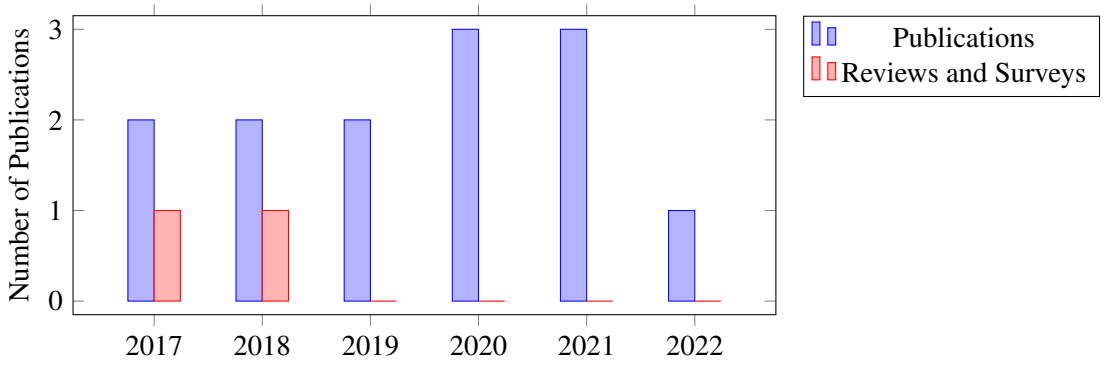

Figure 3.8: Publications per year.

The papers examined in earlier sections were analyzed and categorized according to their publication date. This information is graphically available in Figure 3.8, with the research papers being separated from surveys and reviews. We can observe that most publications are from recent years (2017 onward), with the exception of one publication which is from 2015.

3.2.6 Result Review

Based on the initial results, we defined a grouping of features in Subsection 3.2.4 that allowed us to create Table 3.2.4 and categorize each of the works obtained, where we concluded the following:

In terms of scope The Rule Management, Conversational Interface and Detection method scopes were the most popular, with 3 results each. Other works focused in Prediction and Visualization techniques in order to augment contextual-aware methods.

The interfaces were varied with most works implementing at least one, in order to allow the user to understand what was happening in the ecosystem. Text was the preferred interface, followed by the voice. Only one work did not implement an interface in general, having focused uniquely on detection methods.

In the context aware domain around 50% of the works implement some sort of context awareness, be it through user detection, comprehension of complex inputs and most commonly intrusion detection. However, only a small part of this percentage implemented a more rich and in-depth contextual awareness.

Dialogflow is used by 4 projects as an efficient method for NLP. We can confirm that most works that use Dialogflow do not use ML or AI methods, making the development process substantially faster and much more focused on the tool and aims of the research.

Over half of the publications developed a **Machine Learning** method to apply NLP on the user input. Those who did not, either applied Dialogflow or went for other alternatives in order to process the input text. Generally, the results produced from these techniques were less elaborated.

User Identification happens to be intrinsically connected with Context Awareness, where most works focus on the identification of the user for protection and therefore applying contextual awareness to the logic of the system. Nonetheless, the application of this contextual-awareness is not very rich and seems to lack the ability to cause a noticeable experience to the user.

3.2.7 Survey Research Questions

Considering the previous results and the overall analysis, the previously defined Research Questions in Subsection 3.1.1 are answered:

SRQ1 *What solutions exist for voice assistants?*
There exists a total of 12 results offering relevant solutions to voice assistants. Even if the application does not target voice interfaces, the elaborated framework is capable of being expanded with a speech-to-text technology, rendering similar results.

SRQ2 *Based on the results of **SRQ1**, which of those are relevant to IoT powered smart spaces?*
Every work obtained from **SRQ1** remains relevant for the IoT ecosystem with most works adapting their solutions to smart spaces incorporated with IoT devices.

SRQ3 *Based on the results of **SRQ2**, which of those involve context awareness in decision-making?*
Only 6 out of the existing 12 results employ contextual awareness. However, only 3 of these results specialize in this result, aiming to offer a rich and unique experience to the user.

3.3 Conclusions

After reviewing a number of articles and projects dealing with contextually aware environments in IoT smart spaces, we can confidently say that a broad variety of solutions exist, each addressing a unique set of problems. Nonetheless, numerous concerns remain unresolved, and the precision of these models is insufficient to make them economically feasible. As previously stated, some of the solutions attain average accuracy results (50-60), while others are not thoroughly examined to reach a definitive conclusion. A significant part of the efforts reviewed fall short of producing a contextually aware and natural-speaking assistant. Those that seek to apply this method also test the findings with a small number of participants, rendering the results insufficient to make a fair judgment. For the exception of works [16] [34] and [23], every other work targets individual points but fails to address contextual-awareness in depth. In contrast, most other works rely on user identification and/or localization but fail to employ relevant contextual-awareness methods beyond granular features such as access permissions (or none at all), which although fits the criteria of contextual-awareness, they fail to significantly improve the individual experience of the end-user; instead, they treat the end-user as the entire house-hold and limit the scope of the development to simplistic and unnatural speech patterns. This makes us believe that there's still a lot of room for improvement and that the strengths of each work should be considered in order to implement a more general solution that focuses on comprehensive contextual-awareness.

3.4 Summary

In this chapter an in-depth analysis of the current state of the art is done, evaluating currently existing solutions and categorizing each individually before writing a final conclusion on the contributions done until now. To achieve this, Section 3.1 introduces the methodology for this work, beginning with the research questions, the databases used, search queries, filtering of results and a brief mention of the grey literature.

Following that, Section 3.2 makes an overall analysis of the results, individually describing the main findings, the surveys and reviews and grey literature. It includes a categorization of the results, the history of publications in a histogram and a review of the results.

Finally, Section 3.3 describes the conclusions obtained from the analysis done in Section 3.2, with a brief statement at the end of what's possible to accomplish in the future.

Chapter 4

Problem Statement

Contents

In this section, we will discuss the research topic, the hypotheses behind it, the suggested solution, and the validation procedure that will be done for this work. In Section 4.1 we elaborate on the current issues based on the data obtained from the State of the Art in Chapter 3. We follow the chapter with Section 4.2 where we introduce the crucial aspects that our work plans to focus on. Following that we define the hypothesis of our work in Section 4.3 and conclude the Chapter with a the research questions and methodology in Sections 4.4 and 4.5, respectively. Section 4.6 makes a a brief summary of what is discussed throughout this chapter.

4.1 Current Issues

The Internet of Things (IoT) represents the connectivity of everyday devices to the Internet. The growing complexity of these devices and the ecosystems themselves introduce additional challenges with particular solutions for many specializations. As the field itself grows, the perspective and expectation of end-users also increase, which usually translates into a need for a more intelligent and less invasive assistant. This imposes further limits on conversational techniques, where the automated assistant should only demand extra input when it's absolutely necessary.

As mentioned previously, contextual awareness acts as a catalyst in order to prevent additional input requests and to introduce a more independent and autonomous system that uses the surrounding data and the state to its advantage, but many problems make this difficult to witness. The **first problem** is that current solutions do not take advantage of this and they require consistent user

guidance throughout most steps. Such need for user monitoring defeats the purpose of a voice assistant, which carries the strength of being hands-free and quick. From the analysis on the current state of the art, works [30][6][4][23][31] fail to introduce any type of contextual awareness. The remaining works that introduced contextual awareness, most only achieved 50-60% accuracy, or introduced user detection without properly adapting it to the ecosystem and its queries.

The **second problem** lies on the low support for more complex queries. Unfortunately, in order to have a meaningful interaction with most modern systems, users need to streamline their own requests in a way that allows the system to understand what they're talking about. In most cases, the voice assistant may come across as a hindrance because of its shallow and unnatural conversational style, rather than as a helpful addition to the user's lifestyle. A better approach would relieve the user of this duty and enhance the voice assistant in order to allow for more normal and organic discussions, while focusing on improving the system's accuracy.

From an holistic point of view, the most concerning issue is the sparse collection of relevant solutions, which do not really attempt to complement each other. In fact, given the analysis done in Chapter 3, it's possible to infer that some solutions overcome the weaknesses of others and that by attempting to merge two different paths of development, the resulting system could be much more accurate and complete. Unfortunately, as most works are independent and they do not seek to complement existing ones (for the exception of [42] and [23]), this delays the advancement of VAs in IoT and increases concomitance.

With this, we conclude that we have many problems that motivate the development of our work. Developing an accurate and contextually-aware Virtual Assistant would be critical for IoT research, resolving many of the complexity difficulties in large-scale smart environments and enabling non-skilled people to control those same areas with ease.

4.2 Desiderata

Having defined the existing issues of existing works and the analysis done in the Chapter 3, we define the goals desired with this work:

1. We want a Voice Assistant that is able to handle and understand complex commands, which may consist of an aggregation of clauses or commands through AND / OR operators.

2. We want this assistant to be able to use the data from local sensors to formulate a contextual case of the request, allowing for the assistant to imply which devices are being mentioned without the end-user having to mention them. Both cases, with or without mentions should be supported and the support of complex queries should still be applicable to this case. In addition, we want separate commands to keep the context of the previous interaction, unless the topic is directly changed by the new query.

3. We want the assistant to be able to predict user behavior based on previous patterns of usage and most requested commands.

4.3 Hypothesis

Having defined the goals and objectives, the hypothesis of this work is defined as follows:

> *"In smart spaces, it is possible to empower virtual assistants with contextual awareness based on previous and current knowledge so that complex requests require fewer interactions and less information exchange."*

In a more standardized form, the goal aims for a virtual assistant enabled with contextual awareness that can predict the devices or objects referenced in a user's query using data from previous and current events in order to make the best inference possible. This does not exclude direct queries that lack the need for contextual awareness, such as queries similar to "Turn on all the lights in the Kitchen" which already contain the entire context and do not require the assistant to guess which lights or which room is being refered in the request.

4.4 Research Questions

In order to validate the hypothesis we have mentioned and direct our efforts toward the previously mentioned desiderata, the three following research questions have been formulated:

RQ1 *Can we provide a Voice Assistant that employs comprehensive contextual awareness?*
To make the experience better for end-users this work aims to make the voice assistants effective and to make communication faster and seamless with less interactions.

RQ2 *Can the Voice Assistant be able to understand naturally complex queries?*
Besides contextual-awareness the Voice Assistant should be capable of handling more natural and non-structured queries. The user need not limit themselves to a strict structure and should be allowed to express their own requests in whatever seems more natural to them.

RQ3 *Can the Voice Assistant automatically assume certain usage scenarios when the user does not provide complete information?*
Using contextual-awareness we aim to develop a voice assistant that can predict user behavior and therefore augment its context based on commonly used queries or requests around a specific time.

4.5 Methodology

This work will begin by modeling and developing a Voice Assistant equipped with contextual awareness in smart spaces. Sections not related to the logic of contextual awareness, such as NLP and other section, will be applied through out-of-the-box solutions. The resulting system will be verified and tested by a small group of users (consisting of approximately 10-15 participants) with varied levels of familiarity with Voice Assistants.

The work will define a specific amount of tasks that will be given to each user, and the following attributes will be evaluated:

Time taken: The time that the user took to complete the task (in seconds).

Quantiative feedback on the complexity of each task: An evaluation from 1-5 based on the usability of the voice assistant.

Feedback on the VA's overall intuitiveness and simplicity: A qualitative and general overview of what the VA could have offered.

Based on the results obtained, a statistical inference will be made on the accuracy and performance of the system and it will then be compared to previous literature. Conclusions will be drawn from these results, with the discussion of improvements, weak points or successful innovations achieved with it.

4.6 Summary

In this chapter we discuss the problem statement of this work. We begin with a detailed description of existing issues in Section 4.1, where the results from Chapter 3 are analysed and the issues are stressed in a more detailed format with a perspective for improvements. Section 4.2 introduces the goals and objectives with this work, which helped define the hypothesis present in Section 4.3. The questions that this work will seek to answer are present in Section 4.4 and the methodology that helps trace and visualize the steps to be taken for the development and analysis of the work is present in Section 4.5.

Chapter 5

Implementation

Contents

In this chapter we elaborate on the details of the implementation of our work, describing the inner-workings of our components, the architecture and other relevant information that we believe to be crucial to understand how our system works. Section 5.1 does a simple overview on how the system works. It describes the environment simulation we developed, the assumptions taken, the NLP solution picked and state-agnostic devices. Section 5.2 describes the overall architecture of our system, relevant modules and the way the system works when a request is sent to it. Section 5.3 describes how we leverage existing data about the user's location and devices in order to apply contextual-awareness. In it, we also describe how rules are executed and possible ambiguities we sought to solve. Finally, we include a conclusion section that summarizes the results of the contents described.

5.1 Overview

In this work, we implemented a simulation of a small voice assistant system that's driven by contextual awareness. Our aim was to give this voice assistant the ability to predict relevant entities or devices within a user's query while avoiding disruptive questions that would compel the user to disengage or shift their attention solely to respond to the request, thus detracting from their enjoyment of other activities.

As such, the voice assistant system we developed tries to offer an approach to improving user interactions by incorporating contextual awareness. By leveraging Python's versatility and extensive libraries, we were able to create a robust foundation for the implementation of the logic of the voice assistant.

To enhance the natural language understanding capabilities of our voice assistant, we integrated the WIT.AI [40] platform into our system. WIT.AI provides advanced linguistic processing and machine learning techniques, enabling interpretation and extraction of relevant entities from user queries. Through the integration of WIT.AI, we aimed to improve the voice assistant's comprehension of user commands, especially when referring to specific devices or equipment. Initially, we chose Dialogflow for the same intents and purposes, however, we found several issues when trying to integrate it with our project. The first was that training took more time than we expected, effectively slowing our development progress, and the second issue lied in the consistent inaccuracy of small textual differences that can change the meaning of a sentence. These challenges are elaborated on Section 5.3.2. We also note that the handling of rule conflicts and other specifics of our thesis is part of a work that was done in parallel to this one and was developed by Pedro Ponte [51].

5.1.1 Environment Simulation

The implementation of a real-scale voice assistant that would support multiple device connections within a real environment posed significant challenges, primarily due to time constraints. In addition to this, testing the system would become much more complicated, as users would have to relocate just to make these tests. In addition to this, some of the results would be difficult to measure quickly, such as when asking the assistant to turn on the lights in a different room than the one we are in. Consequently, we devised a simulation interface that enables users to simulate and manipulate the entities within a house plan, allowing them to observe events unfold as they interact with the voice assistant, which in turn directly interacts with the simulation interface.

The simulation interface is implemented as a web-view, utilizing standard web technologies such as HTML, CSS, and JavaScript. This implementation choice offers several advantages in terms of accessibility, compatibility, and user experience. By making this choice, we can structure and organize the interface's elements in a semantically meaningful way, facilitating navigation and understanding for users.

This web-based approach ensures compatibility and accessibility across different platforms. Figure 5.1 offers a preview of the simulation interface, highlighting its intuitive design and user-friendly interface. Users can interact with the simulated environment using their mouse, enabling them to actively engage with and exert control over various elements within the interface.

Additionally, the simulation interface includes a weather simulation feature, allowing users to access and simulate external weather conditions, including temperature and humidity. This capability proves invaluable when establishing rules and defining parameters that depend on these weather variables.

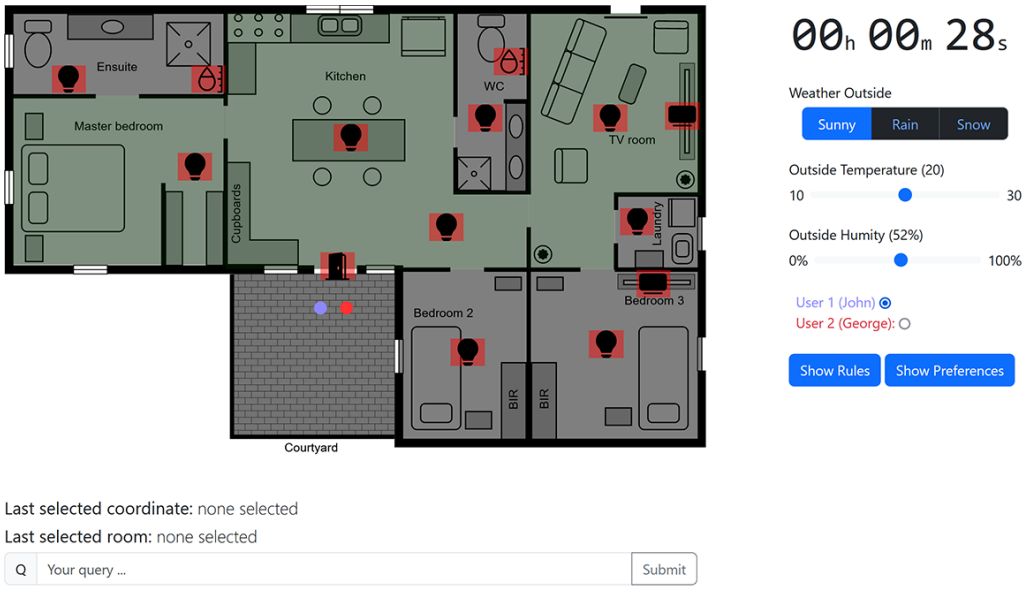

Figure 5.1: Preview of the Simulation Interface

One notable feature of the simulation interface is its ability to provide users with essential information about specific rooms equipped with HVAC systems. By right-clicking on a room with HVAC, a modal window promptly displays additional details such as the room's temperature and humidity levels. This enhances the user's understanding of the simulated environment and enables them to verify if their commands have been correctly processed.

Figure 5.2 provides a glimpse of this additional panel within the simulation interface, demonstrating its integration and functionality.

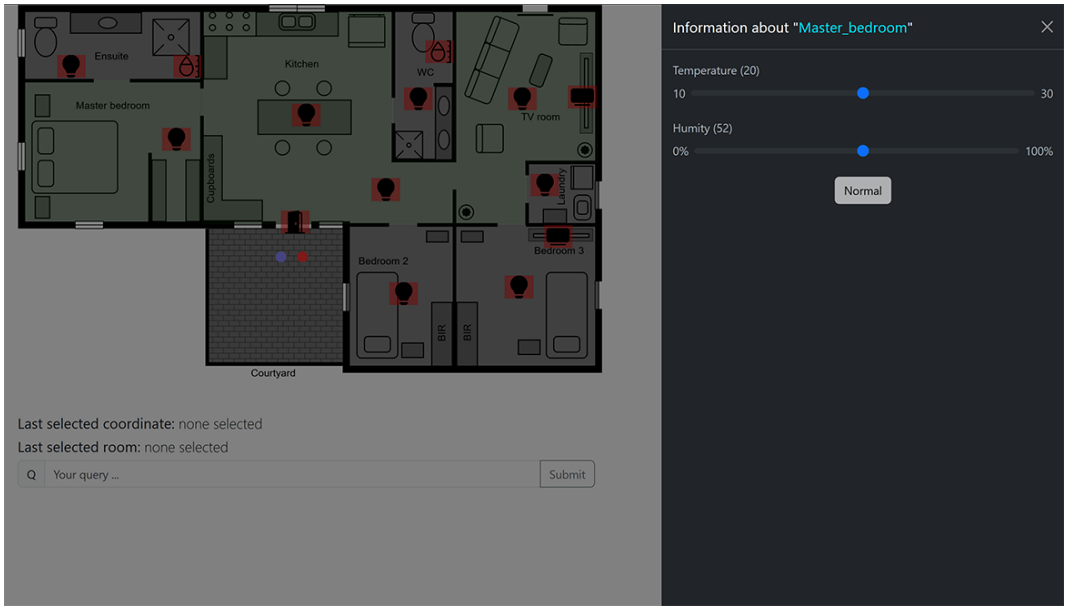

Figure 5.2: Preview of the Simulation Interface (HVAC data)

Additionally, the simulation interface supports two users, although the configuration allows

for the inclusion of more should it be necessary. Each user's click within the interface simulates their movement within the house, dynamically updating their position within the simulation as well as the awareness of the voice assistant. This update process is done through web-socket communication, which is how the assistant communicates with the front-end.

As such, the simulation interface serves as a practical solution to overcome the challenges associated with implementing a full-scale system for multi-device connection. Its web-based nature, interactive capabilities, and integration with the voice assistant establish an effective platform for users to visualize and interact with the smart home environment.

5.1.2 Assumptions

In order to accelerate our research and narrow our focus to the core aspects of contextual awareness, we have made several key assumptions and leveraged existing research and tools. By employing these simplifications, we aim to streamline our development process and allocate our efforts more efficiently.

To begin with, we have assumed the existence of an accurate speech-to-text module that could be easily employed in our system. Building upon the findings of various papers and existing tools[59][36][21][26][60][5][43], we recognize the existence of robust solutions that define reliable speech-to-text systems. Consequently, we treat the conversion of speech into text as a feasible component that we can assume to have already been implemented. This assumption allows us to concentrate on the subsequent stages of our research without developing a speech-to-text system from scratch. As such, the interface represented in Figure 5.1 and Figure 5.2 contain a text box where users can type their queries.

Additionally, we have made the assumption that a module that accurately determines the position of the user or speaker is already in place. This assumption is already expressed in Subsection 5.1.1, where our simulation tool provides users with the ability to effortlessly click on different areas within the house, updating their (X, Y) coordinates accordingly. By building upon the work of previous researchers who have dedicated their efforts to studying this component, we have deemed it unnecessary to allocate significant time to developing a position detection system ourselves. This assumption enables us to focus more directly on the primary objective of our research—contextual awareness.

Furthermore, we have assumed the existence of an already-implemented capability to identify the user who is speaking. This assumption is grounded in the research conducted by Huang *et al.* [30], who have explored the use of machine learning techniques to recognize users based on their unique gait patterns and walking styles. By leveraging this research, we can employ similar methodologies to identify the speaking user within our research framework. Alternatively, we acknowledge the possibility of exploring additional research avenues focused on voice recognition patterns to further enhance this capability.

By incorporating these assumptions and drawing upon existing research and tools, we are able to simplify our work and focus our attention on the fundamental aspects of our research, namely

contextual awareness. These strategic simplifications enable us to expedite our development process and maximize the impact of our efforts within the specified time frame.

5.1.3 Command Processing using WIT.ai

We leveraged the features provided by WIT.ai to handle user commands within our system. As discussed earlier, WIT.ai offers capabilities to identify the intention behind a user query, enabling us to narrow down the set of possible commands relevant to the query. Additionally, WIT.ai allows us to extract specific entities that play a vital role in parsing the query effectively.

The communication between our software and WIT.ai is facilitated through HTTP requests utilizing a dedicated API key. For each user query, WIT.ai attempts to classify it into one of the following predefined commands:

a) **`command_undo`**: This command is used to detect if the user intends to reverse or undo their previous command.

b) **`delete_rule`**: If detected, this command enables the user to delete a specific rule from the system.

c) **`query_state`**: This command is applied when the user seeks information about any device within the system. For example, a query like "What's the temperature in the Kitchen?"

d) **`set_device_state`**: Whenever a user attempts to directly define the state of a device, this command is used. Typical commands associated with this category include phrases like "Turn on the lights."

e) **`set_preference`**: This command is employed when a user declares a preference for temperature, humidity, or other device-related values.

f) **`set_rule`**: Used to set up a rule triggered by constraints such as time, weather conditions, temperature, or humidity.

Furthermore, WIT.ai performs query filtering to extract specific entities from user queries. This entity extraction process plays a crucial role in understanding the user's intent and enabling effective query parsing. You can visualize this on Figure 5.3. Given the query: "Turn on the lights at 2 AM if it's raining outside.", WIT.ai will automatically detect that the intent is "`set_rule`". In the same figure, we can also observe the query with colorized mark-ups representing the specific terms that match existing entity terms. For example, the words "Turn on" are associated to an entity of the type "ON" state. Similarly, the term "lights" is assigned to the entity type "LIGHT". The temporal information "at 2 AM" is automatically recognized by WIT.ai's built-in datetime entity. Finally, the phrase "raining outside" is identified as matching the "RAIN" entity from the "`Time_Conditions`" category.

The visualization exemplifies how WIT.ai effectively captures and assigns entities to their corresponding categories, which are leveraged by our processing module in order to better understand

the command handed by the user. In addition to it, some of the entities are custom, while others are automatically handled by the underlying system. Based on the same Figure (5.3), we can see that the only auto-filled entity was `wit/datetime`, as it is not included within a box and is not modifiable.

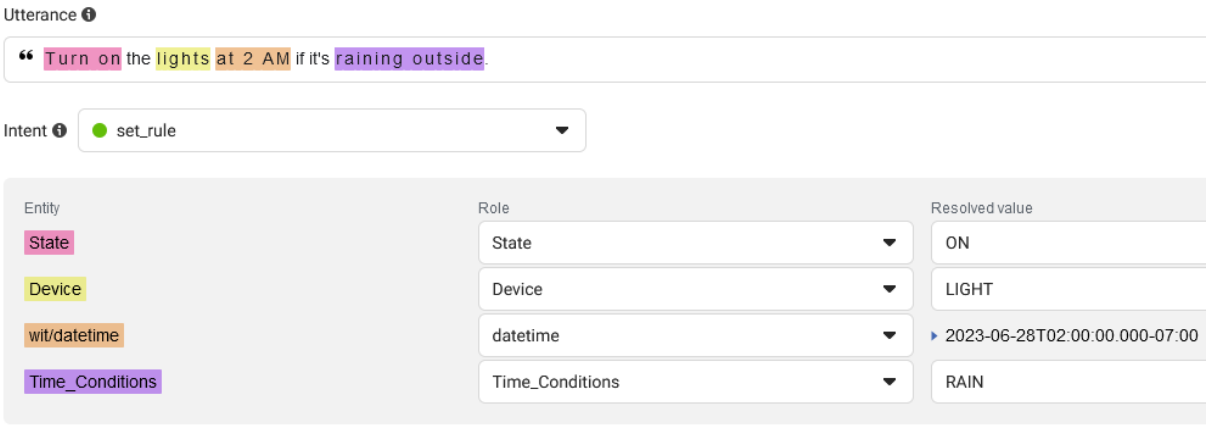

Figure 5.3: Preview of WIT.ai's query processing

The results returned by WIT.ai are formatted in JSON, a widely used data interchange format. This JSON response undergoes further parsing to extract relevant information required for the internal processing of our queries. Through this parsing process, the extracted data is assigned to specific variables, enabling us to navigate through the subsequent stages of the query processing pipeline.

In addition to identifying the intention behind a user query, the request handler also looks for specific entities to make informed decisions. These entities include:

`Action_Target`: This entity determines the target of the action specified in the query. For instance, in the query "Open the door for John," the name 'John' would be captured by this entity.

`Alternate_Counters`: This entity extends the default "number" entity provided by WIT.ai by supporting terms such as "all," "both," and other quantitative terms that are not strictly numerical.

`Device*`: This entity identifies the device that the query will act upon. It is typically mandatory, except for intentions like **command_undo** or **delete_rule**.

`Rooms`: An optional entity that specifies the location or room where the query should be applied. If this entity is not present, queries like "Turn on the lights" will rely on contextual awareness to determine the appropriate room.

`State`: This entity specifies the type of state that will be changed by the query. For example, utterances such as "Lock" or "Unlock" may refer to devices that support the specific state of

being locked or unlocked, such as doors. Therefore, when a user says "Turn on the lights," the 'turn on' phrase represents the state being referred to.

`Time_Conditions`: This entity is utilized for any components related to weather conditions, such as identifying if it is raining or snowing.

By examining these entities alongside the intention of the query, our system can make informed decisions and carry out the appropriate actions based on the user's input.

5.1.4 State-Agnostic Devices

Our system employs a state-agnostic approach to enhance the versatility and extensibility of devices. By utilizing a YAML configuration file, we define each device's characteristics and functionalities, allowing us to create a more cohesive connection between the front-end (e.g. simulation interface) and the back-end assistant logic.

The YAML configuration is used as a centralized repository for system details, including devices, rooms and their respective names. This approach ensures consistency and ease of maintenance throughout the system without the need to change the name of variables everywhere if we ever decide to change one specific feature.

Within these many configurations, we allow each device to contain a list of supported states. That is, each device specified in the configuration file is equipped with a 'state' attribute that dictates the supported states. For example, a device of the type "LIGHT" may have a "TOGGLE" state for basic on/off control and a "COLOR" state for color adjustment. This state-based mechanism enables dynamic control and differentiation of devices based on user queries.

When a user interacts with the system and references a device, the associated state information is utilized to determine the appropriate action. For examples, if a user tells the system to "turn on the light," the system will only identify devices that support the "TOGGLE" state. In a similar fashion, if a user instructs to "set the light to white," the system recognizes the user's intent to modify the color state and applies the change only to devices that support the "COLOR" state. These examples are illustrated in the code below:

```
devices:
  Kitchen:
    -
      xC: 76
      yC: 27.5
      width: 8
      height: 8
      type: LIGHT
      id: 3
      properties:
        - toggle
    -
      xC: 98
```

```
      yC: 48
      width: 8
      height: 8
      type: LIGHT
      id: 4
      properties:
        - toggle
        - color
```

Listing 5.1: Configuration example of devices

By decoupling states from specific devices, our implementation promotes flexibility and ease of integration. Adding advanced devices with additional options becomes seamless by updating the YAML configuration file. This design approach allows for future expansions and simplifies the incorporation of devices with enhanced functionalities, without imposing significant implementation/updating challenges.

5.2 Architecture

The system architecture, depicted in Figure 5.4, is essential to the operation of the Voice Assistant component. It consists of various crucial and isolated parts that communicate together to ensure a scalable and understandable system.

The front-end is the primary and input component of the architecture since it is where the user first interacts with the system. Through the front-end interface, the user may send requests and instructions that set off a cascade of system actions. When a user makes a query or changes any specific setting of the environment, the front-end uses a web-socket connection to relay that information to the Voice Assistant module, which will then operate on devices and update the status of the home or the weather, depending on the type of the quest.

As such, when a user sends a command, the "Websocket Listener" will be the first component to handle it. Its purpose is rather simplistic, and is implemented based on the websocket library of Python 3.9. It monitors for messages arriving from the user interface and relays them to the "Request Handler" accordingly. In another words, it mediates between the user and the rest of the system, which processes and acts on the user's requests.

Following the listener, the message is sent to the Request Handler which is responsible for determining the nature of the incoming user message. It analyzes the message to determine if it is a query, basic update to the user's location or even a change to the environment's settings outside the home (in Figure 5.1 that would correspond to a change of any of the settings on the right side panel). Based on the type of the message, the handler will decide what to do.

If the message is classified as a state change, such the user's location or an external weather variable, the Request Handler sends the information along to the House Manager. The House Manager is an instance that contains the location of the user, the state of the temperature outside and even a list of the devices in it. The handling and inclusion of all house-related data inside

the House Manager, is done so that we ensure centralization of responsibilities, which makes it easier to expand or remove features as we go along. In addition to this, the House Manager is also responsible for storing any existing rules that end up being defined in the conditions of the following paragraph. Every 30 seconds, or whenever a state change is requested, the Home Manager will loop through any existing rules, checking if the new state has triggered any of the existing rules. If that's the case, then the Request Handler will generate the appropriate commands and send them to the Command Manager.

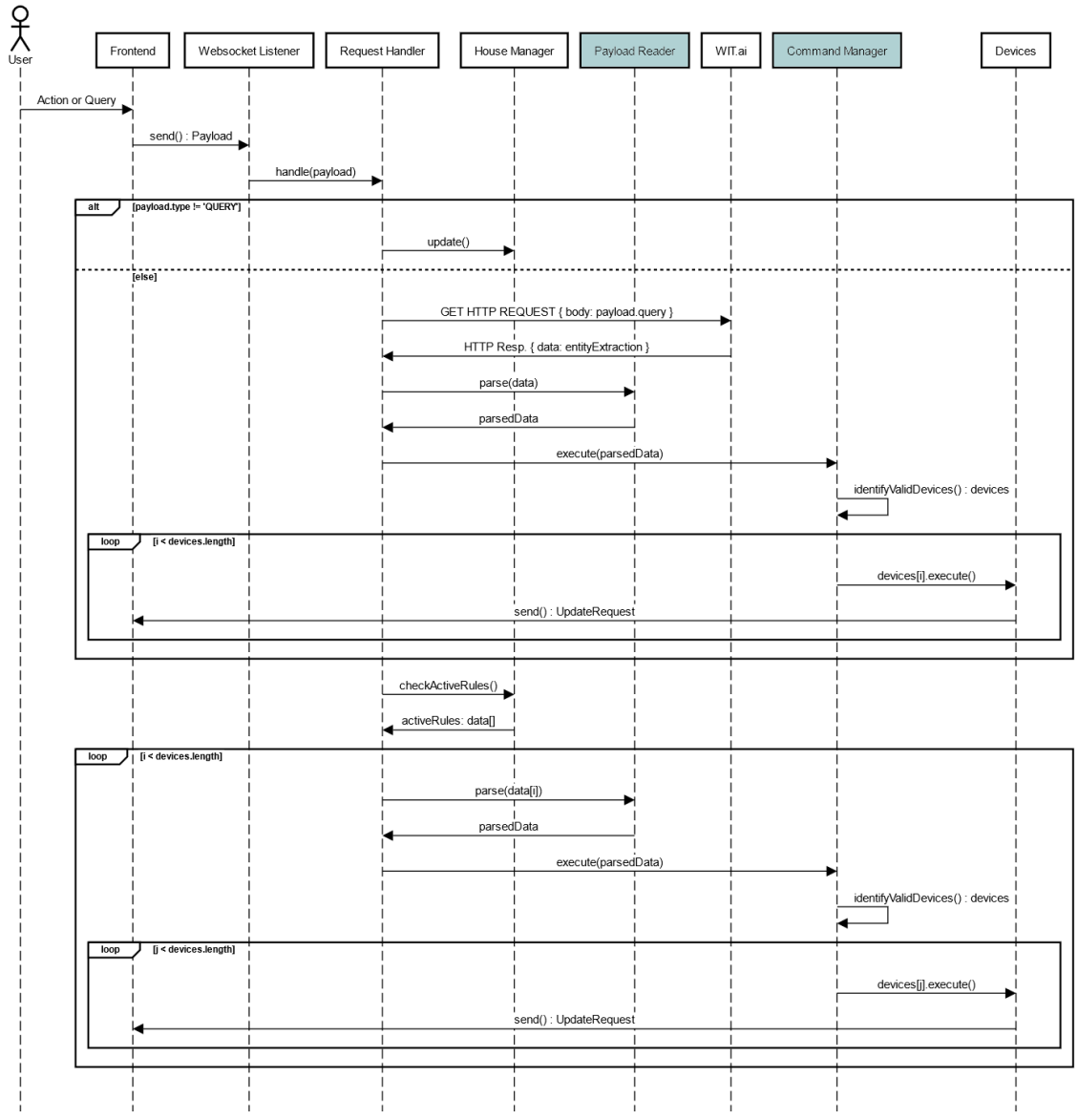

Figure 5.4: High-level sequence diagram of VA's processing

However, if the Request Handler determines that the incoming message is a user query that cannot be handled without a more specified interpretation, it will forward the message to WIT.ai. As mentioned before, WIT.ai will go through the user's text, identifying relevant entities and the intention of the request, which will help the handler to know what to do with the request.

After the extraction of the relevant entities, the handler will instantiate the correct command based on the intention that was identified. The construction of commands obeys the structure mentioned in Figure 5.5. It is up to the system to create the correct concrete command which will be executed by the manager. The command will then look for the relevant devices and run the appropriate changes on based on the states that need to be changed.

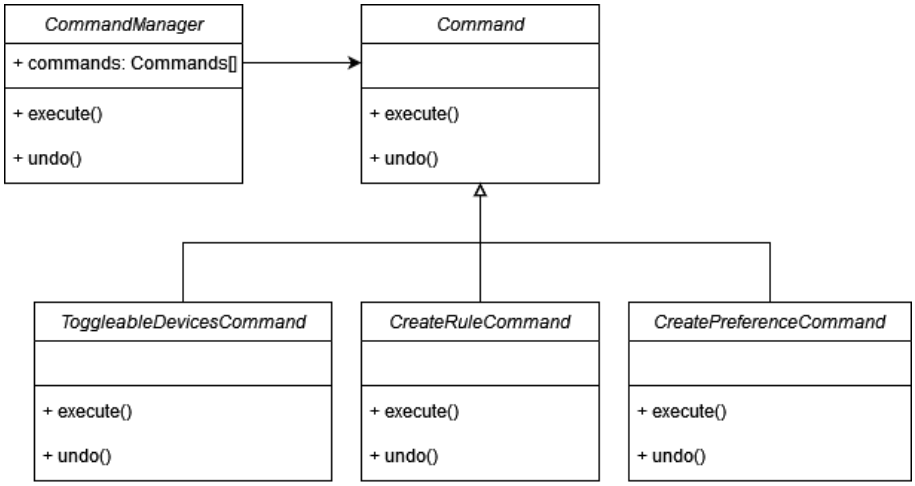

Figure 5.5: UML diagram showing the application of the Command pattern

After the command has been executed, it is then up to each component to publish their updated state. To achieve this, devices create a communication channel with the front-end after making the desired modifications in order to publish their updated state. Through this exchange of data, the front-end interface can then properly reflect the changed state after the command gets executed, giving users the necessary feedback and a graphical depiction of the modifications they made.

5.3 Processing Module

The central focus of our research is the voice assistant's processing module, which coordinates its many parts to provide a responsive and engaging user experience. These modules are visible on Figure 5.4 and are shaded in steel blue. It consists of two crucial components: the WIT.ai handler (visible as "Payload Reader" in Figure 5.4) and the command manager or processor, each relevant to describe how our assistant acts at different stages of the user query's processing.

The WIT.ai handler allows the voice assistant to understand user inquiries by extracting the relevant entities. To that purpose, it relies on WIT.ai to read and interpret user queries. The response comes in the form of a JSON format which must be parsed by the handler. As such, the handler here consists of a straightforward traversal of the response object that attempts to extract the necessary entities. However, it also performs more complex logic, such as determining the total number of devices involved in the query. If a user says "Turn on *all* the *two* lights", generally the most restrictive quantity is considered (in this case, "two").

The two most important entities entities that need to be extracted by the handler are the user intentions and target devices. If the handler fails to find these two entities, then the processing of the user's query is bound to fail. This happens because the system cannot comprehend a query without being able to understand its intention, and it can't know which devices are being mentioned. Simply saying "Turn on" isn't helpful, and not knowing if "Turn on the lights" is a rule or a command is also problematic.

After going through the handler, the extracted entities are sent to the command manager which takes charge of executing predefined sequences of actions based on the detected intentions. Functioning as a "orchestrator," it prioritizes tasks and decides which commands to carry out. The command manager functions on the basis of a list of commands kept in a Last In, First Out (LIFO) queue, enabling simple reversal or undo of actions by providing direct commands.

While most commands handled by the command manager adhere to a straightforward execution path (e.g. undo command), there are three commands in particular that demand special implementations. These are commands **d)**, **e)**, and **f)**. These three commands which were mentioned in detail in Subsection 5.1.3, involve complex logic and must also support the "undo" command. The rest of the commands are handled directly, without the need for elaborate management.

Among these commands, the most significant is **d) set_device_state**. It serves as the foundation for command **f) set_rule**, which relies on the successful execution of command **d)** when implementing a rule. Command **d)** uses a two-pronged method of processing. First, it identifies the appropriate command and the desired state or action to be performed. Then, it uses the query entities and external variables, such as the user's position, to find out which devices it will need to control. The implementation of the device detection is listed below:

```
Input: rooms, userid, deviceType, quantity, houseManager
Output: affectedDevices

init
affectedDevices <- []
if len(rooms) > 0 and 'House' not in rooms then
    for room in rooms do
        if room != 'User' then
            room <- room
        else
            room <- getUserRoom(userid)
        end if
        affectedDevices <- affectedDevices + findDevicesInsideRoom(deviceType, room
            )
    end for
    if quantity != -1 then
        affectedDevices <- affectedDevices[0:quantity]
    end if
else
    affectedDevices <- findClosestDevicesToUser(userid, deviceType, getUserRoom(
        userid))
```

```
20      if quantity != -1 then
21          affectedDevices[0:quantity]
22      end if
23      if len(affectedDevices) == 0 then
24          quantity <- 1 if quantity == -1 else quantity
25          affectedDevices <- findClosestDevicesToUser(userid, deviceType)[0:quantity]
26      end if
27      if len(affectedDevices) < quantity then
28          affectedDevices <- (affectedDevices + findClosestDevicesToUser(userid,
                deviceType, affectedDevices))[0:quantity]
29      end if
30  end if
31  return affectedDevices
```

This identification is based on a prioritized set of criteria. The first thing the command processor does is see if the user specifically mentions any particular rooms. If the user selects a room or multiple rooms, then only the devices within those mentioned spaces will be considered. When processing phrases like "this room," the command manager also takes into account a more abstract concept known as "House." For this particular concept, the system focuses on the user's position and the room where they are located. This method guarantees that the voice assistant will give preference to devices that are physically close to the user. An additional factor in determining which device to use is the order in which rooms are mentioned.

Furthermore, if the user defines a limit on the total number of devices to be affected by the command and the actual number of devices exceeds this limit, then the system will simply ignore the most distant devices for this matter. This technique allows for granular command execution control, letting users affect only the devices they care about without risking having simple commands affecting every device in the house.

Likewise, in scenarios where the command processor fails to detect any explicit mention of a room in the user's query, it shifts its focus to the room where the user is currently located. The processor then considers devices within that room, prioritizing those in close proximity to the user. Only when the required quantity of devices exceeds the total number available in the room, does the system explore devices in other rooms. For instance, if the user asks to control "four lights" but the current room only has three lights, the command processor will expand the search to include devices in other rooms.

In order to introduce flexibility into our system, when the required number of devices is not met, the command processor will look for other ways to find them. First, it starts searching for the closest devices in the whole house-space, going beyond the current room where the user is located. This guarantees that the system uses the most readily available and closest devices to satisfy the user's request. Additionally, the system treats unrestricted terms like "all" or "whole-sized" numbers as a considerably high number for the device limit. Through this, we ensure that when the user specifically states "all", then we will focus on the correct number of devices.

After going though these steps, the command processing phase returns a list of selected devices that will be affected by the specified command that was extracted in the WIT.ai handler. These device instances are responsible for making requests to the front end, detailing the exact modifications that are needed by a given device. By simulating a scenario in which a voice assistant may interact naturally with other devices, the simulation can test how well these tools perform when given instructions to act independently.

By employing this comprehensive processing module, our voice assistant achieves a robust and efficient interaction with users. The combination of the WIT.ai handler and the command manager enables seamless parsing, understanding, and execution of user queries, facilitating intuitive control over a wide range of devices in an intelligent and user-friendly manner.

5.3.1 Rule execution

An examination of the command manager, shown in 5.5, indicates the presence of three separate commands. Looking at the titles, it's clear that **CreateRuleCommand** is the one dedicated to the creation of new rules. However, their execution is not as simple as picking a command.

Rules are carried out in a two-stage procedure. First the handler will seek to identify the situation that triggered the rule. If it fails to detect them, then the rule is not created. When those prerequisites are met, the system proceeds to apply contextual awareness in real time, that is, in the moment when the rule is triggered. This method is an attempt to solve situations where ambiguity exists, such as when formulating a rule like "Turn on the lights at 2 AM.". When looking at this example and the various ways the assistant can serve this request, a unique difficulty emerges, highlighting the need for consistency. Specifically, a choice must be made on which devices to use when a certain rule's criteria are satisfied. This choice depends on whether the devices should consider the context of when the rule was defined or if we should consider the existing context at the moment the rule was triggered. In another words, if we were to create the rule "Turn on the lights at 2 AM." while inside the Kitchen and at 2 AM we happened to be in the Bedroom, the assistant needs to know whether it should pick the lights of the Kitchen (the previous context), or the lights of the Bedroom we are currently in (the new context). To solve this, we assume that unless a precise place is given for the execution, the assistant will always choose real-time contextual awareness rather than stored contextual-awareness.

In addition to this problem, we sought to simplify the code and cut down on potential complexities. When a rule is activated, the system automatically grabs the main query and sends it to the "ToggleableDevicesCommand". We use the House Manager to maintain a rules database and to execute them when all house-environment variables are triggered (e.g. time and weather).

5.3.2 NLP Limitations

We ran into a number of problems while using WIT.ai for the NLP part of our project. The fact that WIT.ai sorted entities by confidence score rather than how they appeared in the text was an enormous obstacle as maintaining uniformity when answering inquiries became more challenging.

A query such as "If it's 20 degrees outside, set the HVAC to 30 degrees." does not guarantee that the order of the temperature entities will match the order in which they occur in the query, so, therefore, "20 degrees" can show after "30 degrees" in the results, making it more difficult to ensure consistency. This discrepancy did not arise often, but it presented a problem as we wished to keep simple queries such as that one understandable and effective.

In addition to this issue, we saw that WIT.ai slowed down considerably during our end user experiment, which led to criticism about its excessive response times and impacted how our results were received.

In addition to these issues, WIT.ai also has trouble combining separate references to dates and times inside the same text. In the query "Tomorrow, turn on the lights at 2 AM," for example, WIT.ai creates two date objects: one for "Tomorrow" at midnight and another for "2 AM" on the present day. In addition, and as mentioned before, it was not helpful that these two dates could not be guaranteed to be in the proper order. To fix these problems, we moved the correction of these particular issues to Future work and implemented hard-coded solutions as a workaround. The solution would require a specialized NLP processing approach which was outside the scope of our work and would not let us keep up with the time-constraints.

In the beginning, we also used Google's Dialogflow and it showed encouraging early results. Unfortunately, we were unable to train the tool using our intended method. One problem was that it took too long to get training results, which slowed down the whole procedure. The training process was further complicated by the fact that Dialogflow's supplied entity names were determined by value rather than class. Finally, one of the biggest drawbacks was that distinguishing even the simplest queries like "Turn on the lights" and "Turn off the lights" led to many errors and inconsistencies, making us move to WIT.ai.

5.4 Conclusion

In this chapter we have shown and described the implementation of our small-scale voice assistant system inside a simulated smart space that uses contextual awareness to guess any relevant information in user queries. In order to achieve this, we have built a solid system for processing natural language and extracting entities by using Python and incorporating it into the WIT.AI platform. In addition to increasing the adaptability of devices, we made sure to isolate corresponding features in a way that would allow manufacturers or any other responsible entity to freely expand the system with more devices.

Throughout most of our time, our efforts centered on developing a simulation of a voice assistant aware of its surroundings, with real-time data of a smart-space. We made sure to implement it, so that the expansion into a rea-time system would be done smoothly and without needing critical changes.

Chapter 6

Experiments and Results

Contents

This chapter provides a comprehensive analysis of the experimentation approach used to assess the quality and usefulness of the developed assistant within the overall system. Additionally, we compare the assistant against existing solutions to validate the hypotheses outlined 4.3. The following sections outline the experimental steps conducted and present the findings in terms of relevant statistical analysis.

6.1 Experimental Setup

To evaluate the efficacy and efficiency of our system, a rigorous three-step methodology was implemented. Firstly, we carried out a comparative analysis with existing solutions, and used WIT.ai for this matter. By doing this comparison, we tried to analyse its capabilities and its usefulness in feature-offering.

In the second step, we extracted a set of queries from an existing dataset of smart-space related questions or requirements. By employing these queries in our voice assistant, we attempted to evaluate its comprehension and processing capabilities and how useful it was.

Finally, a controlled experiment was carried out, in which three users were given a set of six predetermined tasks to complete. The major goal of this research was to assess the effectiveness of our system in terms of three criteria. The first step in evaluating our system's effectiveness was to track how long it took to do various activities. Second, to evaluate the precision and consistency of

the system's answers, the number of errors that occurred throughout task execution was kept track of. Last but not least, participants' own impressions of the difficulties they encountered while completing the tasks were collected. We hoped that by looking at these metrics, we could get a good feel for the system's efficacy from every angle.

This three-stage approach gives a solid basis for evaluating the performance of our solutions. The system's strengths, weaknesses, and room for improvement can be better understood, as well as our initial hypothesis answered, through comparison to existing solutions, evaluation of comprehension abilities through diverse queries, and experimental study.

6.2 Feasibility Experiment

This research makes use of data collected by Danny *et al.* [55] as part of his efforts to verify a voice assistant's interpretive abilities. The queries in their dataset [56] include a wide variety of use cases and domains typical of a smart space, and their purpose is to evaluate the voice assistant's efficacy. This set of questions was hand-picked to exclude the possibility of bias introduced by user-generated queries during the evaluation of the system's efficacy. Depending on the user's needs, the queries can range from simple ones like: "When it is 7:00 today, turn on the bedroom lights," to more complex ones such as: "If there is a CO2 leak in the garage (air quality sensors), open the garage door and internal door, and trigger an alert through the sound system."

To conduct this analysis, we selected 47 queries at random from the dataset to apply it to our voice assistant. However, certain adjustments were needed to adapt these inquiries appropriately owing to the unique nature of our voice assistant and the simulated smart space it supports. An example of a query that we modified was: "Blinds inclination system based on outside light". While our technology does not provide direct assistance for this inquiry, our voice assistant does make use of external elements like weather and lighting. Additionally, although there is no built-in support for window-blind management in the system, this can be quickly remedied by adding the appropriate entities to the configuration file discussed in Section 5.1.4. So, although the system does not directly allow the query (because the blinds and luminance variable are missing), it can easily accept it by adding the necessary components.

In addition to this adjustment, we have grouped the different queries into specific groupings, which are: (1) Lightning Control, (2) Blinds Control, (3) Temperature Control, (4) Security and Safety, (5) Appliances and Devices control, (6) Garden Automation, (7) Miscellaneous. For each grouping, we indicate the number of queries in it and the number of queries that were supported by our voice assistant. Table 6.1 illustrates the results after our individual analysis of each query, and Figure 6.1 contains a Spider-web Diagram visualization of the same data.

Our system was found to provide the least amount of assistance for Group 5 (*i.e.*, Appliances and Devices Control) of the investigated groups, with fewer than 15% of their requests being supported. The biggest contributor to this outcome was the need for intercommunication across devices in this group, a feature that is not supported by our existing system. Requests like "Inform owner when products are on sale in a specific store" demonstrate the need for a more advanced

-	Group name	Queries	Supported	Ratio
1	Lighting Control	8	4	50.0 %
2	Blinds Control	5	3	60.0 %
3	Temperature Control	5	2	40.0 %
4	Security and Safety	7	3	42.9 %
5	Appliances and Devices Control	7	1	14.3 %
6	Garden Automation	4	3	75.0 %
7	Miscellaneous	11	3	27.3 %
Total		**47**	**19**	**40.4 %**

Table 6.1: Statistical Overview of the Dataset comparison

assistant with real-time data integration from e-commerce platforms and information beyond what can be provided by a smart-space manager.

Despite this restriction, we found that our voice assistant was able to answer about 40% of the inquiries in the dataset. We take this to be an important measure of the assistant's effectiveness within smart environments due to the high complexity and ambitious nature of these questions or needs.

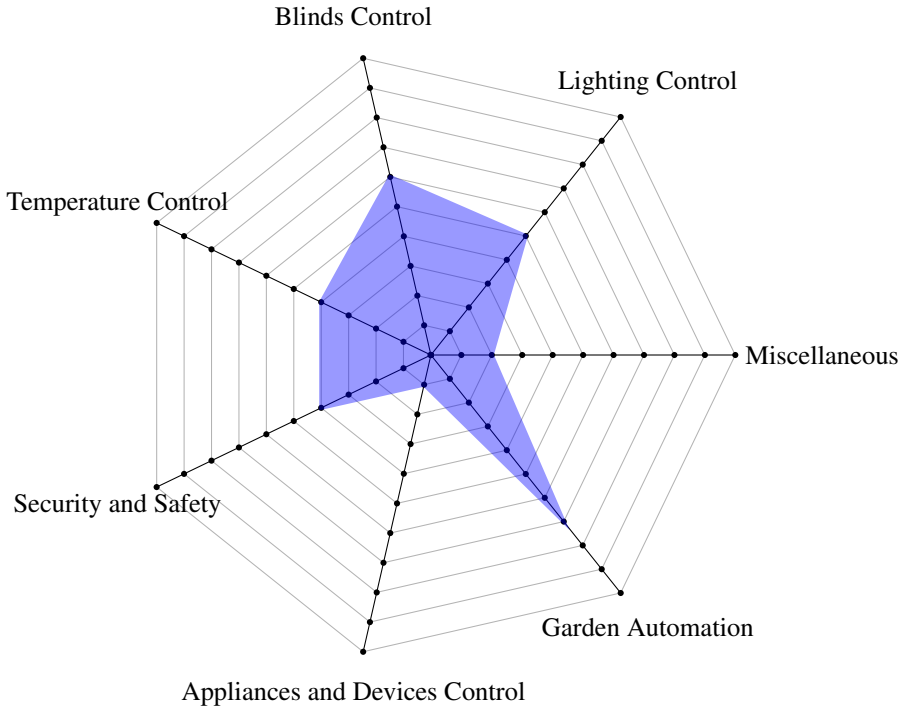

Figure 6.1: Spiderweb Diagram representation of the contents in Table 6.1

The results highlight the significance of understanding the system's existing capabilities and limits, especially in regards to inter-device communication and access to external real-time data. By being aware of these limitations, we may better plan for future improvements, such as better inter-device communication and integration with e-commerce systems. Our technology has the

potential to become an even more useful asset within smart environments as it develops and adapts to facilitate a broad variety of user interactions and meet complicated user demands.

6.3 End-user Experiments

In order to obtain feedback on how end-users would embrace our voice assistant and therefore analyze its usefulness in a real-time scenario, we created six tasks and asked 10 different users with varied levels of experience with smart spaces with writing the commands and watching the changes occur. This approach is similar to the ***Wizard of Oz*** [14] approach, which refers to a technique used in human-computer interaction research to simulate or prototype certain aspects of a system or interface before fully implementing them. This technique allows researchers to gather feedback and evaluate the effectiveness of a design without investing extensive resources in building a complete system. In our scenario, however, instead of having a human simulating the responses of the system we have a simulated smart space. As such, there's no actual smart space involved, and the user is simply interacting with a home space that exists solely within their screen.

Before initiating the feasibility experiment procedure, we first introduced each user to the tool, by show them the interface, explaining how they could interact with the space (*e.g.*, moving the user or sending in queries to the system). In addition to that, they would be instructed on how the experiment would go, along with the type of questions that could come up.

The simulated scenario is identical to the one in Figure 5.1, which consists of 8 rooms, 14 distinct devices and 2 users (John and George). There are 2 water heaters, one available in the Ensuite and the other in the WC room. There are also 2 TVs, each one in the TV room and Bedroom 3. Additionally, we have a front-door which may be locked or unlocked, and the interface will block the user from moving if the user tries to move into the house while the door is locked (the user may, however, exit the house even if it's locked). In addition to this, and similarly to Figure 5.2, the rooms shaded with a green color (*i.e.*, Master bedroom, Kitchen and the TV room) will have an interface that allows the user to see the HVAC settings, such as the temperature and humidity level.

With all these considerations in mind, each user was tasked with doing the following tasks:

T1: Position John in any location of the house and ask the system to turn on the lights before unlocking the door for George.

T2: Schedule the system to turn on the lights in the WC if it's raining outside, and at 8 PM proceed to turn all the lights off.

T3: Set your preference temperature to 15 degrees. Ask the system what the temperature is in the Master bedroom and then tell it to set the temperature.

T4: Lock the door for John and then try to put him in the house. Unlock the door for John again after 7 PM.

T5: Set the temperature of the Master bedroom to 24 degrees if the humidity outside is 20% and turn off the lights in the same case.

T6: Set both the lights in the Kitchen and the TV in the TV room on at 2 PM and apply the previous preference temperature to the Kitchen.

Besides the initial introduction, the only request made from us throughout the task execution was for users to speak naturally without any concerns for possible misunderstandings or errors in the system. We were also available to clarify any possible questions they had about the system throughout the task execution (*e.g.*, How to check the HVAC information).

6.3.1 Experiment Conditions

The experimental study was conducted in a face-to-face setting, with participants being tasked with completing the six previously mentioned tasks. Following their completion, a brief questionnaire was given to them in order to understand the user's opinion on the system's usability, and employing the System Usability Scale (SUS). As such, and as mentioned before, the study involved a group of ten participants, each possessing varying levels of experience with technologies with the specifics being described later on Section 6.3.3.

To establish a baseline and gain an understanding of anticipated completion times, several preruns were conducted before the formal experiment. These initial runs provided us with insights into the time expected for task completion. On average, participants proficient in using the tool were estimated to complete the tasks within an average of approximately 13 seconds. Notably, Task 4, which required multiple distinct actions, was the most time-consuming, with a maximum expected completion time of 24 seconds. In contrast, participants who were unfamiliar with the tool were projected to take approximately four times longer to complete the tasks. Individuals who happened to be slow-typers were expected to take even longer to complete their tasks.

6.3.2 Quantifiable and Verifiable Data

In order to evaluate the success of this experiment, we measured three critical factors for each task done: the amount of time (in seconds) the user took to complete the task, the number of mistakes that the user or the system made, and the subjective opinion of the user. The execution of the task was measured in seconds, and thus offered an objective measurement of the total amount of time required to do each activity. The amount of mistakes that occurred while the task was being carried out acted as an indicator of the user's correctness and competency in their ability to interact with the system, but also the ability of the system to handle more complex and different queries.

In addition, subjective user evaluation was gathered by utilizing a rating system that ranged from 1 to 5, with 1 denoting the lowest quality score and 5 indicating the highest. This assessment was intended to capture the user's opinion of the level of difficulty of the task as well as their overall level of satisfaction with the execution of the system.

As such, overall, these parameters were selected with great deliberation so that a detailed analysis of the experiment could be provided. The amount of time it took to carry out the task enabled an objective evaluation of the system's effectiveness and responsiveness, while the number of mistakes indicates how much our system needs to be improved in order to support a large variety of request styles. In addition, the subjective evaluation offered helpful insights into the user's subjective experience and gave a qualitative review of the system's overall quality as well as the level of user happiness.

6.3.3 User Profile and Statistics

Before discussing the results obtained from the experiment, we shall discuss relevant information about the users that participated in it. This demographic overview is part of a questionnaire that each user filled out after executing the six tasks successfully.

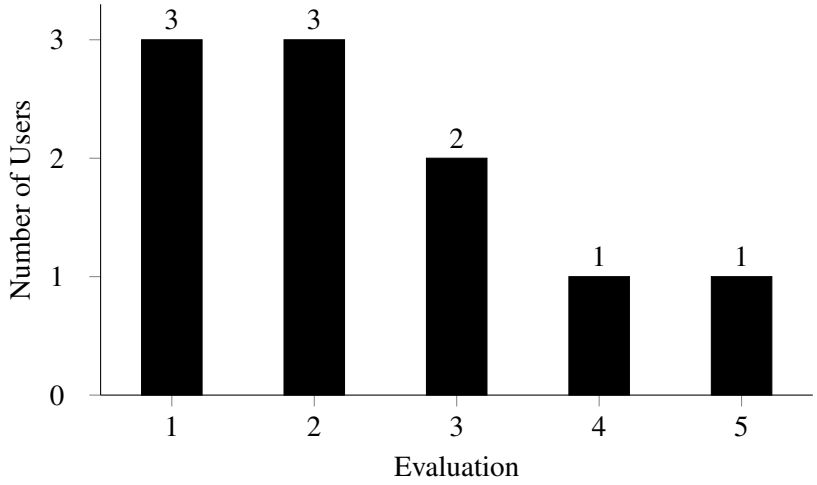

Figure 6.2: Users' Level of Experience with Smart Spaces (Scale: 1-5, with 5 indicating Very Experienced)

Starting from the graph in Figure 6.2, we can see that approximately 60% of the users (6) classify their own experience with smart spaces between 1 and 2. This in turn shows that most of our sample consists of standard users who do not possess any knowledge of how smart assistants work nor how to formulate queries in order for them to be understandable. From a certain point of view, this is an advantage, as it introduces unpredictability and variability in the style and structure of user queries, allowing us to measure how robust our assistant is. On the other hand, it forces us to be a little slower when explaining the features and to make a more careful introduction that is adjusted to the skill set of the user.

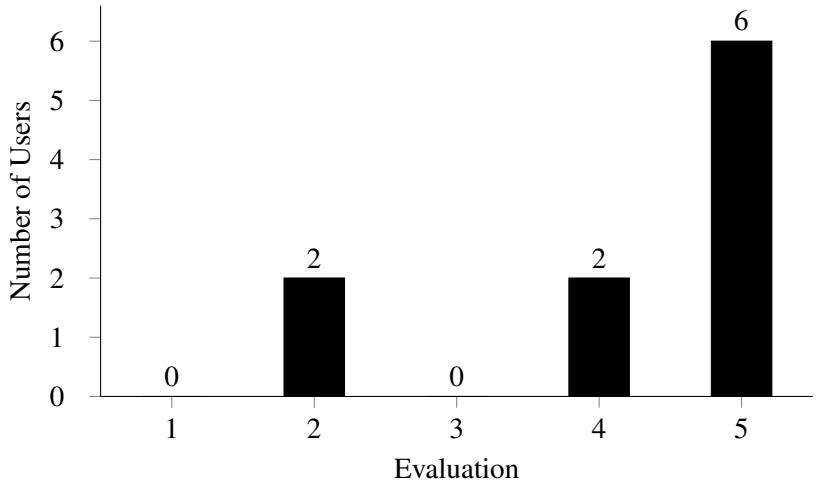

Figure 6.3: Representation of how comfortable users are using technology on a daily basis (Scale: 1-5, with 5 indicating Highly Comfortable)

A more curious result that is shown in Figure 6.3, where the same percentage of users has expressed to be comfortable using technology. In fact, 30% of the users that previously indicated they had a level of experience of 1-2 with smart spaces, classified their own comfort with technology as 5. This in turn shows that despite the growing popularity of smart spaces, users are still not entirely connected with the term and may only know very little about the subject. Alternatively, they may know about the term and what it consists of, but they may have never had the opportunity to own or be inside one.

Age Group	Number of Users
18–24	2
25–37	4
38+	4

Table 6.2: Age groups of respondents

Finally, from Table 6.2 we can see the age groups of our different users. The age distribution of our participants provides valuable insights into their technological familiarity and comfort levels. In addition to this, we observed interesting trends in the attitudes and behaviors of different age groups towards technology. Notably, the younger age groups, comprising individuals aged below 38, exhibited a significantly higher level of comfort with technology compared to the older age group (38+). This observation aligns with the widely acknowledged notion that younger individuals tend to be more tech-savvy and adaptable to new technologies.

In fact, the older age group (38+) predominantly consisted of participants who identified themselves as having either level 4 or level 2 technology proficiency. This indicates a more polarized distribution of technological comfort within the older age group. While some older participants exhibited a high level of familiarity and competence with technology (level 4), others reported a relatively lower proficiency level (level 2). Even so, only the oldest users (with ages 56 and 49) replied with level 2.

The contrasting technological comfort levels observed between the younger and older age groups highlight the importance of considering age as a significant factor in user experiences and preferences. These findings suggest that when designing and developing technological solutions, it is crucial to consider the diverse range of technological competencies and comfort levels among different age groups, which is already a widely accepted perspective in research.

6.3.4 Results

As mentioned previously in Section 6.3.2, we proceeded to measure the time that each user took in each task individually, along with the counting of mistakes made and a subjective evaluation of each individual task. This evaluation is usually based on the perspective of how difficult the task was. As such, to begin, in Table 6.3, we have listed the time that each user took for each task, in seconds. In order to keep the results stable and understandable, their analysis will be done using seconds and not the minute-second notation of this table.

User	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
A	41	82	54	46	117	74
B	46	160	49	106	78	85
C	66	81	114	67	82	62
D	56	90	75	72	79	68
E	118	197	57	119	93	104
F	83	119	61	97	92	110
G	49	123	89	61	63	72
H	51	103	46	55	59	65
I	127	177	182	298	174	116
J	94	111	81	65	122	58

Table 6.3: Time for task completion (in seconds)

In Table 6.4 we have listed all the relevant metrics of our results, including the minimum, maximum, the sum of all times, the mean, the average, the deviation and variance of the results obtained. Based on the results, we can easily perceive that variations from the mean are relatively high for some tasks. However, the one that exhibits the highest results is Task 4, with a variance of 5461.15 s^2. This result is likely to come from User ***I***, who took 4 minutes and 58 seconds (298 seconds) to complete his task. In comparison, the second highest result was 1 minute and 59 seconds (119 seconds). That's more than the double. However, this is not a unique case. Based on every other analysis, User ***I*** is a clear outlier in all tasks, since he was also the one who faced the most difficulties when dealing with the system. If we happened to remove User ***I*** from the statistical analysis, variance in Task 4 would drop down to: 621.52 s^2 and the standard deviation would be 24.93 s. This verification is also obtainable when viewing the Box Plot of this same result set, which is present in Figure 6.4. Every maximum value of the box plots is taken by User ***I***, except in Task 2 where User ***E*** manages to exceed User ***I*** by 20 seconds. Nonetheless, User ***I*** is still a visible outlier in that case and every other.

However, it's important to note that the presence of an outlier can put into question the validity of our results. We took a moment to analyze the nature of this outlier and we noted that this was

Task	Min	Max	$\sum$	$\bar{x}$	$\tilde{x}$	σ	σ^2
1	41.00	127.00	731.00	73.10	30.97	61.00	959.21
2	81.00	197.00	1243.00	124.00	115.00	40.60	1648.68
3	46.00	182.00	808.00	80.80	68.00	41.29	1704.84
4	46.00	298.00	986.00	98.60	69.50	73.90	5461.15
5	59.00	174.00	959.00	95.90	87.00	34.14	1165.88
6	58.00	116.00	814.00	81.40	73.00	21.22	450.49

Table 6.4: Task completion statistics (in seconds)

due to two factors. First, the user rated their own level of experience as '1' in the statistics of the Graph 6.2 which introduces a level of incompatibility with the system. Second, the user consistently tried to use more complex terms, such as joining multiple commands together, sometimes with major syntax mistakes which cooperated to a more challenging result. Although the user managed to complete all tasks successfully, they still gave a negative rating throughout the board. We judge these ratings based on lack of experience, however, it may still be the case that these results could have changed with a different sample and this user wouldn't be an outlier anymore.

In addition to the analysis done, we can conclude that the ranges of Min and Max are also extremely high. Again, this comes from User ***I*** and occasionally another user who takes longer than expected to finish a task. However, even without User ***I***, differences would still range between 20-40 seconds. The average completion time is also stable across most Tasks, with the values staying inside the range between 80 seconds to 98 seconds. However, Task 2 shows a larger average. This happens because Users are suddenly introduced to the rules, which involve a greater planning and a more elaborated approach rather than simple commands. However, after the second task, users quickly get used to the pattern and start taking less time to reply overall.

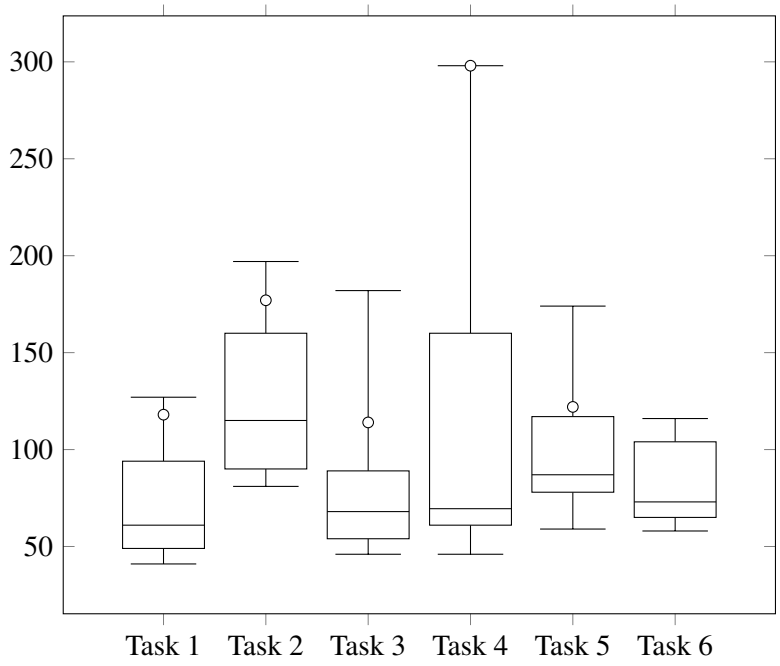

Figure 6.4: Time box plot for task completion (in seconds)

When we consider the mistakes made by each user in the system, it becomes interesting to analyze the relationship between these two variables. Figure 6.5 presents a table that lists the errors made by each user for each task. The numerical values in the table indicate the number of mistakes made, while the numbers of the last two rows show the total sum of mistakes and their respective averages. We can observe a clear pattern from the table, indicating that the number of mistakes tends to be proportional to the time taken. This correlation is further supported by a Pearson correlation coefficient of 0.9278 between the sums of the time taken and the sum of the mistakes, indicating a strong positive relationship between the total time taken and the overall number of mistakes.

Along with that, we can see that Task 2 had the highest number of mistakes among all the tasks. The reason for this, we assume to be because it's the first time the users are tasked with setting up rules, they tend to follow the specific request rather than coming up with a more natural question, which in turn leads to a higher number of mistakes. We can also note that this result with a high number of mistakes, coincides with the longest duration of 20 minutes and 43 seconds (1243 seconds). On the opposite hand, however, Task 1 had the fewest mistakes, despite not having the fastest completion time. In fact, Task 3 had the fastest completion time, but it did not have the fewest number of mistakes.

This leads to an important conclusion which states that although a longer duration may imply more mistakes, it is not frequently the sole contributing factor. Throughout the execution of the tasks, many users took additional time to clarify instructions, carefully write down their commands, and consider other factors that do not necessarily result in a higher rate of mistakes. These factors should be taken into account when analyzing the relationship between time taken and the occurrence of mistakes. We should, however, note that the difference of time between these two tasks (1 and 3) is minimal. It results in a difference of approximately 10.3 seconds between each user, which is perfectly normal in this simulated scenario and given the lack of experience of users which is represented in the graphic in Figure 6.2.

User	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6	$\sum$
A	0	1	0	0	3	2	6
B	0	1	0	1	1	1	4
C	0	1	1	0	0	0	2
D	0	0	0	0	1	0	1
E	1	2	1	1	0	1	6
F	1	0	0	0	0	1	2
G	0	1	1	1	0	0	3
H	0	0	0	0	0	0	0
I	1	3	2	2	4	1	13
J	0	1	0	0	0	1	2
$\sum$	3	10	5	5	9	7	39
$\bar{x}$	0.3	1.0	0.5	0.5	0.9	0.7	3.9

Table 6.5: Number of Mistakes per Task

As previously mentioned, apart from analyzing the time taken and the corresponding mistakes made in each task, we also sought to gather the users' subjective evaluations regarding the difficulty of the project. Table 6.6 presents the evaluations provided by each user for each task. The evaluation scale ranges from 1 to 5, where 1 indicates a perception of high difficulty, and 5 represents the opposite sentiment.

User	**Task 1**	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**	$\sum$	$\bar{x}$
A	5	4	4	5	2	1	21	3.50
B	5	3	5	5	3	3	24	4.00
C	5	4	3	5	4	5	26	4.33
D	5	4	4	4	3	4	24	4.00
E	4	3	3	2	4	3	19	3.17
F	5	4	4	3	4	3	23	3.83
G	5	3	4	3	3	4	22	3.67
H	5	4	4	4	4	4	25	4.17
I	2	2	2	1	1	2	10	1.67
J	5	4	4	4	2	5	24	4.00
$\bar{x}$	4.6	3.5	3.7	3.6	3	3.4	21.8	-

Table 6.6: Subjective User Evaluation. (Scale: 1-5, with 5 indicating the user felt the task was easy)

The obtained average evaluation score across all users is 3.68. This suggests that the users found the activities to be moderately difficult, falling somewhere in the middle between being too easy and too hard. This is in line with our original goals, since we intended to design activities that would not be too taxing on consumers while still giving enough complexity to give an accurate evaluation of our voice assistant's performance.

Intuitively, based on the information shown in the previous tables and graphs, it appears that User ***I***'s rating is consistent with our expectations. Specifically, User ***I*** gave the lowest possible rating, averaging 1.67 out of 5. In contrast, almost half of reviewers gave average scores higher than 3, which stands as the scale's midpoint.

This finding lends credence to the general evaluation that the rated performance of the tasks was satisfactory. However, as previously said, it is essential to recognize that while the majority of users considered the activities to be generally feasible, the lower score provided by User ***I*** provides insight into the difficulties faced by this specific user while engaging with the system. User ***I*** tended to oversimplify, missed key terminology, and showed a shallow understanding of the system's inner workings, all of which prevented him from making full use of the system without requiring constant reminders of its goals and operation. Finally, a careful examination of the users' ratings reveals interesting nuances about their experiences and viewpoints. It emphasizes the necessity of taking into account individual differences in user skill and comprehension when evaluating the success and appropriateness of the activities completed using our voice assistant.

In conclusion, the analysis of the data showed so far provides valuable insights into the effectiveness of our tasks and the voice assistant alike. Insofar, it allows us to conclude that **the tasks**

were feasible and accessible, and what the **average duration of task completion** is. Admittedly the time taken for time completion is too high compared to current solutions, however it's relevant to note that we are going for an approach that consists on writing out the query rather than spelling it out, therefore this increment of time is reasonable as not everyone has the same writing speed. This, therefore, means that we can't directly use time as a valuable metric for our system's efficiency, but we can use this data to state that the system can, indeed, achieve the tasks listed and employ contextual awareness with some occasional mistakes that were, most of the times, caused by typos when mentioning relevant entities (*e.g.*, "degres" instead of "degrees"). With this in mind, and in order to further evaluate the quality of our system and its usefulness, we resorted to quantitative and qualitative data from the questionnaires which is discussed in Section 6.4.

6.4 End-user Feedback

As mentioned before, in Section 6.3.3, users were asked to fill out a questionnaire after doing all the tasks. This allowed us to fill out the spots that we otherwise couldn't with just the data from the execution of the tasks and the perception of difficulty for each task. Having this data, allows us to see how the users embraced the voice assistant in general rather than the individual difficulty of each task.

As such, our questionnaire begins by asking users to rate, in a scale of 1 to 5, their perspective on the usefulness of the features offered by the voice assistant. The value 1 will mean that the users do not see the features as useful, and 5 will mean that the features are, in fact, useful. The results are shown on the graph of Figure 6.5, and we can see that that only one user replied with values 1 or 2, 2 users voted 3 and the remaining 7 users voted on values 4 - 5. The average response for each rating stands at 3.8, which represents a relatively good score and is close to the classification of 4, which maps to "Useful". The gathered results illustrate that the majority of users perceive the features as valuable, with a significant proportion attributing ratings of 4 or 5. The notable absence of low ratings (1 or 2) further affirms the users' overall positive perception of the usefulness of these features.

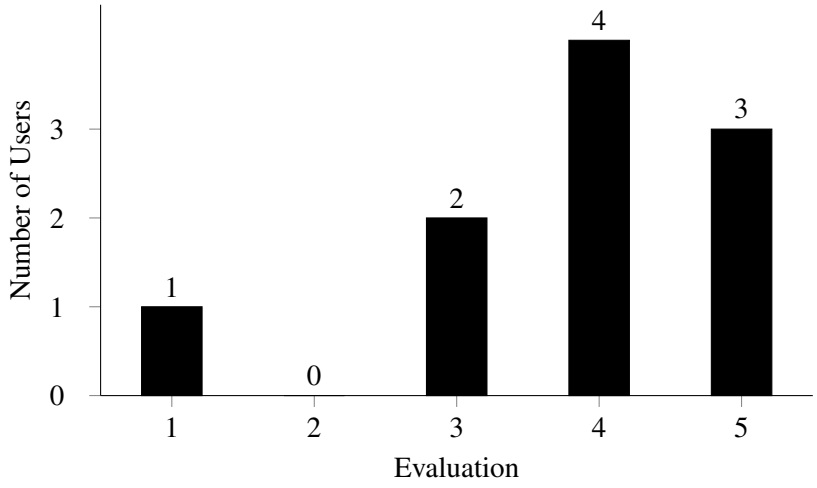

Figure 6.5: Classification of the usefulness of the features offered by the Voice Assistant (Scale: 1-5, with 5 indicating Very Useful)

Additionally, we conducted a survey to gather user feedback regarding various components of our Voice Assistant. Specifically, we sought their evaluations on the following aspects: **Ease of Use**, **Response Time**, **Utility**, and **Innovation**. The criteria for each component were defined as follows: **Ease of Use** pertained to the user's perception of how effortlessly they could comprehend and navigate the features, irrespective of the task complexity. **Response Time** referred to the duration it took for the voice assistant to provide a response or react to user queries. **Utility** reflected the users' opinions on the practical value and usefulness of the voice assistant in real-life scenarios. Lastly, **Innovation** focused on the uniqueness and distinctiveness of the voice assistant.

The survey results are illustrated in Figure 6.6. Each differently colored bar represents a distinct rating within the range of 1 to 5, indicating the number of users who voted for each rating category.

Analyzing the results, we observe that the feedback regarding the ease of use of the voice assistant is somewhat divided. Five users rated it with a 4, indicating that they found it easy to use. However, the remaining users expressed either difficulties or considered it to be an average experience in terms of usability.

Regarding response time, opinions were also varied, with the majority of votes leaning towards negative feedback (classifications 1 and 2). This outcome can be attributed to the limitations of WIT.ai's service during the evaluation period, which resulted in API request throttling. Consequently, users had to wait for several seconds, sometimes half a minute, before receiving a response.

In terms of utility, the feedback was relatively consistent, with the majority of users (7) giving a rating of 4. The remaining three votes were evenly distributed among ratings 2, 3, and 5.

Finally, when it comes to innovation, approximately half of the users agreed that the voice assistant demonstrated innovation, while the remaining users assigned a rating of 3, indicating a perception of it being a conventional and ordinary implementation.

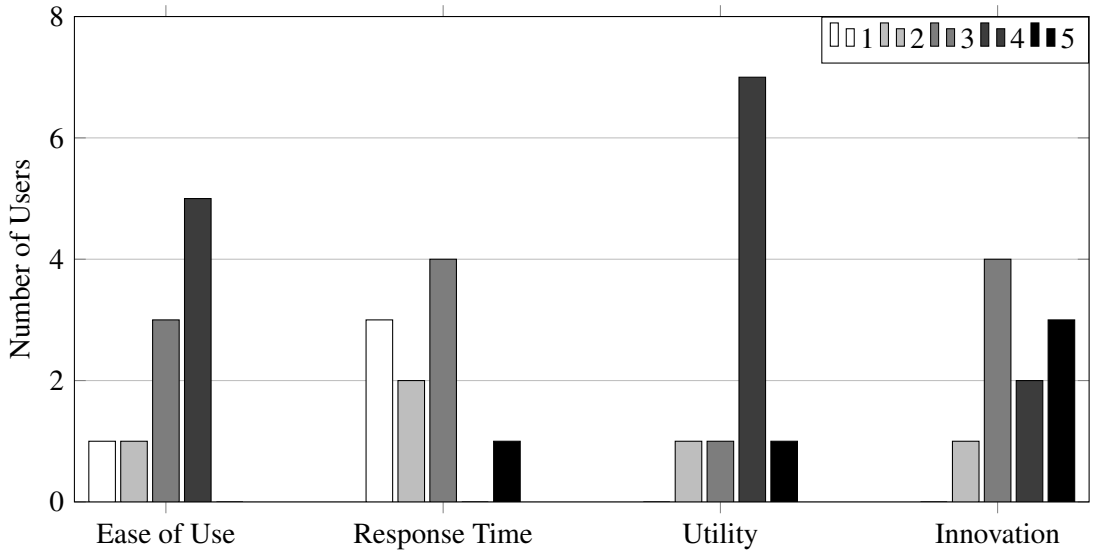

Figure 6.6: Evaluation of Voice Assistant components

Overall, the findings of this graph provide insights into user perceptions of the Voice Assistant's key components. While there were mixed opinions regarding ease of use and response time, the utility of the voice assistant received more favorable feedback. Additionally, the evaluation of innovation revealed a somewhat divided viewpoint among the participants.

Along other questions, we took the opportunity to inquire users about which contextually-aware feature they liked and disliked the most. We gave the following options for each user: **Ease of Use**, **User movement**, **User selection**, **Temperature**, **Rules**, **Preferences**, **Asking for information**. We allowed each user to select more than one feature, seeing as we wanted to if a particular set of features were likeable, rather than just individual ones. Out of every single feature, the most disliked one was the **Rules**, with 80% of the votes. Every other selection was evenly distributed with only 1 vote on each. On the contrast, **User movement** was the most likeable feature with a selection of 100%, next came **User Selection** with 80% and then **Door Locking System** with 70%.

In addition to this, we used the System Usability Scale (SUS) to conduct a thorough analysis of our voice assistant's usability. The SUS is a valid measure for evaluating usability because of its widespread application in studies and validation in a wide range of contexts. It provides a simple and effective method of gauging how well a system is received by its target audience, which can then be used to evaluate the system's usability.

As part of the SUS questionnaire, participants score 10 common-standard statements on a five-point Likert scale, from 1 (Strongly Disagree) to 5 (Strongly Agree). To make the calculations easier, these numbers are equated with the scores 0 through 4. Questions with an odd number focus on the system's advantages, while the questions with an even number focus on its disadvantages. This symmetrical method is useful for detecting random answers and capturing a holistic view of the system's usability.

The final SUS score is the result of two separate calculations. The positive elements of the system are represented by the X variable, which is the sum of the scores for the odd-numbered questions (Q1, Q3, Q5, Q7, and Q9) minus 5, yielding a 0 to 4 scale. The scores for all questions with even numbers (Q2, Q4, Q6, Q8, and Q10) are subtracted from 25, yielding a score that takes into account the question's negative wording, which is represented by variable Y.

After summing X and Y, we multiply the resulting number by 2.5 to get the SUS Score. This factorization normalizes the score, expanding its interpretative range from 0-40 to 0-100. Table 6.7 provides a detailed breakdown of the scores obtained from the participants.

User	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Score
A	5	1	5	2	5	1	4	1	2	5	77.5
B	4	2	4	2	4	1	4	1	4	3	77.5
C	5	1	4	2	5	1	5	1	5	2	92.5
D	3	2	4	3	4	1	5	2	4	3	72.5
E	3	2	4	2	4	2	2	2	3	4	60.0
F	1	4	2	4	2	4	2	4	1	5	17.5
G	3	2	3	3	3	2	3	3	3	3	55.0
H	4	1	5	1	5	2	5	1	4	1	92.5
I	4	2	4	2	5	2	4	1	4	4	75.0
J	4	2	5	2	4	3	5	2	5	4	75.0

Table 6.7: System Usability Scale results

The results are not organized by user, given that the results were done privately and at different times. In any case, we can see that only three users failed to give a feedback that results in a score higher than 68. The average of all results is 69.5, staying above the 68 average, which means that our assistant's usability is higher than the average. The value also stands between the levels of marginal usability (50-69) and good usability. Similarly to previous results, one user assigned a really low score of 17.5, which we assume to be the same outlier as in previous cases. If we chose to ignore that result, our SUS score average would become 75.3.

These insights from user feedback shed light on the potential acceptance and adoption of our voice assistant in a real-life setting. In addition to these quantitative data, users wrote some additional feedback. When asked to give their opinion, one common wish expressed by multiple users is the desire for more natural language instructions, without the requirement to use specific keywords or structured phrases. They appreciate the concept of simulating an intelligent environment into a single view, which could be extended to interact with a smart environment through sensors. However, they find the platform lacking in flexibility for defining rules and preferences, and note that there are occasional instances of lag.

As such, while some users appreciate the overall user experience, they express dissatisfaction with the response time of the system. They appreciate the smooth user flow in navigating the platform but struggle with the requirement of using specific words and phrases, finding it challenging and limiting. They suggest that the system should support more diverse interaction methods and offer greater flexibility in structuring sentences.

6.5 Comparative Evaluation

To conduct a comparative analysis between our solution and existing alternatives, we utilized the queries posed by users throughout Tasks 1 to 6, including any mistakes made. We executed these queries on both our system and the WIT.ai platform to determine if each system was capable of providing support. The queries were categorized according to their respective tasks, yielding the results presented in Table 6.8.

When evaluating the support offered by WIT.ai, we employed specific criteria based on the system's ability to provide an appropriate response, considering the available information. For example, queries such as "Turn on the lights" may be insufficient for WIT.ai due to its lack of contextual awareness necessary to address the problem. In contrast, our tool has the capability to handle such a command by considering the user's position.

Task	Queries	Our tool Support	WIT.ai Support
1	23	20	14
2	30	22	16
3	35	30	16
4	35	20	20
5	29	20	16
6	36	30	15
$\sum$	178	142	97

Table 6.8: Comparative Evolution with WIT.ai

An analysis of the information presented in the table reveals that utilizing WIT.ai alone, relying solely on its entities, would enable us to handle approximately 54% of the queries generated. However, by incorporating our system, this percentage increases to approximately 80%. It is important to note that the non-supported queries, which account for the remaining 20%, primarily consist of typographical errors that hinder WIT.ai's ability to detect relevant entities.

For example, consider the query: "Set the temperature to 10 degres." In cases where no preference is specified, our system would also be unable to handle it due to the syntax error in the word "degres" seeing that WIT.ai can't detect the error and therefore not detect the entity. Thus, if WIT.ai fails to detect an entity, our system won't detect it either, as it depends on WIT.ai's entity detection to make a decision.

Overall, these findings indicate that our system significantly improves query handling capabilities compared to using WIT.ai alone, increasing the supported query percentage from 54% to 80%. However, it is crucial to recognize that both systems are susceptible to the impact of typographical errors, which can hinder their entity detection abilities.

6.6 Threats to validity

During the evaluation process, it is important to acknowledge the potential presence of limitations that could impact the validity and precision of our conclusions. The following problems have been

identified:

- **Sampling Error**: Our inquiry involved a relatively small sample size of only 10 users, each with varying levels of technology experience. It is crucial to recognize that if a different sample were selected, the results obtained could potentially differ from those we obtained. Therefore, the ability to generalize our findings may be limited.
- **Instrumentation**: As highlighted in Section 6.4, certain users experienced processing delays due to the ongoing throttling issue with WIT.ai. These delays could have been influenced by factors such as the specific times at which the experiments were conducted. It is important to note that retesting these users to mitigate the impact of the throttling issue could introduce another potential threat, namely the *Testing* threat, where users would already be familiar with the tasks and potentially alter their behavior, thereby significantly affecting the overall variables under evaluation.
- **Selection Bias**: Most users consisted of individuals with close relationships, such as friends and family. Thus, the conclusions may not accurately represent the opinions of individuals who lack any personal connection to the developer, potentially introducing a higher degree of partiality.

Fixing these problems could be solved by introducing more users and doing all the experiments at the same time, under the same conditions.

6.7 Summary

In this chapter, we presented and analyzed the data at hand, offering a thorough evaluation of the data we acquired. Our investigation of the constituent parts and connections between the data and our findings has allowed us to evaluate the voice assistant's potential to serve its intended function and to answer our initially established hypothesis.

While doing our analysis, we looked closely at the responses of the 10 users who took part so that we could better understand their viewpoints and experiences, and to validate the usability and capability of our voice assistant. Prior to this, we used a large dataset of queries and smart-space situations in order to check how many of them were supported by our system. At the end, we also compared our voice assistant's performance to that of other tools like WIT.ai.

With this, we attempted to provide detailed information about the efficacy and potential of our voice assistant based on the acquired data.

Chapter 7

Conclusions

Contents

This chapter provides a comprehensive analysis of the experimentation approach used to assess the quality and usefulness of the developed assistant within the overall system. Additionally, we analise the results and compile all the information into one single section allowing us to formulate a final conclusion on our initially proposed hypothesis and research questions. We also introduce possible future work in order to guide our research and help develop a more elaborated assistant.

7.1 Conclusions

The Internet of Things (IoT) paradigm has experienced substantial growth in recent years, fueled by the proliferation of IoT devices and their ever-expanding computational capabilities. This trend has led to the development of increasingly complex IoT systems, posing significant challenges for their implementation. Additionally, the growing complexity presents a dilemma for users who find it increasingly difficult to manage the multitude of interconnected devices within their smart spaces. As users aspire to define multiple rules for various devices and occasions, the complexity factor becomes a hindrance, impeding the execution of their desired actions.

To gain deeper insights into the current state of research in this field, we conducted a systematic literature review, focusing on 12 different research projects, as elaborated in Subsection 3.2.1. Our analysis revealed a diverse landscape of solutions, with varying levels of accuracy. While some projects achieved average accuracy results, others lacked comprehensive examination, leaving room for further exploration and definitive conclusions. Notably, we found that most projects fell short in addressing the concept of a contextually aware voice assistant, often adopting simplistic approaches. Many projects solely concentrated on specific aspects of contextual awareness, such as humidity control or a specific case of door permission control. This gap in research inspired our work, which aimed to take a holistic approach to contextual awareness while incorporating the

capabilities of a virtual assistant. As such, we defined three research questions (*cf.* Section 4.4, p. 39), which served as the foundation of our work.

We addressed these problems in Section 5.1, where we introduced our implementation of a voice assistant that goes beyond conventional voice-only interactions by incorporating a visual interface. Our implementation is built on certain assumptions, assuming the availability of an existing tool for converting speech to text and accurate identification of the user's location, including the extraction of the user's specific room.

In Chapter 6 we conducted three different types of evaluations, including two experiments and one comparison. We conducted these experiments in order to: (1) validate our hypothesis, (2) assess the usability of our tool, (3) identify potential key-points where we could improve our tool.

The *feasibility experiment* performed (*cf.* Section 6.2, p. 56) allows us to compare our system with an existing definition of possible requirements / queries done inside a smartspace. This in turn gives us an overview of how many types of varied queries and features our voice assistant can support.

The *end user experiments* that we performed (*cf.* Section 6.3, p. 58) allowed us to see the quality of the tasks we assigned and to evaluate the usability of the system. The usability was measured using SUS, and the results were positive, with our average standing above the average of other research works where the average was 68. Throughout most of our experiment we noticed a user who was a clear negative outlier. Removing them would lead to higher and more consistent scores, but as discussed before, the user puts into question the validity of our results, and is worth further consideration. Including a bigger sample would probably lead to better result and help us avoid bias in our small dataset.

The *comparative evaluation* that we performed (*cf.* Section 6.4, p. 66) allowed us to compare our system with WIT.ai. This comparison was done based on the extraction of entities done by WIT.ai and by checking if the entities alone were enough to infer the complete command. In total, our system outmatched WIT.ai by over 20%, revealing a positive result.

Based on these experiments and results, we managed to answer the research questions proposed in Section 4.4. In addition to this, we managed to answer our initially proposed hypothesis. We confirmed **RQ1**, by developing an assistant that indeed makes use of various contextual aware traits, such as temperature, humidity, user position, weather and even identification of the user. We managed to partially confirm **RQ2**, as our system tolerates complex queries and terms, however, due to limitations with WIT.ai the system can't deal with overly complex and elaborated commands, and it can't deal with AND or OR operators inside a sentence. Finally, **RQ3** can be answered positively as the voice assistant uses contextual awareness to complete information, however, it does not have the ability to predict user behavior based on actions and customary behavior. The research questions listed, had previously led to the definition of our hypothesis (*cf.* Section 4.3, p. 39):

> *"In smart spaces, it is possible to empower virtual assistants with contextual awareness based on previous and current knowledge so that complex requests require fewer interactions and less information exchange."*

Having answered the research questions we can confirm that the hypothesis is valid and that it is indeed possible to use contextual awareness to reduce the interactions and exchange of information. **RQ1** guarantees contextual awareness based on sensor data, **RQ2** makes it possible for the system to handle complex queries that omit information, leading to fewer interactions and **RQ3** further shows this by dictating that the contextual data should be used to fulfill cases of ambiguity in a user query or when there's no specification from the user's end.

However, despite the confirmation of these research questions, we failed to achieve the point 1. and 3. of the desiderata as these would over-extend the complexity of our work and would force the introduction of additional methods to understand aggregation of clauses and the study of all requests in order to identify patterns inside a dataset. We set these as components for future work, available in Section 7.2.

In summary, our implementation of the voice assistant represents a significant step forward, showcasing its potential and affirming the viability of our approach, and confirming our initial hypothesis. While we suggest further enhancements and steps to beat our limitations, our system exhibits promising capabilities, highlighting its relevance and usability within the context of smart spaces and confirming our hypothesis, by empowering virtual assistants with contextual-awareness and requiring fewer interactions. Our system employs previous knowledge by remembering rules, preferences and other aspects of the past in order to improve the present time, even though, admittedly, it does not predict behavior based on the analysis of user behavior.

7.2 Future Work

The previous work we have done contains various limitations which have already been listed in Section 5.3.2 and mentioned briefly in Chapters 5 and 6, and in this section we take the opportunity to elaborate on possible future work that aims to tackle these limitations and to improve our system with additional systems.

Our first improvement suggests the creation of **Specialized NLP module**. As mentioned in Section 5.3.2 our voice assistant displays limitations due to the incorporation of natural language interfaces that are meant for more general purpose scenarios, such as support bots. We therefore propose a specialized module that is customizable and can handle the variety of queries that users give them. It could handle the extraction of entities in a more accurate format without forcing the *request parser* module to make hardcoded adjustments based on known queries.

To further optimize user experience, we also suggest the **addition of more devices and features**. The support of more devices and contextual variables would be an interesting topic. Using Danny's dataset as a basis, we could extract more devices and entities such as: smartphones, swimming pools and coffee machines. In addition to this, we could support rules that depend on the state of multiple devices.

We would also like to **implement STT technology** into the project. As noted in Chapter 6, the answer times were relatively high due the nature of the experiment and the tool. It would be

interesting to watch users setting up rules and giving commands with their voice and checking the time difference.

We also think that **Third-Party Services** could further improve user experience. Integrating popular third-party services, such as weather forecasts, news updates, social media platforms, ride-sharing services and others with our voice assistant instead of relying uniquely on the interface for these settings would be an interesting experience. As per the research done in Section 3.2.1, bringing contextual awareness into this integration would be a creative and ambitious approach compared to existing works.

Finally, reaching out to **ML (Machine Learning)** methods in order to detect patterns of usage and habits would allow us to answer point 3. of our desiderata (*cf.* Section 4.2, p. 38). In addition to this, it would make the assistant much more convenient as it would give another way for users to repeat commands without needing to create rules. Additionally, using **ML** would allow us to detect individual segments of sentences [35].

By addressing these research directions, we predict that these could significantly improve the system's performance, usability, and overall user satisfaction. These advancements have the potential to expand the tool's functionality, making it more adaptable to user needs and enhancing its contextual awareness in smart spaces.

References

[1] Agnar Aamodt. Case-based reasoning-an introduction. *Expert Systems With Applications - ESWA*, 6, 01 1993.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In Hans-W. Gellersen, editor, *Handheld and Ubiquitous Computing*, pages 304–307, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[3] Adobe. Voice trends and statistics: What designers need to know about the new tech boom, 2019. Available at https://blog.adobe.com/en/2019/07/22/voice-assistant-statistics-trends-2019#gs.17vx4r, last accessed in January 2023.

[4] Sakib Ahmed, Debashish Paul, Rubaiya Masnun, Minhaz Uddin Ahmed Shanto, and Tanjila Farah. Smart home shield and automation system using facebook messenger chatbot. In *2020 IEEE Region 10 Symposium (TENSYMP)*, pages 1791–1794, 2020.

[5] Folio3 AI. Converse smartly. Available at https://www.folio3.ai/converse-smartly-try-now/, last accessed in May 2023.

[6] George Alexakis, Spyros Panagiotakis, Alexander Fragkakis, Evangelos Markakis, and Kostas Vassilakis. Control of smart home operations using natural language processing, voice recognition and iot technologies in a multi-tier architecture. *Designs*, 3(3), 2019.

[7] AppMaster. What is visual programming and how does it work? Last accessed in January, 2023. Available at https://appmaster.io/blog/what-is-visual-programming-and-how-does-it-work.

[8] Jonas Austerjost, Marc Porr, Noah Riedel, Dominik Geier, Thomas Becker, Thomas Scheper, Daniel Marquard, Patrick Lindner, and Sascha Beutel. Introducing a virtual assistant to the lab: A voice user interface for the intuitive control of laboratory instruments. *SLAS Technology*, 23(5):476–482, 2018. Special Issue: The Internet of Things in the Life Sciences Laboratory.

[9] Cyril Joe Baby, Faizan Ayyub Khan, and J. N. Swathi. Home automation using iot and a chatbot using natural language processing. In *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, pages 1–6, 2017.

[10] Sumathi Balakrishnan, Hemalata Vasudavan, and Raja Kumar Murugesan. Smart home technologies: A preliminary review. In *Proceedings of the 6th International Conference on Information Technology: IoT and Smart City*, ICIT 2018, page 120–127, New York, NY, USA, 2018. Association for Computing Machinery.

[11] D. Buhalis and R. Leung. Smart hospitality – interconnectivity and interoperability towards an ecosystem, 2018. International Journal of Hospitality Management, Vol. 71, pp. 41-50.

[12] P. Christen C. Perera, A. Zaslavsky and D. Georgakopoulos. Context aware computing for the internet of things: A survey.

[13] European Commission. Report on consumer internet of things sector inquiry, 2022. Available at https://ec.europa.eu/commission/presscorner/detail/en/ip_22_402, last accessed in June 2023.

[14] N. Dahlbäck, A. Jönsson, and L. Ahrenberg. Wizard of oz studies - why and how. *Knowledge-Based Systems*, 6(4):258 – 266, 1993. Cited by: 417; All Open Access, Green Open Access.

[15] Joao Pedro Dias, Joao Pascoal Faria, and Hugo Sereno Ferreira. A reactive and model-based approach for developing internet-of-things systems. In *2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pages 276–281. IEEE, 2018.

[16] Joao Pedro Dias, André Lago, and Hugo Sereno Ferreira. Conversational interface for managing non-trivial internet-of-things systems. In *Proceedings of the 20th International Conference on Computational Science*, pages 27–36. Springer, 2020.

[17] Joao Pedro Dias, Bruno Lima, Joao Pascoal Faria, André Restivo, and Hugo Sereno Ferreira. Visual self-healing modelling for reliable internet-of-things systems. In *International Conference on Computational Science*, pages 357–370. Springer International Publishing Cham, 2020.

[18] Joao Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Empowering visual internet-of-things mashups with self-healing capabilities. In *2021 IEEE/ACM 3rd International Workshop on Software Engineering Research and Practices for the IoT (SERP4IoT)*, pages 44–51. IEEE, 2021.

[19] Joao Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Designing and constructing internet-of-things systems: An overview of the ecosystem. *Internet of Things*, 19:100529, 2022.

[20] Joao Pedro Matos Teixeira Dias. Increasing the dependability of internet-of-things systems in the context of end-user development environments, 2022.

[21] A. Elakkiya, K. Jaya Surya, Konduru Venkatesh, and S. Aakash. Implementation of speech to text conversion using hidden markov model. In *2022 6th International Conference on Electronics, Communication and Aerospace Technology*, pages 359–363, 2022.

[22] Electronics and B. I. T. S. G. (n.d.). Lidar Robotics Club (ERC). Lidar - erc handbook, 2022. Available at https://erc-bpgc.github.io/handbook/electronics/Sensors/lidar/ , last accessed in January, 2023.

[23] Simone Gallo and Fabio Paterno. A conversational agent for creating flexible daily automation. In *Proceedings of the 2022 International Conference on Advanced Visual Interfaces*, AVI 2022, New York, NY, USA, 2022. Association for Computing Machinery.

[24] J. Antonio Garcia Macias, Jorge Alvarez-Lozano, Paul Estrada, and Edgardo Aviles Lopez. Browsing the internet of things with sentient visors. *Computer*, 44(5):46–52, 2011.

[25] John Gero and Udo Kannengiesser. The situated function-behaviour-structure framework. *Design Studies*, 25:373–391, 07 2004.

[26] Yogita H. Ghadage and Sushama D. Shelke. Speech to text conversion for multilingual languages. In *2016 International Conference on Communication and Signal Processing (ICCSP)*, pages 0236–0240, 2016.

[27] Arindam Giri, Subrata Dutta, Sarmistha Neogy, Keshav Dahal, and Zeeshan Pervez. Internet of things (iot): A survey on architecture, enabling technologies, applications and challenges. In *Proceedings of the 1st International Conference on Internet of Things and Machine Learning*, IML '17, New York, NY, USA, 2017. Association for Computing Machinery.

[28] Google Cloud. Dialogflow's documentation. Available at https://cloud.google.com/dialogflow/es/docs/fulfillment-overview , last accessed January, 2023.

[29] Matthew Hoy. Alexa, siri, cortana, and more: An introduction to voice assistants. *Medical Reference Services Quarterly*, 37:81–88, 01 2018.

[30] Long Huang and Chen Wang. Walkid: Towards context awareness of smart home by identifying walking sounds. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6, 2020.

[31] Maksym Ketsmur, António Teixeira, Nuno Almeida, Samuel Silva, and Mário Rodrigues. Conversational assistant for an accessible smart home: Proof-of-concept for portuguese. In *Proceedings of the 8th International Conference on Software Development and Technologies for Enhancing Accessibility and Fighting Info-Exclusion*, DSAI 2018, page 55–62, New York, NY, USA, 2018. Association for Computing Machinery.

[32] Mohammad Amin Kuhail, Shahbano Farooq, Rawad Hammad, and Mohammed Bahja. Characterizing visual programming approaches for end-user developers: A systematic review. *IEEE Access*, 9:14181–14202, 2021.

[33] Slang Labs. Voice assistants. Available at https://www.slanglabs.in/voice-assistants, last accessed in January 2023.

[34] André Sousa Lago, Joao Pedro Dias, and Hugo Sereno Ferreira. Managing non-trivial internet-of-things systems with conversational assistants: A prototype and a feasibility experiment. *Journal of Computational Science*, 51:101324, 2021.

[35] John Lee and Janithapriya Buddhika K. Pathirage Don. Splitting complex english sentences. In *IWPT 2017 : 15th International Conference on Parsing Technologies*, pages 50–55. Association for Computational Linguistics (ACL), September 2017. The 15th International Conference on Parsing Technologies (IWPT 2017), IWPT 2017 ; Conference date: 20-09-2017 Through 22-09-2017.

[36] Rafael Dantas Lero, Chris Exton, and Andrew Le Gear. Communications using a speech-to-text-to-speech pipeline. In *2019 International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, pages 1–6, 2019.

[37] T. Lu and W. Neng. Future internet: The internet of things, 2010. in 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE), vol. 5, August 2010, pp. V5–376–V5–380. [Online]. Available: http://dx.doi.org/10.1109/ICACTE.2010.5579543.

[38] Knud Lasse Lueth. State of the iot 2018: Number of iot devices now at 7b – market accelerating, 2018. Available at https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/, last accessed in January 2023.

[39] P. Markopoulos and G.W.M. Rauterberg. Livinglab : a white paper. *IPO Annual Progress Report*, 35:53–65, 2000.

[40] Meta. Wit.ai. Available at https://wit.ai/ , last accessed June, 2023.

[41] MetricFire Blogger. Best grafana dashboard for iot device monitored via mqtt metrics and graphite, 2021. [Online; accessed 8-January-2023].

[42] Konstantinos Michalakis, John Aliprantis, and George Caridakis. Intelligent visual interface with the internet of things. In *Proceedings of the 2017 ACM Workshop on Interacting with Smart Objects*, SmartObject '17, page 27–30, New York, NY, USA, 2017. Association for Computing Machinery.

[43] Microsoft Garage. Microsoft dictate, Aug 2019. Available at https://www.microsoft.com/en-us/garage/profiles/dictate/, last accessed in May 2023.

[44] John Naughton. Alexa, how did amazon's voice assistant rack up a $10bn loss? Available at https://www.theguardian.com/commentisfree/2022/nov/26/alexa-how-did-amazons-voice-assistant-rack-up-a-10bn-loss, last accessed in January 2023.

[45] Raquel Navarro-Prieto and Jose J. Canas. Are visual programming languages better? the role of imagery in program comprehension. *International Journal of Human-Computer Studies*, 54(6):799–829, 2001.

[46] Node-RED. Node-red repository package. Available at https://www.npmjs.com/package/node-red , last accessed in January, 2023.

[47] Mahda Noura and Martin Gaedke. Voisma: Indoor location identification for voice-based smart home. In *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, WI-IAT '21, page 582–588, New York, NY, USA, 2022. Association for Computing Machinery.

[48] Keyur Patel, Sunil Patel, P Scholar, and Carlos. Salazar. Internet of things-iot: Definition, characteristics, architecture, enabling technologies, application and future challenges., 2016.

[49] Varsha K Patil, Omkar Hadawale, Vijaya R Pawar, and Mayank Gijre. Emotion linked aiot based cognitive home automation system with sensovisual method. In *2021 IEEE Pune Section International Conference (PuneCon)*, pages 1–7, 2021.

[50] Bruno Piedade, Joao Pedro Dias, and Filipe F Correia. An empirical study on visual programming docker compose configurations. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, pages 1–10, 2020.

[51] Pedro Ponte. *Privacy-aware Rule Conflict Prediction in Home Automation*. Master's thesis, University of Porto, R. Dr. Roberto Frias, 4200-465 Porto, 2023.

[52] Sanket Salvi, V Geetha, and S Sowmya Kamath. Jamura: A conversational smart home assistant built on telegram and google dialogflow. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, pages 1564–1571, 2019.

[53] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. *Proc. of The IEEE Workshop on Mobile Computing Systems and Applications*, pages 85 – 90, 01 1995.

[54] Albrecht Schmidt. *Ubiquitous computing - computing in context*. PhD thesis, The University of Lancaster, UK, 2003.

[55] Danny Soares, Joao Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Programming iot-spaces: A user-survey on home automation rules. In *Computational Science – ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part IV*, page 512–525, Berlin, Heidelberg, 2021. Springer-Verlag.

[56] Danny Soares, Joao Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Smart home automation survey, February 2021.

[57] Team SoundHound. Iot devices with custom voice assistants connect people to brands, 2021. Available at https://www.soundhound.com/voice-ai-blog/iot-devices-with-custom-voice-assistants-connect-people-to-brands/, last accessed in January 2023.

[58] Spy-Fy. Google home and your privacy. Available at https://spy-fy.com/google-home-and-your-privacy/, last accessed in January 2023.

[59] Rumia Sultana and Rajesh Palit. A survey on bengali speech-to-text recognition techniques. In *2014 9th International Forum on Strategic Technology (IFOST)*, pages 26–29, 2014.

[60] Md. Tahsin Tausif, Sayontan Chowdhury, Md. Shiplu Hawlader, Md. Hasanuzzaman, and Hasnain Heickal. Deep learning based bangla speech-to-text conversion. In *2018 5th International Conference on Computational Science/ Intelligence and Applied Informatics (CSII)*, pages 49–54, 2018.

[61] Tibco. What is the internet of things (iot)? Available at https://www.tibco.com/reference-center/what-is-the-internet-of-things-iot , last accessed in January, 2023.

[62] Diogo Torres, Joao Pedro Dias, André Restivo, and Hugo Sereno Ferreira. Real-time feedback in node-red for iot development: An empirical study. In *2020 IEEE/ACM 24th International Symposium on Distributed Simulation and Real Time Applications (DS-RT)*, pages 1–8. IEEE, 2020.

[63] Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, page 1–9, New York, NY, USA, 2008. Association for Computing Machinery.

[64] Raoul Wijgergangs. Iot technologies pave the road to the context-aware smart home, 2018. Available at https://www.techtarget.com/iotagenda/blog/IoT-Agenda/IoT-technologies-pave-the-road-to-the-context-aware-smart-home, last accessed in January 2023.

[65] T. Yi, I. Lee, C. Lee, G. B. Lee, M. Kim, and J. Lee. Interactive data acquisition for cbr system based smart home assistant: Utilizing function-behavior-structure framework. In *CAADRIA 2018 - 23rd International Conference on Computer-Aided Architectural Design Research in Asia: Learning, Prototyping and Adapting*, volume 2, pages 525–534, 2018. Cited By :1.