Prediction rule mining in an Ambient Intelligence context

Benoit Vuillemin

Université Claude Bernard Lyon 1

N° d'ordre NNT : 2020LYSE120

THESE DE DOCTORAT DE L'UNIVERSITE DE LYON

opérée au sein de
l'Université Claude Bernard Lyon 1

Ph.D. THESIS

Recherche de règles de prédiction dans un contexte d'Intelligence Ambiante

Prediction rule mining in an Ambient Intelligence context

Soutenue publiquement le 08/07/2020, par
Benoit Vuillemin

Ecole Doctorale N° 512
Ecole Doctorale Informatique et Mathématiques

Spécialité de doctorat : Informatique

Devant le jury composé de :

Mme EL FALLAH SEGHROUCHNI, Amal	Professeure, Sorbonne Université	Présidente
M. FOURNIER-VIGER, Philippe	Professeur, Harbin Institute of Technology	Rapporteur
M. REIGNIER, Patrick	Professeur des Universités, Grenoble INP	Rapporteur
M. CHAMPIN, Pierre-Antoine	Maître de Conférences, Université Lyon 1	Examinateur
M. NAPOLI, Amedeo	Directeur de Recherche CNRS, Université de Lorraine	Examinateur
Mme HASSAS, Salima	Professeure des Universités, Université Lyon 1	Directrice de thèse
Mme NICOL, Rozenn	Chercheure, Orange Labs Lannion	Co-directrice de thèse
Mme MATIGNON, Laetitia	Maître de Conférences, Université Lyon 1	Co-encadrante
M. DELPHIN-POULAT, Lionel	Chercheur, Orange Labs Lannion	Co-encadrant et invité

Université Claude Bernard – LYON 1

Administrateur provisoire de l'Université	M. Frédéric FLEURY
Président du Conseil Académique	M. Hamda BEN HADID
Vice-Président du Conseil d'Administration	M. Didier REVEL
Vice-Président du Conseil des Etudes et de la Vie Universitaire	M. Philippe CHEVALLIER
Vice-Président de la Commission de Recherche	M. Jean-François MORNEX
Directeur Général des Services	M. Pierre ROLLAND

COMPOSANTES SANTE

Département de Formation et Centre de Recherche en Biologie Humaine	Directrice : Mme Anne-Marie SCHOTT
Faculté d'Odontologie	Doyenne : Mme Dominique SEUX
Faculté de Médecine et Maïeutique Lyon Sud - Charles Mérieux	Doyenne : Mme Carole BURILLON
Faculté de Médecine Lyon-Est	Doyen : M. Gilles RODE
Institut des Sciences et Techniques de la Réadaptation (ISTR)	Directeur : M. Xavier PERROT
Institut des Sciences Pharmaceutiques et Biologiques (ISBP)	Directrice : Mme Christine VINCIGUERRA

COMPOSANTES & DEPARTEMENTS DE SCIENCES & TECHNOLOGIE

Département Génie Electrique et des Procédés (GEP)	Directrice : Mme Rosaria FERRIGNO
Département Informatique	Directeur : M. Behzad SHARIAT
Département Mécanique	Directeur M. Marc BUFFAT
Ecole Supérieure de Chimie, Physique, Electronique (CPE Lyon)	Directeur : Gérard PIGNAULT
Institut de Science Financière et d'Assurances (ISFA)	Directeur : M. Nicolas LEBOISNE
Institut National du Professorat et de l'Education	Administrateur Provisoire : M. Pierre CHAREYRON
Institut Universitaire de Technologie de Lyon 1	Directeur : M. Christophe VITON
Observatoire de Lyon	Directrice : Mme Isabelle DANIEL
Polytechnique Lyon	Directeur : Emmanuel PERRIN
UFR Biosciences	Administratrice provisoire : Mme Kathrin GIESELER
UFR des Sciences et Techniques des Activités Physiques et Sportives (STAPS)	Directeur : M. Yannick VANPOULLE
UFR Faculté des Sciences	Directeur : M. Bruno ANDRIOLETTI

Résumé

Cette thèse traite du sujet de l'intelligence ambiante, fusion entre l'intelligence artificielle et l'internet des objets. L'objectif de ce travail est d'extraire des règles de prédiction à partir des données fournies par les objets connectés dans un environnement, afin de proposer aux utilisateurs des automatisations. Notre principale motivation repose sur la confidentialité, les interactions entre utilisateurs et l'explicabilité du fonctionnement du système. Dans ce contexte, plusieurs contributions ont été apportées. La première est une architecture d'intelligence ambiante qui fonctionne localement et traite les données provenant d'un seul environnement connecté. La seconde est un processus de discrétisation sans *a priori* sur les données d'entrée, permettant de prendre en compte les différentes données provenant de divers objets. La troisième est un nouvel algorithme de recherche de règles sur une série temporelle, qui évite les limitations des algorithmes de l'état de l'art. L'approche a été validée par des tests sur deux bases de données réelles. Enfin, les perspectives de développement du système sont présentées.

Abstract

This thesis deals with the subject of Ambient Intelligence, the fusion between Artificial Intelligence and the Internet of Things. The goal of this work is to extract prediction rules from the data provided by connected objects in an environment, in order to propose automation to users. Our main concern relies on privacy, user interactions, and the explainability of the system's operation. In this context, several contributions were made. The first is an ambient intelligence architecture that operates locally, and processes data from a single connected environment. The second is a discretization process without *a priori* on the input data, allowing to take into account different kinds of data from various objects. The third is a new algorithm for searching rules over a time series, which avoids the limitations of state-of-the-art algorithms. The approach was validated by tests on two real databases. Finally, prospects for future developments in the system are presented.

Remerciements

Aux encadrants et membres du jury

Je tiens tout d'abord à remercier Mme Salima HASSAS d'avoir accepté d'être ma directrice de thèse. Merci pour ton encadrement, ton écoute, pour avoir trouvé le stage de fin d'études et lancé, en coordination avec l'équipe IAM d'Orange, l'initiative de la thèse.

Un immense merci à Mme Rozenn NICOL, sans qui la thèse n'aurait pas pu se faire. Merci pour ton encadrement exceptionnel, tes questions pertinentes, ta patience, et pour m'avoir supporté durant ces trois ans.

Un énorme merci aussi à M. Lionel DELPHIN-POULAT, qui a bien voulu m'encadrer par pur intérêt scientifique. Merci pour ta perspicacité, ta bonne humeur, ton apport scientifique considérable et ta mémoire absolument phénoménale. Merci aussi pour tous ces débats passionnés qui ont animé la thèse.

Merci à Mme Laetitia MATIGNON, qui a bien voulu encadrer la thèse. Merci pour ta disponibilité, tes commentaires qui ont toujours visé juste, et ton apport scientifique tout au long de la thèse.

Merci aux rapporteurs, M. Philippe FOURNIER-VIGER et M. Patrick REIGNIER, dont les échanges ont été particulièrement intéressants. Merci pour vos retours sur la thèse.

Enfin, merci aux autres membres du jury, Mme Amal EL FALLAH SEGHROUCHNI, M. Pierre-Antoine CHAMPIN, et M. Amedeo NAPOLI. Merci pour vos questions pertinentes et votre intérêt pour ce sujet.

Aux collègues d'Orange et du LIRIS

J'ai un remerciement tout particulier à adresser à l'équipe IAM d'Orange Labs Lannion, chez qui j'ai pu effectuer mon stage de fin d'études et cette thèse. Ce furent quatre années magnifiques passées à vos côtés. Merci pour toutes les opportunités que vous m'avez offertes, votre bonne humeur, vos valeurs, votre expertise et surtout votre écoute.

Un merci particulier à Christian, Katell, Nicolas, Patrick, Loïc, Cédric, Dominique, Cyril, Gildas, Julien, Régis, Joachim, Lénaïc, Lucie, Joseph, Théo, Alexandre, Georges, Thierry, Clément, Yann, Vincent, et j'en oublie sûrement beaucoup trop. Vous me manquez déjà.

Merci à l'équipe SMA du laboratoire LIRIS, ainsi qu'aux autres doctorants de ce laboratoire, de m'avoir accueilli chaleureusement lors de mes séjours à Lyon.

A ma famille

A mes parents, merci infiniment pour votre éducation, pour m'avoir poussé à donner le maximum dans ce que je fais, pour vos valeurs inculquées, et votre soutien sans faille. Moi aussi, je vous dois tout.

A ma grande sœur Aurélie, un immense merci. Je peux enfin te répondre dans ma propre thèse ! Merci de croire en moi. Tu vois, j'ai foncé ! Profite bien de la merveilleuse famille que tu as pu bâtir avec Marc et Gaspard.

A mon beau-frère Marc, merci de m'avoir appris à lâcher prise et à toujours voir la vie du bon côté.

Merci à tonton Serge de m'avoir transmis la passion de l'informatique, et merci à tonton Gilles et tata Véronique de m'avoir soutenu durant toutes ces années.

Merci à mes grands-parents, du côté de mes deux parents, pour leur amour et leur aide depuis de nombreuses années. Pépé, tu n'as malheureusement pas pu me voir docteur, mais j'espère que tu reposes en paix.

Enfin, un énorme merci à ma compagne Insun, qui m'a soutenu et m'a aidé à y voir clair dans ces périodes d'incertitudes. Maintenant que la thèse est terminée, il nous reste toute une vie à bâtir !

A mes amis

Merci enfin à mes amis, pour avoir été là à chaque instant, pour les meilleurs moments comme pour les pires. Dans le désordre, merci à Vincent, Alexandre, Guillaume, Camille, Jiseon, Pierre, Corentin, Dimitri, Luc, Seunjae, Matthieu, Jean-Pierre, Giseok et Axel. J'ai hâte de partager d'avantage de bons moments en votre compagnie !

List of Figures

List of Tables

Nomenclature

Acronyms

AI	Artificial Intelligence
AmI	Ambient Intelligence
UbiComp	Ubiquitous Computing

Symbols

δ_t	Data observation period used in the pre-processing part of the AmI system
Δ_{tr}	Duration used to divide a time series into limited subsets, which will become transactions
θ_{clu}	Threshold used in the clustering algorithm, present in the pre-processing part of the AmI system
θ_{seg}	Threshold used in the segmentation algorithm, present in the pre-processing part of the AmI system
A	Set of all atoms
a	Atom: categorical data describing a variation coming from an object, built after pre-processing of the events
A_c	Multiset of atoms representing the condition of a rule

A_o	Set of all atoms sent by the object o
A_p	Multiset of atoms representing the prediction of a rule
$category$	Category of an object, i.e. a sensor, interface, or actuator
d	Distance measure between two segments
$datatype$	Type of data sent by an object, either quantitative or categorical
$desc$	Set of descriptors of the value represented by an atom
$duration$	Duration of a segment
E	Set of all the events sent by all connected objects
e	Event: primary and unprocessed data sent by an object
e_i	Event observed at timestamp t_i
E_o	Set of all events sent by the object o
end	End timestamp for the sliding window, used in TSRuleGrowth
I	Itemset: set of all events or atoms observed at a precise timestamp
int	Interest of a rule, determining whether a rule is reliable or not
$iter$	Iterator used for multiprocessing in TSRuleGrowth
$mean$	Mean of a segment
min_{int}	Minimum interest for a rule to be reliable, used in TSRuleGrowth
min_{sup}	Minimum absolute support for a rule to be frequent, used in TSRuleGrowth
O	Set of all objects in the environment
o	Object present in the environment
r	Prediction rule
$relSup$	Relative support, determining whether a rule is frequent or not
s	Segment
$start$	Start timestamp for the sliding window, used in TSRuleGrowth
sup	Absolute support, determining whether a rule is frequent or not
t	Time stamp
t_i	Time stamp occurring at position i in the time series
TR	Set of transactions
tr	Transaction

TS	Time series of events
TS^{ref}	Representative time series, result of the segmentation algorithm of the pre-processing part of the AmI system
TS^{sim}	Simplified time series, result of the segmentation algorithm of the pre-processing part of the AmI system
TS_a	Time series of atoms
TS_s	Time series of segments
$value$	Value of an event
$variation$	Variation of a segment
$window$	Time frame in which the rules must occur, used in TSRuleGrowth

Contents

Introduction

> The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.
>
> **Mark Weiser**

Here is the basic premise that led to this thesis: the Internet of Things is a fundamental trend that will change not only the world of computing, but also our society in the coming decades. Since the 2010s, we have seen an explosion in the sale of connected objects, and in their diversification [Bosche et al., 2018]. For example, many of us have a smartphone, a personal device connected to several networks and whose possible uses are infinite. Computers, previously beige towers with a screen on a desk, have also seen a profound mutation. The towers still exist, but no longer represent the majority of sales. Indeed, new form factors have appeared, such as touch tablets and ultrabooks, making these devices more portable, and diversifying their uses. Everyday objects are also beginning to be connected, such as televisions, watches, bulbs, switches, or locks. Finally, new categories of objects have emerged, whose function depends on this connectivity specific to the Internet of Things, such as a tracker to easily find keys, called a key finder. Today, manufacturers are trying to make all possible objects connected. And, even if some examples are amusing regarding their interest [Wouk, 2019], this underlying trend may lead to uses that were not possible just a few years ago, such as detecting breakdowns on cars or helping people in distress.

These connected objects are very diverse today, but they all provide services. We distinguish two main categories of objects: sensors, which monitor the environment, and actuators, which act on it. A quick example would be a door opening sensor, to see if a door is open or not, and a connected bulb, acting on the brightness of a space. Let us take the following

example: when a user comes home, he opens the door and then turns on the light in the living room. This can therefore be observed by a door opening sensor and a connected bulb.

However, the main problem with connected objects as we know them today is that they are managed independently of each other. There is often an application to control a specific type of object, but there are few systems to operate these objects in synergy to enhance the services offered. In the previous example, the user can turn the bulb on or off via an application on his/her smartphone, and monitor the door via another application.

In this thesis, we take a first step to make these objects work in synergy, through a system that orchestrates the services proposed by the connected objects, whose goal is to offer customized automations. In the previous example, the purpose is to automate the lighting according to the data returned by the door opening sensor, so that the user no longer has to do so. For these proposals to be adapted to the environment, and customized for users, it is necessary for the system to analyze the data from the connected objects, to find habits that can lead to automation.

Thus, the scientific problem of this thesis is defined in this manner: we have, in a physical environment, one or several persons, and several connected objects, returning data over time, without necessarily a fixed sampling frequency, and which can be quantitative or categorical. How can we analyze these heterogeneous, time-stamped data, coming from multiple sources, and without *a priori* knowledge on these data, to observe regularities and identify predictions, in order to find habits in this environment and to propose automation to users in a context of service orchestration?

This is a very complex and difficult subject to tackle, combining several domains that are explained in this thesis: ambient intelligence, artificial intelligence, data processing, ergonomics, ubiquitous computing are only a few of them. This thesis not only raises scientific issues, it also raises societal issues, by putting users at the heart of the system's design.

The thesis subject being atypical, the outline of the thesis had to be as well: it guides the reader, whether an expert in the field or a neophyte, from the definition of the context, to the problems to be solved, then finally to the proposed system, its results, and its perspectives.

It is structured as follows:

In the first chapter, we take a step back by looking at a non-exhaustive history of computing. This shows that the Internet of Things is not a passing fad, but a fundamental trend, theorized in the 1990s as ubiquitous computing. This history also identifies the main issues encountered by the Internet of Things as we know it today: communication between objects, ease of use, personalization, confidentiality, security. To meet these challenges, we focus on the field of ambient intelligence: the fusion of artificial intelligence and ubiquitous computing. We conclude this chapter with the main purpose of the thesis: to propose a service orchestrator, i.e. a system that operates the different objects, providing automation in a connected environment. This allows us to have more complex services, because they include different connected objects.

In the second chapter, we identify the research questions addressed in the thesis, which motivated some choices. For example, the issue of confidentiality leads us to consider a system that operates locally, in an environment, and not in the cloud. In addition, current artificial intelligence techniques and the dangerous situations they may cause on the environment are leading us to let the control of this system to the users. The ambient intelligence only makes automation proposals, which the user can activate at will. This raises a major challenge: the understanding of these proposals, and therefore the semantics to be applied to them. We also characterize the data that can be obtained from connected objects, and the possible forms of automation proposals. Taking into account these heterogeneous data which can also be sent

temporally in an irregular way is also one of the main problems of the thesis.

Then in the third chapter, we detail the architecture of the desired ambient intelligence system. This architecture is intended to address the issues raised in the previous chapters. In this system, data of various types are pre-processed in order to unify them. Then, a search for prediction rules is made on these pre-processed data, in order to discover habits, and therefore automation proposals. These proposals are then sent to users who can accept them or not. Then, the system improves based on user feedback, and applies the accepted automations. In this thesis, the focus has been on data pre-processing and the search for prediction rules, which are detailed in the following chapters.

The next two chapters describe the algorithms used in the implementation of the system. Chapter four specifies the algorithms used in the pre-processing part of the data, allowing the different types of data to be unified. Chapter five reveals the search for rules. This part contains some of the major contributions of the thesis, such as a new algorithm for searching rules that addresses several problems in the state of the art in this field.

Finally, in the sixth and final chapter, we discuss the prospects for the proposed system. These perspectives extend well beyond the simple scientific framework, and aims to provide advice on the development of future ambient intelligence systems, regarding interactions with users, intelligibility, respect for their privacy, system improvement from user feedback, and also technical aspects, such as the structure of the prediction rules or the pre-processing of data.

To sum up, this is a vast and very varied subject, raising several problems. So let us get started by taking an overview of the history of computing, to get a clearer idea of where the Internet of Things is heading and what remains to be done to get there.

Chapter 1

Context

1.1 Introduction

This thesis presents a system looking for predictions in data from connected objects scattered in an environment. To clarify the motivations that led to this topic, we define in this chapter its context and the main objectives. We first focus on a short history of computing, with a focus on Ubiquitous Computing (UbiComp) to argue that the Internet of Things is a fundamental trend, and to identify the major issues of the Internet of Things as we know it today. We also present the major scientific domains related to the thesis, with a description of some of the sub-domains related to the thesis: Artificial Intelligence (AI) and Ambient Intelligence (AmI).

1.2 Ubiquitous Computing (UbiComp)

As stated in the introduction, we want to propose a system that orchestrates the services provided by the connected objects. The fact that we are in an environment populated by connected objects of all kinds is no coincidence. In the 1980s and 1990s, Mark Weiser [Weiser, 1996], Chief Scientist at the Palo Alto Research Center, theorized this type of environment, where computers remain in the background to help the user, as **Ubiquitous Computing**.

It is interesting to analyze the history of computer science under the prism of this theory,

because it makes it possible to identify the major problems that still remain, some of which the thesis attempts to solve. According to Mark Weiser, there are three major eras of computing: Mainframes, Personal Computers, and UbiComp, which will be detailed in the following.

1.2.1 Mainframes: multiple users per machine (1940-1970)

In the 1940s, the first electronic computers appeared. They took up a lot of space, took years to develop, produce and install, and used a tremendous amount of electricity. However, they allowed automated calculations, which made them extremely useful despite their shortcomings. Thus, several users were working on a single powerful machine.

During the Second World War, computers such as the Zuse Z3 in Germany provided statistical analysis. The Bombe, made in part by Alan Turing, and the Colossus series in the United Kingdom made it possible to break the secret codes used by the enemy. Thus, from the dawn of computer science, computers were not only used to do calculations. They were also used to make a series of logical actions.

The physical elements for making logical actions were first electro-mechanical machines, then became vacuum tubes. Although fundamental for the computers of the time, they were also their main limitation. Indeed, they could very easily break, making the device unreliable [Randall 5th, 2006]. The invention of the Transistor, a replacement of the vacuum tube in the late 1940s solved those problems. As a result, computers became more and more common in the decades that came after, because they were cheaper to manufacture and more reliable.

Computers came into companies in the form of mainframes: an entire room was used to store the machine, and programs could be executed, among others, with perforated paper tapes. Computers were becoming more and more powerful, thanks to the miniaturization of components. The fields of application of these machines have expanded, from the guidance of spaceships in the NASA, to Sketchpad, a real-time and interactive drawing system.

Mainframes are still used today, for applications with a high need for computing power. However, from the late 1970s onwards, they were competing with computers that were certainly less powerful, but much smaller in size.

1.2.2 Personal Computing: one user per machine (1970-2010)

1.2.2.1 Democratization of computers

The increase in power and miniaturization led to the invention of the microprocessor by Intel in 1971. The microprocessor is based on the principles of the processor, the central component of the computer that allows logical actions, with less power but in a much smaller size. With this component, anyone with good electrical and electronic skills could build their own computer. This innovation led to the creation of the first operating systems, like CP/M, by Gary Kildall [Computer Chronicles, 1995], and some hand-built computers could be sold to the public, like the Apple I in 1976.

However, the democratization began in the late 1970s and 1980s with the mass production of personal computers: the Apple II, in 1977, was one of the first personal computers sold in millions of units. Computers entered homes and offices because the applications of these machines were multiple, in the form of programs that could be purchased in stores. Thus, two computers of the same model could be distinguished by the programs installed and the data recorded on them. The computer became personal to the users.

The introduction of the IBM PC in 1981 contributed to this democratization. This machine was sold with an operating system, Microsoft MS-DOS, which could be installed on

other machines called "PC Clones". The operating system served as an intermediate layer between the physical computer and the programs. Thus, a program was no longer created for a single computer model, but for an operating system, which itself was compatible with a set of computers that could be different. The democratization of PC Clones and the fact that there was only one main operating system at the time led to the diversification of applications. The introduction of the graphical user interface and mouse has made computers easier to use, facilitating the entry of computers into the home. These two advances, from the Xerox Research Center, were brought to the general public by the Macintosh in 1984, and Windows in the following years.

This led to a period of mutual attractiveness between program developers and the general public: developers were attracted to the now called software market, because they could create programs that millions of computers could run, for millions of people. In addition, people were attracted to computers because they could do useful things for them: business applications like spreadsheets or word processing, entertainment like games and creative tools like music sampling.

The need for miniaturization, for internal components and the machines themselves, has continued. People wanted more power in their computers, as well as more portability. Thus, laptops appeared in the 1980s and then became thinner, lighter, and more efficient over time. Meanwhile, other categories of computers have begun to emerge, such as game consoles, cell phones, MP3 players or Personal Digital Assistants. With the increase in power over time, more and more advanced applications were available to the public, such as multimedia, picture, and video editing.

The original vision of Microsoft's founders in the 1970s became a reality: "A computer on every desk and in every home" [Beaumont, 2008], but the democratization of computers became really important with the introduction of the Internet.

1.2.2.2 Internet

The need to connect existing machines has increased as the machines themselves have become more common. The potential for information transmission between machines was enormous, and research on connections and protocol communication was carried out between the 1960s and the late 1980s, mainly in the military [Hauben, 2007] or in large research organizations, such as CERN [Berners-Lee, 1989].

The Internet, essentially a computer network, became publicly accessible worldwide in the late 1990s. At the beginning of its public access, it was possible to browse web pages composed of text and images, send emails and exchange files. Before the Internet, some networks of interconnected machines were accessible to the public, such as the Minitel in France, but they consisted only of consuming content [Computer Chronicles, 1990]. On the Internet, anyone can host a web page or exchange data with others.

Just like personal computing, the democratization of the Internet has led to a multiplication and diversification of its applications: music and video streaming, social media, online shopping... With these services, the Internet has become the main reason to buy a computer.

As mentioned earlier, more and more different devices were beginning to appear, increasing the number of computers per person. However, one product category accelerated this transition in the late 2000s: the smartphone.

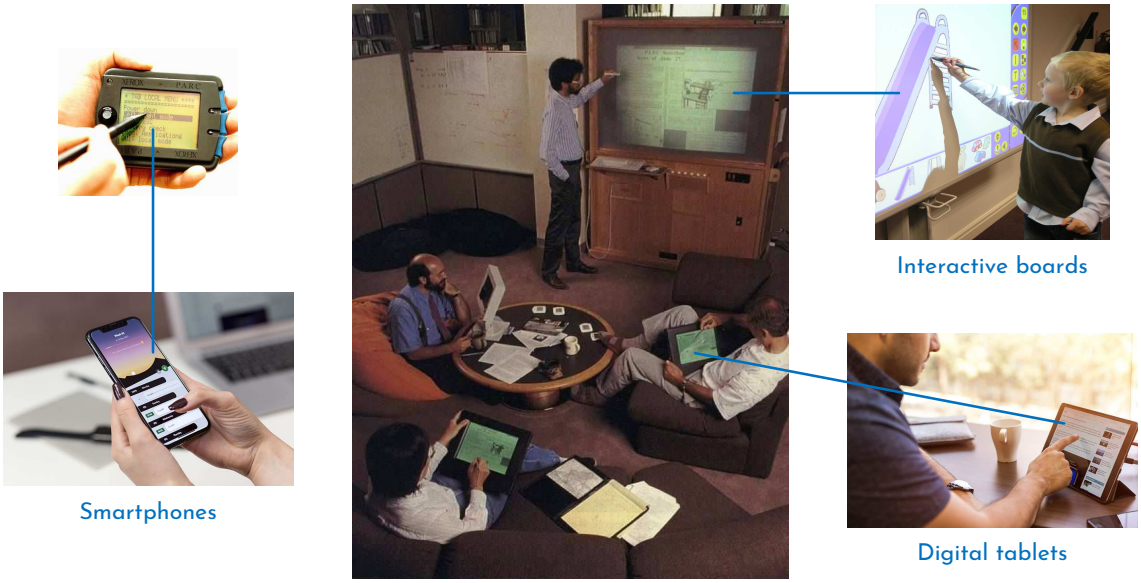


Figure 1.1: Transposition of the devices invented at the Xerox Lab into today's world[1]

1.2.3 Ubiquitous Computing: multiple machines per user (2010-*)

1.2.3.1 The original vision

The main idea of UbiComp, according to Mark Weiser, was that computers would act as a background for the users' environment. According to him, in the era of the Personal Computer, machines were a focus of attention for users. It was on these computers that programs were executed, movies were played, music listened to, Internet consumed, etc. In UbiComp, machines would assist the user, help him to do actions, and would no longer represent a center of attention. The environment would become populated by multiple machines, usable by anyone.

With this in mind, Mark Weiser's team of researchers had developed several product categories. Among the best known are the "tabs", inch-scale machines in the spirit of the Post-it. A personal badge has been developed in this category. It was used to unlock doors, automate the forwarding of telephone calls, and included a touch screen to display the calendar and send messages. Then come the "pads" [Want et al., 1995], digital tablets in the spirit of a sheet of paper, with a screen and a stylus, that can be used by anyone. Finally, "boards", large screens used to display content during meetings, as seen in figure 1.1.

To reach this era, technical advances have been identified: hardware, with efficient and inexpensive computers, network, one short-range and one long-range, and software for applications in the ubiquitous environment. But a main danger had already been identified by researchers at the time, the possible lack of privacy: "hundreds of computers in every room, all capable of sensing people near them and linked by high-speed networks, have the potential to make totalitarianism up to now seem like sheerest anarchy. Just as a workstation on a local-area network can be programmed to intercept messages meant for others, a single rogue tab in a room could potentially record everything that happened there." [Weiser, 1991].

[1] Permission given by Bill Schilit for the top left photo and Matthew Mulbry for the center photo. Thanks to Robert S Mulbry for kindly putting me in contact with Matthew Mulbry.

1.2.3.2 The reality

In the early 2000s, the services offered by the Internet made it possible for it to be widespread in homes and offices. Several technical advances made it possible to make the Internet accessible from anywhere: a short-range network called Wi-Fi, and the new generations of long-range mobile networks. This allowed mobile phones to evolve into a new category of devices: smartphones. They integrated all mobile phone services, Personal Digital Assistants, and Internet access. But, despite this, they were difficult for the general public to use, due to their user interface and complex inputs.

The introduction of the iPhone in 2007 led to a simplification of the smartphone category. Indeed, both the user interface and the input method have become simpler, mainly based on a touch screen. The introduction of these new smartphones has led to an intensification of research on embedded terminals. As a result, in parallel with smartphones, other forms of computers have evolved and become more widespread: digital tablets, used at home and in the office, and interactive boards, used in meetings or to teach a course.

Thus, the objects invented and theorized at the Palo Alto Research Center have finally found their successors. The "tabs" have become smartphones, the "pads" digital tablets, and the "boards" interactive boards. And, with different short- and long-range networks, all these devices were able to communicate with each other.

Furthermore, with these smartphones, hardware manufacturers could create new types of devices, which were connected to the phone via an application. This led to the emergence of the Internet of Things (IoT). Today, we tend to connect all the objects in the house. From the car to the refrigerator, everything can be connected to the Internet, share data from on-board sensors, and take action on the environment.

Mark Weiser's vision is beginning to come true: connected objects, small energy efficient computers, are integrated into the environment in the background, helping the user to do actions. Connected objects assist users, no longer represent a focus of attention, and do not necessarily have an owner. They are connected by different networks, long-range like mobile networks, and short-range like Bluetooth or Wi-Fi. The era of UbiComp has begun.

1.2.4 Problems

This short history explains how UbiComp was born and exists today. It highlights several dynamics that have existed since the beginning of computing:

- The constant miniaturization of components and machines, in line with their increasing power.
- The diversification of these machines and their uses.
- The need to simplify their use, to be accepted by the general public; through the examples of the graphical interface and smartphones.
- The need to make the machines work together, either by unifying their characteristics or by connecting them in a single network.

Mark Weiser's theory became a reality: we are in the era of UbiComp. Today, any object can be connected, such as door opening sensors, bulbs, televisions, object locators, cameras, shutters, electrical outlets, speakers, alarm clocks, watches, switches... Many categories of connected objects exist, allowing to monitor the environment and make remote actions. Few

houses are equipped with connected objects at the moment, but this proportion is constantly increasing [Griffith, 2019].

However, at the moment, objects still communicate little together. Indeed, several competing networks exist for these objects: Bluetooth, Wi-Fi, Sigfox, Zigbee, LoRA are only examples. These networks are not compatible with each other. Some objects may be connected to others within the same ecosystem, but several ecosystems coexist. Unless adhering to a single ecosystem, it is necessary to use several different applications to act on these objects.

The aim in the coming years is to make these different objects work together, with a common goal. A connected object being a set of services, the idea of having all these different objects interact is part of the domain of service orchestration.

A **service orchestrator** is a program that chooses the order of invocation of the services to which it is connected. Thus, in a connected environment, this can solve the non-interoperability of objects, by making them work together for a common purpose, through a single program. It also makes it possible to provide more complex services. Also, such system must make sense of its multimodal data retrieved from the different connected objects. Service orchestrators already exist, but the interactions they allow are basic. They allow to act manually on objects within the same interface such as "Apple Homekit", or to make basic rules, as proposed by IFTTT ("If This Then That" [Betters, 2018]). At present, connected objects have no easy way to work together to achieve the primary vision of UbiComp.

Service orchestrators are currently interfaces used to make complex scenarios manually. Thus, it remains difficult to use these tools, as any composite service must be designed by the user. So necessarily, the system provided is highly customized, because it is the user who feeds it, but it is still difficult to imagine all the scenarios to automate. One solution to this problem would be for the system to provide automation on its own, based on the habits observed among users. This system would still be customized but easier to use.

Another issue raised by Mark Weiser is **privacy**. With objects communicating in an environment, it is technically possible to monitor everything that happens in it. Another problem is related to the previous one, that of **security**. Hacking into these connected objects can lead to a gold mine of information about the environment, and its users, and can lead to control of the involved environment. Thus, access to and control of these objects must be controlled to avoid these risks.

The customization of a connected environment, and therefore of the service orchestrator, must also be taken into account, to make it suitable for users, in other words, useful.

1.2.5 Summary

In this section, we have identified the main needs in the field of the Internet of Things: interoperability between objects, ease of use, personalization, privacy and security.

To address some of these issues, our target system is an automatic service orchestrator. It aims to make these objects work together, to provide more complex services. It must therefore observe, collect, and process multimodal data, as it comes from different connected objects. It offers automation based on the habits observed among its users, which makes it more personalized and easier to use. The principles of confidentiality and personalization will be at the heart of the thesis. However, the security aspect is not covered here, but work on it may complement the presented work.

Searching for user habits from raw data of connected objects is a difficult task. It requires extensive data analysis that can adapt and modify its results as changes are perceived in the environment and among users. The thesis therefore falls within the scope of the field of AI,

seeking to integrate a notion of intelligence into computer programs. We will detail this area and then focus on a related research area called AmI, which is the fusion of UbiComp and AI.

1.3 Artificial Intelligence (AI)

Since the beginning of computers, scientists have tried to create programs that simulate the cognitive functions of living beings, especially human beings. This field, AI, is extremely vast, and extends even more from year to year. AI is very often mentioned in the media and within companies. It is also used and imagined in the artistic field with examples such as HAL 9000 in 2001: A Space Odyssey [Kubrick, 1968], Jarvis in the Iron Man series [Favreau, 2008] or Cortana in the video game franchise Halo [Jones, 2001], as well as Isaac Asimov's "Robots" series, which started with the book "I, Robot" [Asimov, 1950].

But what does this field really mean, and where are we now? To do this, we will also observe a short history of AI to understand its definition, and make a non-exhaustive state of the art to observe what can and cannot be done.

1.3.1 History and definition

AI has a history that goes back to the very beginnings of computer science. Alan Turing, quoted in section 1.2.1, had asked the question "Can machines think?" in [Turing, 1950]. He proposed a test to evaluate this type of machine, later called the Turing test. It consists in putting a human being, called an interrogator, in a blind verbal confrontation with a computer and another human being. If the interrogator is not able to say which of his interlocutors is the computer, the computer has passed the test.

The Dartmouth workshop in 1956 is considered to be the founding event of the AI field. It was proposed by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon, who are considered to be the inventors of this field. In this workshop, the term "AI" and its global definition were defined.

According to John McCarthy, **Artificial Intelligence** is "the science and engineering of making intelligent machines, especially intelligent computer programs", where intelligence is defined by "the computational part of the ability to achieve goals in the world. Varying kinds and degrees of intelligence occur in people, many animals and some machines" [McCarthy, 2007]. AI has a fuzzy definition, since the notion of intelligence itself is also fuzzy. This explains why nowadays, almost a century after the beginning of computer science, it is a gigantic field of research that is highly valued by researchers.

The field of AI was launched internationally and by the late 1950s, more and more work were being integrated into it. For example, the Perceptron, invented in 1957 by Frank Rosenblatt, was an image recognition algorithm, and formed the first artificial neural network [Rosenblatt, 1958]. Then, with the rise in power and the democratization of computers, several fields developed. Among them, Machine Learning, aimed to enable computers to learn from data, or developmental learning, aimed at simulating the mechanisms that enable the learning of new skills and knowledge in living beings, or robotics, also taking into account social issues. More recently, Deep Learning, a field that develops much denser neural networks, has allowed computers to perform more complex tasks than before with a high degree of accuracy, especially in image recognition [LeCun et al., 2015]. A deep reinforcement learning algorithm, for example, could play basic Atari 2600 video games at a level comparable to that of a human player [Mnih et al., 2015].

Several achievements have been attributed to the AI domain, particularly in the field of gaming. Chess [Feng-Hsiung Hsu, 1999], Jeopardy [Ferrucci et al., 2010], and more recently the game of Go [Yan, 2016] have seen an AI beat the best players in the field.

Overall, AI tools are in a widespread use in today's world. It is possible, at this very moment, to recognize people on pictures, to transcribe voice into text, or to know people's buying or consuming habits. Today, AI is a term that is widely used and known, in full expansion and far from being fully explored. Work is going in all directions, and because of its vague definition, there have been several lively debates since the beginning of the field.

1.3.2 Philosophical debates

AI is such a vast field that it brings together several disciplines. Among these is philosophy, which discusses AI and its technologies at a higher level.

1.3.2.1 Strong/Weak AI

The greatest philosophical debate is found in the distinction between strong and weak AI. To simplify, AI is strong if it is a "machine intelligence with the full range of human intelligence" [Kurzweil, 2006], where it was not built for a specific task. Any AI that does not meet this definition is considered weak. A weak AI is programmed to do specific tasks, and cannot be used to do tasks other than what it has been programmed to do.

A good example of the distinction between strong and weak AI is the "Chinese room" [Searle, 1980]. In this scenario, we imagine locking someone in a room. This person, who is referred to as an operator, has no knowledge of the Chinese language. Then, a set of rules, perfectly clear to this person, is made available to him or her to answer questions written in Chinese. The operator receives questions in Chinese from an interrogator who knows this language. The operator can therefore answer the questions, by writing other sentences in Chinese, thanks to the available rules. Thus, for the interrogator, the operator has an advanced knowledge of Chinese, because he or she can answer the questions. But the operator does not understand the received questions nor the produced answers. The operator only follows predefined rules.

This experience shows that complex problems can be solved with the tools we are given, without understanding the meaning of the problem. If we get a new problem that is not very different from what was given before, we can solve it with the tools used previously, but if we get something radically different from usual, but still within the scope of the problem, we could not answer this new problem, because the tools given do not allow us to, and we did not understand the deep meaning of what we were doing.

The AI developed today are considered weak. This is put into perspective by Hans Moravec: "it is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility." [Moravec, 1990]. It is possible to make intelligences that beat the best players in the world in chess [Barden and Leonard, 2011], or Go [Silver et al., 2018] but not to build an intelligent system for a robot, which learns to walk alone on any surface, simply by observing the environment [Kuipers et al., 2006]. Why? It depends, in part, on the environment in which the intelligence is located. If the environment is defined *a priori*, or easily determined, AI can easily evolve to achieve the objective set, thanks to the tools given to it. In a complex environment with uncertainties, such as the real world, this is very difficult if not impossible. Also, human intelligence, and especially its functioning, is far from being defined, and several competing theories attempt to describe it. According to

psychologists, there are even several forms of human intelligence. For example, Howard Gardner's theory states that there are several forms of intelligence, including linguistic intelligence, which allows us to communicate orally and in writing, but also spatial intelligence, logical intelligence, interpersonal intelligence for knowing others, and intrapersonal intelligence for knowing oneself [Gardner, 1983; Gardner, 1993]. Thus, according to Gardner's theory, an AI must be strong by being multiple, containing specialized intelligences in specific fields.

Strong AI is still only a concept today, but work is underway in this direction [Silver et al., 2017; Tuyls et al., 2018]. Deep learning aims to recreate levels of generalization between input data and results, in order to be able to respond better when it encounters unknown data.

1.3.2.2 Top-Down Approach

Two approaches, already identified by Alan Turing in the 1940s [Turing, 1950], are possible in AI. In top-down AI, problems are dealt with at a high level. The problem would be described as much as possible to achieve a model that provides a solution. The types of AI that fall into this category are called "Symbolic AI", because the problem is described as much as possible via symbols, to obtain a simpler representation on which the intelligence can reason.

Let us take an example: a system playing chess. Here, we can code the rules of a chess game, like the types of pieces, the moves that each type can make, the times when the system can make a move, and the winning conditions. Those are the pre-given symbols. With that said, and by looking at a lot of matches, or by generating matches, the system can perfect itself and learn to have the best behavior to win the game every time, against anyone.

Several attempts to achieve strong AI have been made using this approach, like Soar [Laird et al., 1987], Epic [Kieras and Meyer, 1997], or ICARUS [Langley et al., 1991], and a theory has been made by Allen Newell and Herbert Simon in 1976: "The Physical Symbol System Hypothesis. A physical symbol system has the necessary and sufficient means for general intelligent action" [Newell and Simon, 1976].

"Symbolic AI" represented the dominant paradigm in AI until the 1990s [Vernon et al., 2007], because it is easy for some problems to be expressed by a mathematical representation. Also, in this top-down approach, there is no need for the system to learn by itself the low-level knowledge relative to the problem, as it would take much more time to develop and study.

However, symbolic AI systems have three main problems, according to [Christensen et al., 2004]:

- The Symbol Grounding Problem [Harnad, 1990]: in a symbolic system, symbols can refer to characteristics, or objects, in the environment. How are these symbols related to those objects? To understand this question, we can make the analogy with the words we use. How do these words have a meaning, and what is the meaning of a word? This problem is related to the problem of the "Chinese room", explained in section 1.3.2.1. An operator can answer questions in Chinese, even without knowing this language, in the same way that a symbolic system can solve a problem, without understanding the meaning of what is being done. Thus, this problem is also related to the problem of consciousness, and its definition [Harnad, 2007].

- The Combinatorial Explosion Problem: to get knowledge, the system can consider any possible relation between data. However, this can lead to a huge amount of computation needs. The number of combinations to find a solution can grow exponentially, leading to prohibitively long computing times. Therefore, the problem is to find valuable knowledge while avoiding the exponential growth of needed combinations.

- The Frame Problem [McCarthy and Hayes, 1969]: How do we model the effect of an action in the environment? Also, how do we model the things that have not been modified by an action in the environment? It is possible to consider all the changes in the environment, but this leads to the previous problem, namely the combinatorial explosion. In addition, some aspects of the environment may have been modified by a factor other than the action being analyzed.

These problems can explain why Symbolic AI systems have difficulties making robust sensorimotor interactions in other than still, restrained, noiseless environments [Vernon et al., 2007].

1.3.2.3 Bottom-up Approach

Bottom-up AI researchers take the opposite approach of top-down and simulate cognitive structures or functions in humans or animals, in an attempt to obtain more powerful tools for problem solving. Recently, the expansion of deep learning, larger neural networks requiring high computing power, and the development of robotics have refocused AI on a bottom-up approach. There are several adherent movements in this approach, among them neural networks, detailed above, and constructivism.

Constructivism is a philosophical theory born during the 19th and 20th century [Hawkins, 2012]. One of the first publications introducing the constructivist theory for learning was written by Jean Piaget, a Swiss psychologist, during the early 20th century [Piaget, 1936]. Its original paper [Piaget and Cook, 1952] defines three main process in learning: Assimilation, Accommodation and Organization:

- Assimilation is the process where new information is transcribed into the internal knowledge structure.

- When the new information differs radically from the internal knowledge representation, the process of Accommodation makes the changes in the structure to integrate it.

- In parallel to the two previous processes, the structure is organizing itself to maintain its viability: this is Organization.

In simple terms, Constructivism stipulates that every act of learning consists of:

- Transforming the new information into its own framework: Assimilation,

- Transforming old knowledge into new knowledge, renew old knowledge: Accommodation,

- Organizing the knowledge to maintain its structure: Organization [Masciotra, 2007].

Some researchers expanded the definition of Constructivism, by introducing radical constructivism. This theory stipulates that the environment cannot be known as is, but only with the interactions we have with it. Therefore, we can never know what the environment really is: "Radical constructivism, thus, is radical because it breaks with convention and develops a theory of knowledge in which knowledge does not reflect an "objective" ontological reality, but exclusively an ordering and organization of a world constituted by our experience." [Von Glasersfeld, 1984].

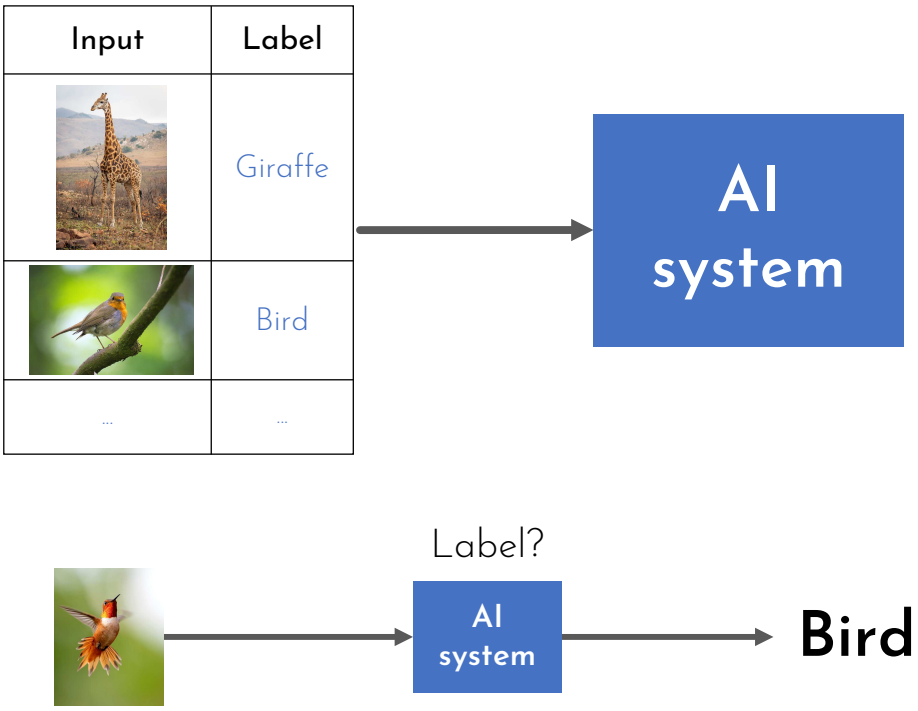


Figure 1.2: Illustration of a supervised learning algorithm

1.3.3 Techniques

In this section, we will detail some of the techniques used in AI. As a reminder, AI is a gigantic domain, so it is very difficult, if not impossible, to have a precise, delimited, and complete mapping that is accepted by all researchers in the field. Also, this section is not intended to be exhaustive, and will only detail some techniques.

1.3.3.1 Supervised learning

Supervised learning is an area of AI where an algorithm learns, from input data, to find a result that corresponds to the expectations of the algorithm designer. The term "supervised" refers to the fact that, during its learning phase, the result that the algorithm must obtain is given explicitly, or via an indication.

In simple terms, a supervised learning algorithm has two main components: inputs and outputs, also called labels. The aim of this kind of learning is to find relations between inputs and outputs, in order to **predict** correct outputs for given inputs. To make those relations, we give to the system a set of labeled inputs, meaning that those inputs already have the correct outputs embedded. This set is called the training set. In fact, it can be symbolized by a mathematical function, such as $f(x) = y$, where x is the input data, f the function with parameters to be determined, and y the output data [Murphy, 2012]. Training the system consists in determining those parameters.

A simple example would be an algorithm capable of classifying animal images according to species (figure 1.2). If we send it an image of a magpie, it could send as an answer that this is a bird. So, in the previous formalization, x is a picture of the animal, y its related species in the form of a word, and f the function to transform x into y.

There are two types of outputs for supervised learning [Murphy, 2012]:

- Classification, where the output belongs to a class; the system must predict the correct

category of a given input. A good example of classification is the one about animal images mentioned above.

- Regression, where the output does not represent a class, but is rather a continuous data, a quantifiable number. An example of such a system could be to predict the brightness in an environment according to weather conditions (such as cloud density, season, etc.).

Reinforcement Learning Reinforced learning is a specific sub-domain of supervised learning where the results to be expected are not given explicitly, but rather an indication, in the form of a reward proportional to the quality of the response provided by the algorithm. In a sense, the system must **control** its actions in order to get the best reward. As [Sutton and Barto, 1998] explains, "Reinforcement learning is learning what to do--how to map situations to actions--so as to maximize a numerical reward signal. The learner is not told which actions to take, as in most forms of machine learning, but instead must discover which actions yield the most reward by trying them".

A good example of reinforcement learning is a simulated leg control system that must learn to walk as quickly as possible on a flat surface. For this to work, the system can simulate a movement. A monitoring program can calculate the distance traveled by the leg using this movement, and give a reward proportional to it. The learning system then influences the search for an optimal movement in order to obtain the maximum possible reward, and after some time, knows the correct movement to walk quickly by itself on a flat surface.

A well-known paradigm in this field is that of exploration/exploitation. Indeed, in the previous example, a system can find a good solution to move fast. In this case, there can be two choices of evolution:

- He can exploit this movement set, by refining it, but he may miss the optimal solution. This is called a local optimum.

- He can explore completely new movement sets, to try to find another series of movements that may be even better than the previous one. However, he may not find a movement set as effective as the previous one.

Reinforcement algorithms, as well as other algorithms in AI, must take this dilemma into account. Several methods deal with this subject, including the multi-armed bandit [Katehakis and Veinott, 1987] and bio-inspired algorithms [Yang and Karamanoglu, 2013; Pazhaniraja et al., 2017].

Artificial Neural Networks Artificial neural networks are a central domain in AI. The objective is to take inspiration from brain structures from living beings, to perform tasks that are still too complex to do using conventional programming [LeCun et al., 2015]. Neural network techniques are most often supervised, hence its inclusion in this section. Let us take again the example of the animal image classification program. Defining a hand-written program to describe an animal's species from a photo is extremely difficult, if not impossible, due to its complexity. Yet we, as human beings, are able to know which animal it is just from the images sent by our eyes.

It is to address this type of problem that neural networks have been studied by computer scientists. One of the first examples of artificial neural networks was the Perceptron [Rosenblatt, 1958], which allowed letters to be recognized from an image. To simplify, a brain is composed of interconnected neurons, which can be activated electrically. An artificial neural

network is inspired by its biological counterpart, with a structure of interconnected neurons. An introduction to the functioning of artificial neural networks can be found in [Zou et al., 2008].

Neural networks can achieve very good results in a problem where supervised learning can be applied, such as our algorithm for classifying animal images by species. The field of neural networks is constantly evolving. In recent years, deep learning has made great progress, making it possible to respond to even more complex problems.

However, they may require a lot of data to produce convincing results, and there is a risk of **overfitting**, i.e. obtaining a system that is too specialized in the data it has been given, but does not have the necessary generalization to respond correctly to unknown data. Moreover, it is still very difficult to understand why, on the basis of what features, a neural network has chosen one particular result over another. The **explainability** of neural networks is therefore an emerging field today [Vaughan et al., 2018; Yang et al., 2019].

1.3.3.2 Unsupervised Learning

By definition, Unsupervised Learning is a type of learning that is not supervised by anything but the learning system: "the machine simply receives inputs [...] but obtains neither supervised target outputs, nor rewards from its environment." [Ghahramani, 2004]. Here, we do not give results at all, just input data. The goal of such system is to find regularities, or create a representation of the data given in input: "discover 'interesting structure' in the data" [Murphy, 2012], "find regularities in the input" [Alpaydin, 2014]. There are some main types in unsupervised learning:

- Clustering: "data clustering is to group a set of data (without a predefined class attribute), based on the conceptual clustering principle: maximizing the intraclass similarity and minimizing the interclass similarity" [Chen et al., 1996]. In this example, we give data that we must put into separate and distinct groups, called clusters.

- Dimensionality reduction: this is used when the dimensionality of input data is high, i.e. it has a lot of features, which can be difficult for a human being to analyze [Murphy, 2012].

- Matrix completion: this can help filling missing input data inside a matrix, or a table of data. A notable example of this application is image inpainting: "The goal is to "fill in" holes (e.g., due to scratches or occlusions) in an image with realistic texture." [Murphy, 2012]. It is also possible to do matrix completion and image inpainting with supervised learning, especially with deep learning [Yang et al., 2017; Yu et al., 2018].

- Pattern/Rule Mining: this can help building a structure representing relations inside input data. There are two main goals: discover new knowledge, and find correlations to make predictions [Murphy, 2012].

1.3.3.3 Online/Offline Learning

In AI, in parallel to the supervised and the unsupervised, there are traditionally two main learning methods: online and offline learning.

- In Online Learning, "the sequence of instances is chosen by an adversary and the instances are presented to the learner one-by-one." [Ben-David et al., 1997].

- In Offline Learning, "the learner knows the sequence of elements in advance." [Ben-David et al., 1997].

In other words:

- In the life of an Offline Learning system, there are only two phases: training and application. First, we train the system with a single batch of data. Then the system is applied to new data in order to find the expected result. Here, the system does not evolve.

- In Online Learning, the system progressively evolves as the data arrive. The system matures over time.

Naturally, hybrid systems exist, which can be updated in batches of data, for example. In this way, they can relearn about the new data, and therefore evolve, but only in a piecemeal way.

1.3.4 Summary

In this section, we have provided a brief summary of the field of AI, and demonstrated that many issues are present, both technically and philosophically. Other techniques than those presented above are possible, in particular techniques based on statistics, which we will detail later in the thesis. Let us now look at some applications of AI, especially in the context of UbiComp, with AmI.

1.4 Ambient Intelligence (AmI)

AmI is a relatively recent concept, dating back to 1998 at Philips [Aarts and Encarnação, 2006]. It can be defined as follows: "A digital environment that proactively, but sensibly, supports people in their daily lives" [Augusto and McCullagh, 2007]. This definition is very similar to that of UbiComp, in the sense that we are no longer talking about a personal machine, but an environment populated by machines in the background. The difference with UbiComp lies in the proactive aspect of this environment, which requires intelligence.

Indeed, Mark Weiser's various publications did not deal with intelligence, but focused on the technical aspect of UbiComp, such as the chips to create and which interface to bring to the user. AmI can thus be interpreted as the fusion of UbiComp and AI, whose purpose is to recreate cognitive and perceptual aspects of the brains in machines.

This thesis is fully in the field of AmI. Indeed, creating a personalized recommendation system requires learning, to know the users' habits and predict their actions. To illustrate the field of AmI, we will detail some of its research approaches.

The concept of AmI is a broad one, which can be distinguished in several layers described in figure 1.3, and inspired by [Olaru et al., 2013]. First is the physical environment, populated by connected objects, then the network that allows these objects to communicate with each other or with distant machines. Then there is the intelligence layer, which can be centralized, i.e. all the data is processed by the same machine, or it can be distributed, i.e. processed by several machines. The last level of AmI is the interaction with users, because of the main purpose of AmI, which is to help its users.

Thus, building an AmI system requires work in AI, but also on the machines present in the environment, on the communication networks, and also on the interactions that this AmI

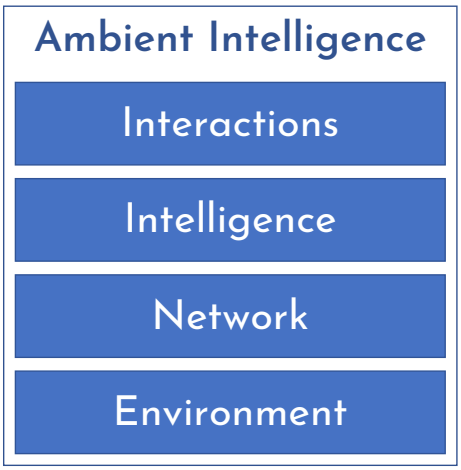


Figure 1.3: The different layers of AmI, inspired by [Olaru et al., 2013]

system has with its user. Work can also be done on the social implications of such an environment. Scientists are therefore trying to address this subject through several complementary approaches.

This thesis focuses on the intelligence layer of AmI, although it also asks questions about the interactions the AmI system has with its users. Here, we detail work on the AmI layers, including the environment, with the developed test platforms and their purpose, user interactions, whether explicit or not, and intelligence, the central area of the thesis. Work on networks that allow the environment to communicate, such as Bluetooth, Wi-Fi or 5G, is not treated here, as it is too far from the thesis problem. This state of the art was inspired in part by [Cook et al., 2009], [Nakashima et al., 2010] and by [Acampora et al., 2013] which specializes in healthcare.

1.4.1 Environment: test platforms

Working on AmI inevitably involves working with environments. Thus, several test platforms have been created since the emergence of the field. We will see that some platforms are developed for a specific purpose, such as helping the elderly or workers, and others are developed to provide generic test environments that can be used by several research projects. However, whatever the purpose, these platforms all represent a specific environment, which shows that research on AmI focuses on specific contexts. AmI aims to develop these contexts into intelligent environments that meet users' expectations. In this field of research, the adjective "smart" is therefore often used. Those locations are referred to as "smart" homes, hospitals, schools, transport, or even workplaces.

One of the first platforms that can be integrated in the field of AmI is called AwareHome [Georgia Institute of Technology, 1998]. It is a home that aims to help older people in their daily lives. We are here in a sub-domain of AmI, called Ambient Assisted Living. In this platform, two issues are addressed: the interactions between the inhabitants and their environment, and the development of tools to monitor the activities of the inhabitants, in order to help them. In these interactions, there is for example the display of information helping the user to do his daily tasks, questions stimulating the memory, or sending information to the inhabitants' family to inform them of their activities, and thus reassure them. Through this last point, the aim is clearly to help a person of advanced age to stay at home, and the family to be informed and reassured to avoid putting this person in a specialized institution. Cameras are placed in

different rooms of the house in order to analyze the activity of the inhabitants.

Some platforms focus on specific issues. ALADIN [Nakashima et al., 2010], for example, aims to observe the psychological effects of light in order, among other things, to improve sleep cycles in the elderly. Here, the main aspect of AmI, that of helping people in everyday life, takes on its full meaning and offers services that connected objects cannot provide on their own.

Platforms are mostly focused on the home, but are not limited to it, they can also be made for education, to control the classroom [Ramadan et al., 2010] or for students to attend a remote learning course [Shi et al., 2003]. We can also cite workplaces, with the emergence of Ambient Assisted Working [Bühler, 2009], which focuses on workplace adaptation for elderly and people with disabilities, as well as well-being and health care for all workers [Pancardo et al., 2018].

Other platforms focus instead on energy savings. One of the oldest and best known, MavHome [Cook et al., 2003], aims to maximize the comfort of residents while minimizing costs. In 2003, connected objects such as connected roller shutters did not yet exist, so objects had to be made by hand to make up for this lack. Thus, stepper motors from 5 1/4" floppy drives were diverted from their original use to control shutters. The goal was to predict user actions, using Markov models combined with a frequent event discovery algorithm, which are sets of actions that the user does every day or every week for example.

There are also test platforms that are not environments per se, but can be integrated into a physical environment. CASAS [Cook et al., 2013], for example, consists of a box containing various sensors, for temperature or door opening among others, and a computer processing the data from these sensors. The different objects in this box must be placed in predefined rooms. The aim is to make the recognition of predefined activities, where the goal is to find the activity that a user performs in the environment among a choice of activities defined in advance. Predefined activity recognition is an important sub-domain of learning in an AmI context, and is explained in section 1.4.3.2.

More recently, "Intelligence of Home" [Gomes et al., 2019] is a platform offering several services, including counting people in the environment or automating light and room temperature to reduce energy consumption. Through a web interface, the user can define his preferences in terms of temperature and brightness. The system will have to control the shutters, the brightness of the TV or the light bulbs in the environment to meet these preferences, while reducing electricity consumption. So these are basic interactions, but a starting point in the field of AmI.

Finally, other test platforms are available to host experiments in an AmI context. This is the case of Amiqual4Home [Inria, 2013], located in Grenoble, France, in which two databases were built for the recognition of predefined activities: Orange4Home [Cumin et al., 2017a] and ContextAct@A4H [Lago et al., 2017].

1.4.2 Interactions

When we look at work in terms of user interactions in AmI, we can distinguish two forms of interactions: explicit interactions, where the user communicates with the environment via a dedicated interface, and gives instructions, and implicit interactions, also called "context awareness", where the environment constantly observes the user, and, depending on his behavior, responds to his needs.

Relation	Symbol	Symbol for inverse	Pictoral example
X before Y	<	>	$XXX\ YYY$
X equal Y	=	=	XXX YYY
X meets Y	m	mi	$XXXYYY$
X overlaps Y	o	oi	XXX $\ \ YYY$
X during Y	d	di	$\ \ XXX$ $YYYYYY$
X starts Y	s	si	XXX $YYYYY$
X finishes Y	f	fi	$\ \ XXX$ $YYYYY$

Table 1.1: Allen's thirteen temporal relationships between events. Taken from [Allen, 1984].

1.4.2.1 Context awareness

The purpose of context awareness in AmI is to define contexts from the data of the connected objects. Here, the context can be defined as "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." [Abowd et al., 1999]. Once defined, the actions of the AmI will be done according to the recognized context.

This area is closely related to the intelligence layer of the AmI, as the system must understand user actions from the environment data in order to respond to them in return. Here, therefore, several works in this area are detailed.

First, work has been done to characterize the context, and more specifically the notions of time and space. Formalisms have been designed, such as Allen's temporal formalism [Allen, 1984], representing the temporal relationships of events in an environment. Table 1.1 describes this formalism detailing all the possible temporal relations between two events, X and Y. This formalism can be used to represent a complex situation composed of events occurring over time, such as user actions, or data from connected objects. [Randell et al., 1992] proposed a measure called Region Connection Calculus for estimating the proximity between two spatial regions. More and more research is integrating these notions into the reasoning aspect of these intelligent environments [Augusto and Nugent, 2004; Gottfried et al., 2006].

One of the most widespread application areas for context awareness in AmI is Ambient Assisted Living. Here, the environment must be able to provide assistance to the occupants in dangerous situations.

To illustrate, let us take the example of the system proposed by [Keshavarz et al., 2006], which can detect falls for elderly people. These falls are detected through several objects scattered in the environment, such as accelerometers and cameras. With scene analysis tools, a person's falls can therefore be automatically detected. In this case, a direct line to the emergency services is set up with a badge worn by the elderly person, thus enabling rapid action to be taken in these dangerous situations. In this way, the environment, without explicit user interaction, can provide assistance depending on the context, interpreted from data of connected objects.

Still in the field of Ambient Assisted Living for the elderly, [Blasco et al., 2014] offers a per-

sonal assistance system. The machine, named "e-servant", is linked to the connected objects, and proposes one interface among several, chosen according to the computer skills of the individual. Without the user having to control this machine, it can give useful information to the user, such as the time remaining for a washing machine, or warn him/her of dangers, such as the presence of smoke in a room.

[Tang et al., 2016] proposes a system that helps people with Autism Spectrum Disorder to do predefined daily tasks, such as cleaning or cooking. To help these people, the AmI system relies on location sensors, placed at strategic points in the room, such as the fridge. This makes it possible to see how long the person stayed in front of these places. Also, a smartphone interface can be used to display the tasks required to perform a complex action, such as cleaning, and sends reminders if the system infers that the user is having trouble doing the task. Unlike the previous examples, this system can also be monitored and controlled by an external person in charge of the person with autism.

Finally, other, more general systems are placed in the context of awareness. [Meurer et al., 2018] proposes a system defining contexts, and controlling the environment from these contexts, and taking into account user preferences. This system relies on a neural network to control the environment automatically. As mentioned in this paper, neural networks must be trained before operating, and they do not have a lot of data to process, because only the data from one environment are taken into account in this experiment. Thus, expert knowledge is put at the input of the system to reduce the training time of the neural network. An example of this *a priori* knowledge can be in the form of a rule, such as turning on the light in a room if a user is there. This network then trains in a supervised way, by predicting the actions to be taken on the connected objects, and comparing these predictions with the actions taken by the users. A conclusive experiment has been done to switch on the lights automatically according to the context defined by the objects' data.

Although promising, this last approach raises several issues regarding the use of neural networks for context awareness. First, how to reduce the learning time of the neural network to a minimum? Indeed, the learning period of the experiment presented is two weeks, and only to be able to control lights in an automatic way. Second, can we make such system without expert knowledge? After all, the system works with expert knowledge during its initialization. Furthermore, the architecture is based on assumptions, such as the one that the user will be absent most of the time from the environment. How can the neural network control several devices at the same time, based on high-dimensional data? How can the network adapt to changes in objects in the environment? But above all, how can we understand the reason why this system made a particular decision, such as turning on the light? Indeed, the neural network being a black box, the contexts established in this structure will not be understandable by the users. This kind of information can help control this AmI system. Thus, even if these systems can be efficient to learn contexts and answer some problems, they are not yet completely adapted to the AmI domain.

1.4.2.2 Explicit interactions

To control an AmI system, several choices are possible. [Gomes et al., 2019], mentioned above, uses a web interface. Also, the preferred means of controlling connected objects are through mobile applications. They often make it possible to control connected objects of the same brand. Examples of applications are "Philips Hue" for light bulbs [Philips, 2019], or "Somfy MYLINK" for shutters [Somfy, 2019].

However, work on new modes of interaction is ongoing. [Nazari Shirehjini and Semsar, 2017] is exploring the idea of recreating the physical environment virtually in 3D, in order to

Figure 1.4: Screenshot of the Smart AR Home application, showcasing a connected bulb

more easily identify and control the connected objects present. It is also possible to use newer forms of interaction. For example, "Smart AR Home" is an application for Android based on augmented reality [Melnick, 2018]. Augmented reality allows to integrate 3D objects in real time in a real environment, for example through the camera of a phone. Here, there is no need to recreate the whole environment in 3D. To add a connected object, simply point the phone camera at the object in question. It will then be displayed in 3D, and show its settings (figure 1.4). It is thus possible to browse the environment, and to control objects through the application.

Finally, an additional mode of interaction is developing more and more, and could be integrated in the field of AmI: robots. This has been addressed by [La Tona et al., 2018], which proposes a robot in the framework of Ambient Assisted Living. The user can interact with the robot in three ways: voice via microphones, gestures via a camera, and touch, via a tablet contained in the robot. At the user's request, the robot can thus control the objects in the environment, but it can also help the occupant in case of a problem by calling for help.

Interaction with robots is a field in itself, which is in full development. Work is being done, for example, to make these robots more expressive [Balit et al., 2018], to avoid them hindering the user's path [Paulin et al., 2018], or to learn behaviors based on interactions with their users [Galdeano et al., 2018]. The idea is to make these robots more social, which can bring a more natural interaction. In this way, they could thus personify the AmI system, and represent the primary means of communication.

1.4.3 Intelligence

The intelligence dimension is the heart of AmI. It is this layer that will define the purpose of an AmI system, whether it is to help people in need with Ambient Assisted Living, to recognize activities or discover new ones, to secure the environment or to save energy. We previously talked about the sub-domain of context awareness, in section 1.4.2.1. Here, we showcase

some sub-domains, first detailing areas that are not very or not related to the thesis, and then explaining the sub-domain in which the thesis is positioned: activity discovery.

1.4.3.1 Anomaly detection

The detection of anomalies is an important security issue for AmI, even if relatively few works have been done yet on this matter in ambient environments [Acampora et al., 2013]. This can be done via algorithms which profile the signals of the connected objects. They then group these profiles together in order to have a typical profile of the environment, and thus detect changes that may be anomalous [Jakkula and Cook, 2008]. This can also be done by more specific algorithms, such as creating a system that detects unusual noises, like broken windows, to identify intrusions [Duman et al., 2019]. Also, specific use cases have been made for elderly people in the context of Ambient Assisted Living in addition to fall detection, such as [Tran et al., 2010].

We see that this type of research can be useful for the average person, but even more so for the elderly, the disabled, or those who can no longer be autonomous. This makes it possible to assist these people when they need it most, when they are the most vulnerable.

1.4.3.2 Activity recognition

Activity recognition is a central domain in AmI. The goal here is to know what predefined activities users are doing in the environment. Thus, it falls within the range of supervised learning. Recognizing activities can have several use cases. For example, it is possible to define whether a person is available to receive a call based on his/her activity, as in [Cumin, 2018].

Two approaches are possible for the recognition of activities: knowledge-based and data-driven approaches [Chen et al., 2012a]:

- Knowledge-based approaches are based on ontologies to describe the situations to be recognized. Activities are described with properties related to constraints on data retrieved by connected objects. The sensor data are then compared with these descriptions to find the corresponding activity. Several studies have been done on this subject. For example, [Gu et al., 2004] proposes a formal context model used for, among others, semantic context representation and context reasoning. On top of that, it proposes a software to build services upon this model. [Chen et al., 2012b] also proposes an explicit modeling of the context and the activities in the environment, for activity recognition, with a 94.44% accurate recognition of certain activities. [Ye et al., 2015] enhances the previous concept with "Knowledge-driven approach for Concurrent Activity Recognition" (KCAR), a model for recognizing several activities that can take place at the same time in an environment, with a 91% accuracy. In these cases, they are therefore symbolic systems, explained in section 1.3.2.2. Thus, even if these systems perform well for activity recognition, the problems mentioned in section 1.3.2.2 still remain, and activity recognition is still limited by the representation of the context. Moreover, if such systems are to be used by users, each context will have to be modeled by hand, which makes the installation of the system cumbersome.

- Data-driven approaches fall into the field of supervised learning, where the algorithm must learn a prediction function from annotated examples. In this case, it is a history of the data of the connected objects, accompanied by the names of the activities that the users were doing. This approach is the most widely used in the field of activity

recognition, with a lot of work on this subject. Several AI techniques have been used to do this, such as classification techniques [Stikic and Schiele, 2009], Hidden Markov Models [van Kasteren et al., 2011], or more recently neural networks, with [Singh et al., 2017].

More original research projects are attempting to integrate several existing techniques into a more complex system in order to improve performance. For example, [Cumin et al., 2017b] proposes a system composed of several parts. The first one is an activity recognition model per room, allowing to make an estimate of the activity that the user would be doing in this room, accompanied by a measure of certainty for each room. The second one is a system gathering all these results, then determining the most plausible activity with regard to the certainty values.

In our case, we want to discover the regular situations of users in order to offer them automation, which does not really fit with the definition of activity recognition. We do not want to identify pre-established situations, we want to discover the custom user situations, in order to offer truly personalized automation. That is why this thesis does not apply to this field.

1.4.3.3 Optimization problems

A lot of work in AmI can be considered as optimization problems. For example, we have already mentioned systems made to save energy, in section 1.4.1: [Cook et al., 2003; Gomes et al., 2019]. To this we add [Gil-Quijano and Sabouret, 2010], which uses a reinforcement algorithm to predict the activities the user will perform. This is a derivative of the recognition of predefined activities: all the activities are defined by hand, and the system tries to predict the possible activities of the users, in order to optimize the heating in order to reduce the overall consumption, while maintaining comfort.

Other objectives can be formalized as optimization problems, particularly in the field of Ambient Assisted Living, which sometimes requires quick reactions to help occupants. For example, fall detection for the elderly [Keshavarz et al., 2006], or the optimization of lights to improve sleep cycles.

1.4.3.4 Activity discovery

The discovery of activities, complementary to the detection of activities, is the other central domain of AmI. The aim here is to discover regularities in the environment, through the data reported by the connected objects. Unsupervised learning tools are most often applied here because the regularities to be sought are not known in advance. Several studies have attempted to develop systems to predict the actions of people in the environment. It is thus a question of discovering patterns or prediction rules from the data reported by the connected objects. Since 1995, learning systems have been created using neural networks, trained to find the next action of the user, according to its history. The best known are [Cook et al., 2003], using data compression and Markov Models algorithms, and its evolution [Jakkula and Cook, 2007], taking up Allen's temporal relationships [Allen, 1984].

During his thesis, Sébastien Mazac built a sensorimotor learning system in a context of AmI [Mazac, 2015]. To do this, he was inspired by the constructivist paradigm, which has been detailed above. He created a multi-agent system, i.e. a system with several processes interacting in parallel with each other for a common purpose. Several agents that cut, compare and associate pieces of signals allow the system to learn some of the consequences of value changes reported by connected objects. This system is therefore in the bottom-up paradigm

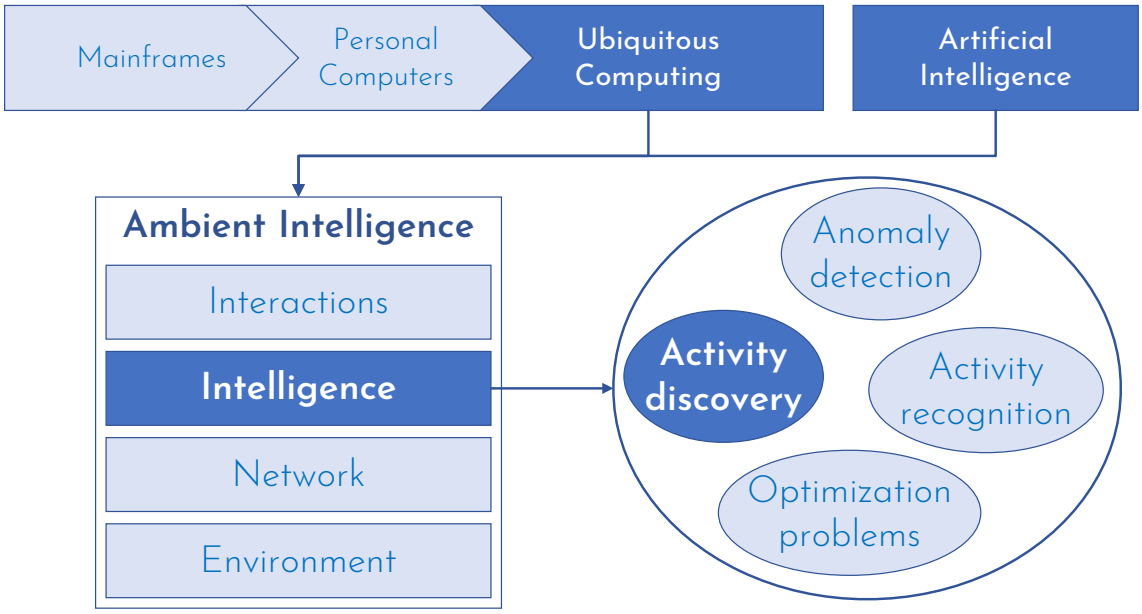


Figure 1.5: Positioning of the thesis in the AmI domain

of AI. Through its inspiration and architecture, the system learned basic relationships between connected objects, and could adapt over time. Even if activity detection is a higher level concept than sensorimotor learning, Sébastien Mazac's work is part of this approach.

Other works use tools to mine patterns or prediction rules, which will be detailed in the following chapter as we precise the research questions of our AmI system.

1.5 Conclusion

In this chapter, we were able to clearly express the stakes of the thesis. By focusing on UbiComp, we have demonstrated that the advent of the Internet of Things is no coincidence, and we have listed the challenges that remain to be addressed: interoperability, ease of use, personalization, privacy and security. To address some of these challenges, we propose to create a service orchestrator, which observes users' habits to provide them with customized automation. Knowing how to recognize habits requires intelligence, so we became interested in the field of AI, which is gigantic, very diffuse, and includes many issues. It also includes many data processing techniques on which the thesis will be based.

We finally saw a field that merged the two previous ones, namely UbiComp and AI: AmI. AmI aims to help people in their everyday life in a proactive and reasonable way, and comprises several dimensions, which can be put together in layers: the environment populated by connected objects, the network to exchange information, the intelligence to act proactively, and the interactions with users. This thesis is positioned in the intelligence layer of AmI, and more precisely in the sub-domain of activity discovery. To illustrate what has just been stated, figure 1.5 shows the overall positioning of this thesis.

We therefore want to create a system, which, from various connected objects, analyses the data from these objects to find habits and thus provide automation, in order to make these objects work in synergy. Now that this major issue has been defined, we can focus on the problems that arise from the design of such a system, which is the core of chapter 2.

Chapter 2

Research questions

2.1 Introduction

In the previous chapter, we have set the context for the thesis, and we have listed the main issues to be solved in the Internet of Things: interoperability, ease of use, customization, privacy and security. To address these challenges, we concluded with the idea of a service orchestrator providing automation based on user habits. For example, the AmI system can automatically turn lights on or off, depending on the presence in the rooms, or turn on a specific radio channel based on the situation and habits of users. The first basic view of this system is shown in figure 2.1, where we imagine an AmI system receiving data from connected objects. Then, based on this data, the AmI system decides to send actions on these objects to act on the environment.

The objective of this chapter is to raise the issues to be addressed to develop such a system. This allows us to clarify the vision we have of our AmI system. Indeed, a system offering automation to users brings together several different domains, such as ergonomics or AI. The problems of this system are therefore at the crossroads of these fields, and can therefore be complex and numerous.

It will lead us to the scientific problem of this thesis, mentioned in the introduction: we have, in a physical environment, one or several persons, and several connected objects, returning data over time, without necessarily a fixed sampling frequency, and which can be

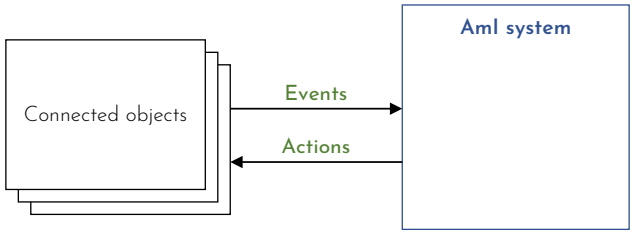


Figure 2.1: Abstract view of the AmI system. The first idea is to have a system that receives the data from the connected objects and controls them

quantitative or categorical. How can we analyze these heterogeneous, time-stamped data, coming from multiple sources, and without *a priori* on these data, to observe regularities and predictions, in order to find habits in this environment and to propose automation to users in a context of service orchestration?

Thus, to simplify the understanding of this chapter, the issues are structured in three subdomains: the users of the system, the place on which the system acts, i.e. the environment, and finally the purpose of the system, i.e. the automation to be provided.

2.2 Users

The AmI algorithms developed in this thesis have the sole and only purpose of serving users. This problem alone will impose constraints on the rest of this thesis. In addition, the risk of losing control of the environment and the one of losing privacy is part of people's fears about AmI [Ben Allouch et al., 2009]. In general, for this system to be able to provide custom automation, it must:

- Be useful
- Be easy to use
- Be personalized
- Respect the privacy of its users

Let us first ask ourselves a simple question: how does the AmI system provide this automation?

2.2.1 How to provide the automation?

If we refer to the original vision of UbiComp, cited in section 1.2.3.1, we can imagine a system that controls the environment in a completely automatic way. It operates without interaction with users and without informing them, and it decides what actions to take to help users in their daily lives. However, it is quite easy to imagine the dangers that this algorithm represents: imagine it leaving the fridge open for hours, turning the heating on all the time or leaving the gas on when there is no pan on it. Add to that an infant and you have a disaster. To avoid these terrifying scenarios, three ways are possible:

- Completely model the environment *a priori*, and define all the dangers to be avoided, which can take a long time to do on all environments.

- Give control of the system to the users.
- Have a strong AI (section 1.3.2.1) deducing from itself the dangers to be avoided. Current AI tools do not allow this second possibility at this time, but we can imagine that this may be the case in the future.

In addition, some habits that can be found in an environment are not necessarily acceptable as automations. Let us take the example of a child who watches television programs after school. The AmI system can thus observe this frequent scenario, which is a verified and therefore relevant habit. However, his parents, who are the main users of this system, would not find it useful to automate this, since they would rather see their child do his homework than watch television shows. This notion of **usefulness** in automation, not to be confused with the relevance of the found habits, must therefore be materialized by learning the goals that users have for the AmI system. In addition, interactions with users should not be too numerous, to avoid a feeling of irritation from users, resulting from a lack of interest in using this system. This can be accomplished by this notion of usefulness, among others.

Finally, since the users of the AmI system are not machines, their actions, and therefore their habits, will change over time. The AmI system must therefore adapt to these changes, identifying these new habits and dropping those that are obsolete.

To sum up, several major problems are already arising:

- Take into account the desires of users
- Avoid dangerous situations
- Adapt to change in user habits
- Reduce interactions with users, to avoid their lack of interest

In this thesis, we will not consider the *a priori* modeling of the environment. Indeed, this representation is by nature limited, and therefore less flexible to adapt to potential changes, which is incompatible with the prerequisites of the system, i.e. adaptation to change. In addition, it can be laborious to implement. We prefer to imagine a system that can adapt to any environment, without knowing it in the first place. With this in mind, we identify three levels of complexity for an AmI system:

- Stay at a basic level of control for the AmI system and leave the hand to the users. To do this, the AmI system makes **automation proposals** to them.

  This can be painful for users, as they have to accept or reject these proposals, which requires a lot of interaction. However, it also helps to avoid dangerous situations, as it is the user who has his hand on it. The system remains intelligent in that it can describe habits from connected objects data. It is also possible to optimize the search for habits by estimating the notion of usefulness among users as they interact with the system.

- Personify the AmI system with a conversational agent, like a **butler**.

  Interactions would thus be reduced, the concepts of danger and usefulness could be integrated *a priori* into the AmI system or built through communication with users. In the same way as a real person, the butler can decide on his own to automate certain actions, if he considers them useful for users, and if they do not represent a danger. Also, it can keep users informed of the automations put in place, with the possibility for them to deactivate them. This requires algorithms that are more oriented towards

strong AI, to allow communication with users to understand their wishes and adapt its behavior according to them, which is not possible with current conversational agents.

- Have an AmI system completely **melted into the environment**.

  The AmI system is not personified, and acts as a background. It has general knowledge, which allows it to provide assistance to users through connected objects. It can be adapted to the goals of users according to its internal representation, through the rare interactions it has with them. For example, if the AmI system closes a door and a user opens it immediately after, the AmI system concludes that its action was unwanted, searches for the cause, and changes its behavior accordingly. In this last conception of the system, we return to Mark Weiser's original vision of AmI, where computing acts in the background (section 1.2.3.1).

In this thesis, we will take the first step, because the last two require further progress in the field of AI. This approach obviously has its share of problems.

- It is crucial to display the automation proposals in a way that is **intelligible** to users.
- Users must be able to **control the system through interactions**, to validate proposals, to make the AmI system easier and more pleasant to use, and to set limits to ensure privacy.
- It is important to try to **understand the goals** of the users for the AmI system. This optimizes the search to avoid showing them too many rules they consider unnecessary, and thus reduce the lack of interest they may have using this system.

Regarding user interactions, it would be interesting to design a user interface. This is outside the scope of this thesis, but it is possible to imagine one easily. To do this, simply take the interface of an existing orchestrator, such as IFTTT, that allows the manual creation of automations, and then have a section below it containing the automation proposals, in the form of suggestions, in the same way that an online sales site would. Further interactions with users will be detailed in the following chapters.

2.2.2 In what physical form will the AmI system be?

It is important to ask the question of the physical form of the AmI system. Indeed, it will set constraints to be respected for the rest of the thesis. This problem is strongly linked to the issues of security and confidentiality. Indeed, it is easy to imagine improper uses of an algorithm that scrutinizes an environment and acts on it: spying, sabotage, user profiling...

First, the connections that can be made on these objects, i.e. receiving their values and sending them actions, must be secured. This avoids unwanted monitoring and action on the environment. To do this, the objects themselves must be secure, and so must their communications. Even if this problem does not fall within the scope of the thesis, it is possible to imagine a solution to this problem: the AmI system acts as a firewall, controlling all communications of the objects to which it is attached. To do this, the AmI system must also be secure.

In addition, it is also necessary to secure the data of the AmI system, i.e. users' habits. They represent private knowledge about them, and this issue of privacy has been a growing issue in recent years [Cadwalladr and Graham-Harrison, 2018; Ng, 2019]. For example, it is for this reason that the European Union has adopted the General Data Protection Regulation

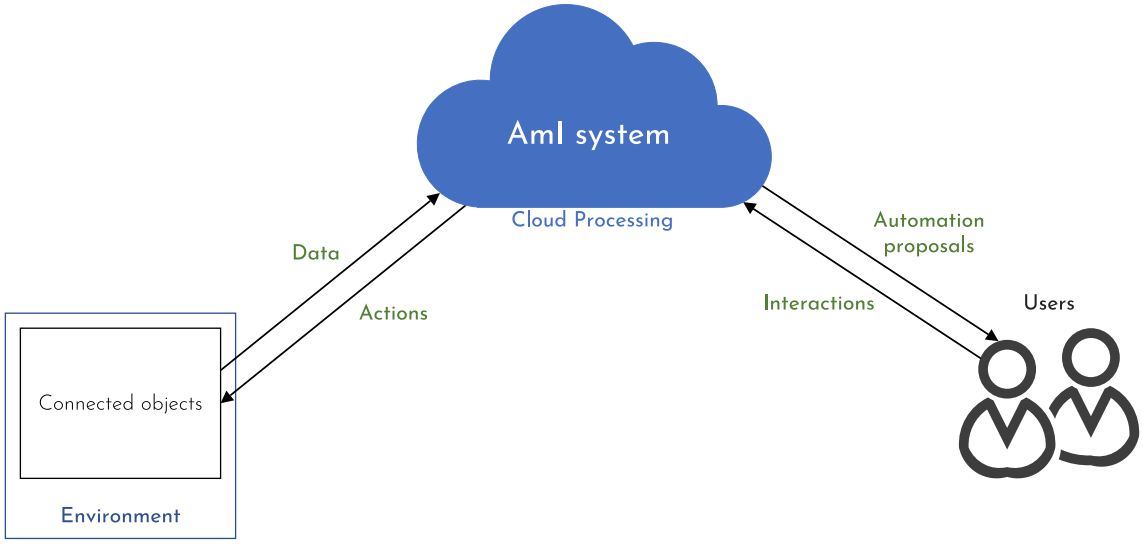


Figure 2.2: Representation of an AmI system operating in the cloud

(GDPR), which requires IT companies to give users more control over their personal data [European Parliament, 2016].

So, considering that, how can we imagine the AmI system? The main trend today is to build applications for "the cloud", i.e. running on remote computers connected to the Internet. Many cloud applications exist that allow to store files, like Google Cloud, OneDrive or Dropbox, work on office suites with Office 365 or Google Docs, etc.

2.2.2.1 Cloud processing

How can we imagine an AmI system in the context of the cloud? We can initially take inspiration from online sales site. The system has access to the data of all users at all times. From this data, the AmI system forms groups of users who are similar in their habits and makes automation proposals to them. Users can thus accept certain proposals, which will allow the AmI system to act remotely on connected objects in their environment. This process is illustrated in figure 2.2.

Making such a system would be complex, as it will be very difficult to cross-reference data from several environments, due to the diversity of connected objects, their location, and the environments themselves. Let us consider, however, in our case, that this is possible.

On the other hand, it is also possible to process only data from a single environment in the cloud. Thus, each AmI system would be made for a single environment, there would be no user groups considered similar in their habits.

What does it bring? Since the AmI system is available on the Internet, users can act on it outside the environment. It is possible to display new automations for users, because they do not come from their data, but from those of the group. Also, the AmI system would be fast, because it would run on computers with enormous computing power, even more powerful than those usually found in homes.

What are the disadvantages? If the Internet no longer works in the environment, the AmI system can no longer act on the environment, and therefore becomes inoperative. Also, the energy cost of a design in the cloud can be high [Elegant, 2019].

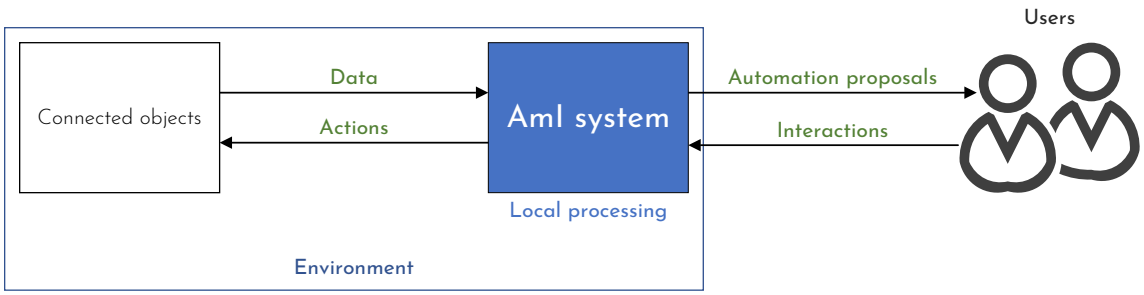


Figure 2.3: Representation of an AmI system operating locally

What is the main danger? The lack of privacy. Companies offering this type of service can use the data to their advantage, to better offer advertising [Cuofano, 2018; Rushe, 2020], at worst to do mass surveillance [Mazzetti et al., 2019]. The purpose of the AmI system would therefore no longer be to help users in their daily lives, but to guide them towards something else, such as new products to buy.

It is also possible that these companies, in good faith, prove that they do not have access to the data, by not grouping people and by treating each user independently of others. However, the data of all users would remain concentrated in the same place, thus becoming a single point of attack for hackers.

2.2.2.2 Local processing

The alternative to remote treatment is to make a local AmI system, i.e. running on a computer integrated in an environment, as shown in the figure 2.3. As the data processed are sensitive, we can consider that the AmI system only processes the data from the environment in which it is present. It is also possible to picture an evolution of this concept in the form of a distributed processing between the connected objects present in the environment. Thus, data would no longer be processed by a central computer, but rather by the objects in the environment.

What does it bring? First of all, the AmI system is resistant to internet outages. If the Internet no longer works, the AmI system still works. Then, the habits found by the AmI system will really be adapted to the users of the environment, because they will be the result of their data alone. Finally, *a priori*, the main strength of this proposal remains the respect for privacy. Indeed, the data is stored and processed on a machine in the environment, and not on remote computers. Thus, it is possible for a company to offer an AmI service, i.e. the algorithms and why not the machines on which they will run, without the need to access the users' data.

What are the disadvantages? The computer in the environment must have sufficient computing power to run the AmI algorithms. It is therefore necessary to have efficient algorithms, and more powerful machines. This must be combined with the need to be energy efficient, as these computers would be constantly running. Secondly, there will be, at least for the time being, no automation proposals coming from other users' data, unlike processing in the cloud. Sharing of data and habits remains possible, with the agreement of users, to extend learning to a neighborhood for example. Finally, in principle, users can only interact with the AmI system if they are physically in the environment. However, it is quite possible to provide a remote interface via the Internet, so that the user can act on the AmI system from anywhere.

What is the main danger? Security. In the same way as for the cloud, it is imperative to secure the computer on which the AmI system is installed, the connected objects, as well as the connections between them. In addition, a constant update system must be put in place to protect machines from future attacks.

2.2.2.3 The choice retained in this thesis

Each solution, whether in the cloud or locally, has its advantages and disadvantages. However, the issue of confidentiality must be taken into account when developing such a system. More and more companies are positioning themselves on this subject, such as Apple, which favors local data processing on their machines when possible [Apple, 2017]. This issue is so great that it is necessary to explore the view of local treatment in the field of AmI. The combination of this issue with the other advantages of local processing, like running without the need for the Internet, has therefore pushed us in this direction. Providing a local and secure AmI system, rather than a remote system, that can be considered obscure to users, seems to be the best solution.

To summarize, the AmI system of this thesis is intended to be local, processing only user data. It must therefore be able to operate with relatively little data, and be efficient.

2.2.3 Summary

In this thesis, we seek to make an orchestrator of services bringing automation through observed habits. This section has allowed us to further clarify how the AmI system works.

To avoid dangerous situations that can be generated by a fully automated AmI system, we give control of the system to the users. The algorithm provides automation proposals, which the user may accept, or not. This involves several issues, including the need to be efficient, to make the found habits intelligible, to allow system control through interactions with users, and to understand users' objectives for this AmI system, in order to minimize their lack of interest.

In addition, to respect the privacy of users, we propose local data processing, not in the cloud, which means that the AmI system will have to process little data. Changes in user habits must also be taken into account. Thus, the system must adapt to it over time.

The overall functioning of the AmI system is fixed (figure 2.3), let us now talk about the environment in which it must evolve.

2.3 Environment

The environment, i.e. the physical place on which the system operates, is the central element of an AmI system. It is in this one, monitored and modified by the connected objects, that the algorithm evolves and searches for habits. In this section, we will therefore formalize the data we can collect, in order to explain our choices regarding the architecture of the AmI system detailed in the next chapter.

The previous section indicated the need for an efficient system, due to local processing, and the need to express habits in a way that is understandable to users. Thus, four global goals are identified in the consideration of the environment:

- Take into account all possible data from the environment to characterize habits

- Adapt to changes in the environment, i.e. additions, deletions, and failures of connected objects
- Be efficient in data processing
- Express habits in an understandable way

2.3.1 Definitions

First of all, what is an **environment**? It is a delimited physical place. It can have objects or living beings. It is defined by **properties**, like the temperature, force of the wind, luminosity, the state of a door (closed or opened), or the radio that is playing. In an environment, we can measure, hence monitor, those properties as **variables**. Not only can we monitor the environment, but we can also act on it. One can change the radio, close or open the door, or change the temperature via heating or air conditioning. Thus, an action in the environment can have an impact on the variables observed in it.

So how can we observe the environment and act on it? We can do it directly through our body, or use objects. There are an infinite number of objects: windows, hammers, video recorders, Minitels, washing machines, speakers, smoke detectors, plates... And all can either measure the environment, act on it, or both. In the scientific literature [Aztiria et al., 2010; Sanchez-Picot et al., 2016; Frey, 2013; Warneke et al., 2001], two categories of objects are defined: sensors and actuators. A sensor monitors a variable in the environment, and an actuator acts on the environment.

A **connected object** is an object that can communicate, i.e. send or receive data, with other objects. Note that a connected object is not necessarily directly connected to the Internet network. We can take the example of a watch that can only be connected to a phone via Bluetooth. In this case, it is not directly connected to the Internet, it is the phone that can share its information on the Internet. The term **Internet of Things** therefore refers to the network formed by all these communicating objects, and not to the Internet as we know it today.

A connected object shares all the properties mentioned above, except that in this context, we distinguish a particular type of actuators as interfaces. An interface acts on another connected object, while an actuator acts directly on the environment. So, for a switch acting on a lamp, the switch is an interface, and the lamp is an actuator. This distinction between interfaces and actuators is inspired by the notion of "actions" from [González García et al., 2017]. This will be useful for the rest of the thesis, as it avoids redundancies between the actions of the interface and those of the object it controls.

To summarize, in an environment, we have the following types of connected objects:

- A **sensor** monitors a variable in the environment, e.g.: a temperature sensor.
- An **interface** acts on another connected object in the environment, e.g.: a switch acting on a light.
- An **actuator** acts directly on the environment, e.g.: A light or a heater.

An object can be either elementary, i.e. with only one sensor, interface, or actuator, or composite, i.e. composed of several sensors, interfaces and/or actuators. A smartphone, for example, has a multitude of sensors, and can affect the environment by sending sound or images.

Regarding the variables that can be monitored, we distinguish two of them: quantitative and categorical.

- A **quantitative variable** has values that represent a quantifiable property of the environment, such as temperature (in degrees Celsius) or brightness (in lumens).

- A **categorical variable** has values that are not quantifiable, and represent distinct categories. It may be difficult to estimate a measure of distance between these categories without prior knowledge. For example, the status of a door (closed or open), or the radio channel being played.

- There are also connected objects that return more complex data, such as connected cameras that return a video signal, in other words vector data. The processing of this data is a field of research in itself, belonging to picture analysis. In addition, this is highly sensitive data from a privacy perspective. Thus, they will not be taken into account in this thesis.

As with the variables, we can identify two types of actions:

- A **quantitative action** sends an order in the form of a quantifiable value, such as the desired temperature for a heater, or an electrical current for a motor. In this case, the values are necessarily numeric.

- A **categorical action** sends an order from a list of completely separate possibilities. Here again, the distance between two categorical actions cannot be easily determined. For example, choosing a radio channel, or switching on a lamp.

It should be noted that quantitative action will not necessarily have an observable quantifiable impact on the environment, and vice versa. For example, we can increase the current intensity of a door closing motor. It is a quantitative action because it influences a quantifiable value, which can be transcribed by a categorical observation: a closed door.

It is also worth noting that it is possible to obtain environmental data not directly from connected objects. Indeed, data such as weather or traffic information can be provided by Internet services. In this thesis, we can consider that they also belong to the category of connected objects, because nothing distinguishes these services from objects, at data level.

2.3.2 Data

Before we look at the issues and the different areas that deal with data, let us make an inventory of the data that can be obtained.

2.3.2.1 Context

By definition, connected objects communicate, so they can send and receive data. For example, a temperature sensor may return the measured temperature at a given time, a door opening sensor may return the status of the related door, etc. Also, a radio can be turned on by data sent by another object, such as a switch. Here, we define these primary and unprocessed data as **events**. All the events sent by all the objects are gathered in a set noted E. For each event received, the system knows its source, i.e. the object that sent it.

Connected objects can share two types of events: monitored variables if they are sensors, and actions taken if they are actuators or interfaces.

Let us take the example of a room containing two connected objects, as seen on figure 2.4: a presence sensor, used to define whether a person is in the room or not, and a radio.

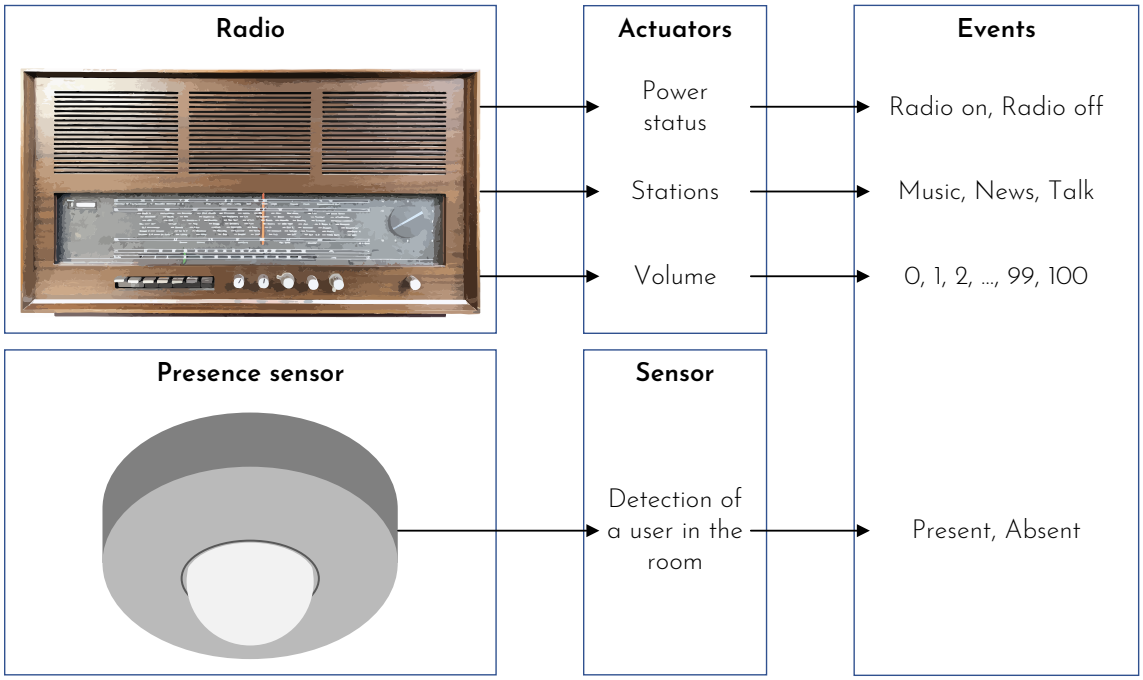


Figure 2.4: Environment example

- The presence sensor just detects if someone is present in a room, and cannot determine the number of people present. Thus, it can detect the following: "Present" and "Absent".
- The radio has three actuators:
  - Its power status can be: "Radio on" or "Radio off".
  - It can select one of the following stations: "Music", "News", "Talk".
  - It can change its broadcast volume between 0 and 100.

Therefore, the set of all events is E = {Present, Absent, Radio on, Radio off, Music, News, Talk, Volume}, with Volume $\in [0, 100]$.

All those events are returned by the objects over time. We can therefore set an occurrence time for each sending of an event by an object. We can even put all the event submissions of an object on a timeline, to have a history of the events sent.

This data representation is called a **time series** (see figure 2.5). It is noted TS=$\langle(t_1, I_1), ..., (t_n, I_n)\rangle$, $I_1, ..., I_n \subseteq E$, where:

- t_i is a **time stamp**. It defines the time coordinates of the occurrence.
- $I_i \subseteq E$ is called an **itemset**. It is the set of individual events of E which are observed at time stamp t_i.

The figure 2.5 is an example of a time series which represents activities of a user in the environment. Its mathematical representation is:

$TS = \langle$(10:00 am, {Present}), (10:44 am, {Radio on, Music}), (11:36am, {Radio off}), (1:57 pm, {Radio on, News}), (2:25 pm, {Music}), (3:41 pm, {Volume = 40}), (5:14 pm, {Radio off}), (6:05 pm, {Absent})$\rangle$.

As seen above, the connected objects return events of different kinds: temperature, radio volume, presence of someone in a room, etc. We have seen that there are two possible types

Figure 2.5: Representation of a time series

of events: quantitative and categorical. The structures of these formats are fundamentally different: there is no notion of distance between categorical events, as opposed to quantitative events. It is possible to have an order or a notion of distance between the different categories, with expert knowledge, but we assume in this thesis that we do not have any.

New algorithms, based on Deep Learning for example, are trying to learn features over data of different kinds [Ngiam et al., 2011]. However, they require a lot of data, and the operation of these neural networks is not intelligible (section 1.3.3.1), which makes it inapplicable in our case. We must therefore assume that we do not have a tool powerful enough to adapt to all possible data. Thus, the AmI system must process quantitative events differently from categorical ones.

2.3.2.2 Categorical data

How can these data be taken into account? There are two complementary ways to take this data into account when searching for habits:

- Observe the **changes** in this data, i.e. the transition from one event to another that is different from the previous one. Let us take the example of a change from an event a to an event b, which can be formalized as $a \rightarrow b$. Several pieces of information can be taken into account in a change, in particular:
  - Only the end value, b
  - Only the start value, a
  - The complete change, i.e. the transition from a to b

  Those information pieces are complementary, and can be used to describe habits. Such an example would be: if the light turns blue, then turn on the heater.

- Take into account the **stationary states** of these data. A usual example would be: if the door opens and the heating is on, then close the window.

The first category, i.e. taking into account changes in data, and more precisely the information concerning the final value, are fundamental for our subject. Indeed, automation implies at least one change of state at the input and output. For example, if the user opens the door, turn on the light; if the fridge opens, then turn on the light in the fridge, etc.

The second category is complementary to the first, as it provides contextual information, making it possible to specify the conditions under which habits take place. However, it may involve a very high density of data to be taken into account. This can significantly increase the time required to operate the system in question. Indeed, if we take a snapshot of the environment at a given time, we can observe some changes in the connected objects, and many stationary states in all the others. Therefore, for each snapshot of the environment, it is necessary to take into account the data of all objects, instead of only those which change.

Taking into account the notion of negation It is also possible to take into account the categories that are not selected, in other words, to take into account the notion of negation. After all, if one category has been selected, the others are not. This only makes sense if at least three categories are possible among the events sent by the connected object. Otherwise, with one category acting as a mirror of the other, taking into account negation would simply be redundant.

This provides more ways to characterize habits to look for. Let us take the example of a connected window, which can have three states: open, ajar, and closed. We can imagine that, from the moment the window opens or becomes ajar, the room temperature drops. Thus, we can imagine the following rules stipulating that if the window opens, the temperature drops, and if it becomes half-opened, the temperature drops too. But a more interesting rule would be the following: if the window is no longer closed, then the temperature drops. Such rules could be found through a hierarchical description of the data. Thus, we would have two categories, such as "closed window" and "open window", with two sub-categories: "window ajar" and "window fully open". However, this description must be made *a priori*, with expert knowledge of the data.

This can simplify the number of rules that the system finds, and therefore displays to the user, which is in line with intelligibility. Taking into account negation would certainly be beneficial for the search for rules but it would lead to either *a priori* on the data, or too dense data to be processed, because it would be necessary to constantly take the status of all possible categories. Thus, these data are not essential to make a functional AmI system, but will be very useful for its evolution.

Redundancies In general, it will be necessary to remove redundancies in categorical data. As a reminder, a connected object gives the state of an environment at a given time. They send notifications of environmental states, not changes. It is the difference between an event and the previous event that indicates a change, or not. Thus, if we take the example of a door opening sensor, if the sensor returns the event "door open" 5 times in a row, it does not mean that the door has been opened 5 times. This would require an alternation of "closed door" and "open door". Thus, to avoid misleading the system, it is useful to remove redundancies in the events returned by the objects.

For some objects, each data sent can be considered important. For example, a smartphone can send as data the fact that a new message is received. For these types of objects, the data sent is the notification of a change. It is thus possible to make an exception for this type of object, by not deleting what is considered to be redundancy, thus taking into account each data transmission. However, since these objects are in the minority, we will not take them into account in this thesis.

Summary The basic AmI system must take into account changes in categorical data. In its evolutions, it can also take into account stationary data, in order to have contextual information, and the notion of negation to potentially reduce the number of rules to be found. However, both these developments require more data to be processed.

2.3.2.3 Quantitative data

Integrating quantitative data into an analytical system or AI is a very complex area in itself.

In our case, it is necessary to clarify some concepts regarding input data:

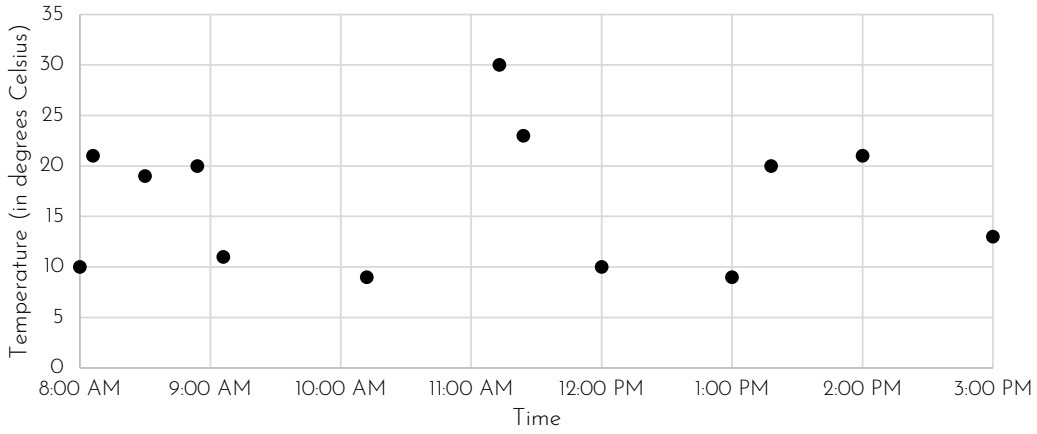


Figure 2.6: Example of quantitative events from a temperature sensor

- Basically, the AmI system has no idea of what kind of data it has in input. It does not know if a sensor captures temperature, humidity or sound, if the data from a switch turns on a light or closes a shutter. With this information, it would have been possible to apply algorithms adapted to this data. For example, knowing that an object is a microphone can induce using speech processing or sound recognition algorithms. It knows neither the scale of values nor their meaning, other than by the name of the connected object, which can be difficult to interpret. Thus, on this point, the AmI system is completely agnostic.

- The arrival times of the data depend on the objects that send them, as shown in the example in figure 2.6, mainly for energy saving reasons. We are rarely in a situation where a fixed sampling frequency exists, as in audio files. The AmI system is completely dependent on the object from which it receives events.

- The AmI system will need to process this data to identify habits, but also to execute automation online, i.e. treating the events as they go along. Thus, if there are processes to be done on the input data, they will have to be done online.

Let us take an example of a temperature sensor, sending events within a day, in figure 2.6. To unify quantitative and categorical data, and thus process them homogeneously, we can either convert all data into categorical data or into quantitative data.

In our case, we must transcribe the automation proposals, based on the data, in a way that is intelligible to users. Converting data into categorical data can make a result more intelligible than converting it into quantitative data. We will therefore discretize quantitative data, i.e. convert them into categorical data. How to do this? Several ways are possible:

- Consider categories as **value ranges** (figure 2.7). For example, if a temperature is less than 17, it is in the "cold" category, between 17 and 25, in the "comfy" category, and if it is greater than 25, in the "warm" category. This discretization is very easy to understand, but requires expert data to determine the boundaries of the categories.

- Consider **basic variations** in the data, according to several of their characteristics (figure 2.8). This can be done by smoothing the signal, to remove very small variations considered as noise, and then grouping the variations that are considered to be similar. It is possible to understand this discretization because it refers to simple variations in the signal. Cleaning the signal to keep only great variations in terms of amplitude can

be done with signal processing algorithms [Kay, 1993], if these data have a fixed sampling frequency. For example, a low-pass filter can be applied to audio data [Porle et al., 2015]. If the data do not have a fixed sampling frequency, the signal can be cleaned by segmentation algorithms, such as the Top-Down algorithm, the Bottom-Up algorithm and the Sliding Window algorithm [Lovrić et al., 2014]. Grouping variations together can be done via clustering algorithms, such as K-means, DBSCAN or Hierarchical Clustering [Madhulatha, 2012].

- **Profiling** the signal, to directly find similar signal pieces (figure 2.9). This can be used in particular for anomaly detection, where we look for signal fragments that differ greatly from the usual observed profile. Systems, recently using neural networks, make it possible to search for them [Baccouche et al., 2012; Bascol et al., 2016]. Furthermore, tools can be applied to compare two portions of a signal, such as Dynamic Time Warping [Berndt and Clifford, 1994], or to group portions of signals, such as self-organizing maps [Kohonen, 1990] or growing neural gas [Fritzke, 1995]. But, even if these systems are powerful, the signal pieces identified as repetitive may be more difficult to interpret than basic variations.

It is also possible to develop the basic techniques mentioned above, in particular by integrating user feedback. If we repeat the previous example of discretizing a temperature, the boundary between "cold" and "comfy" can change via observations of users' heating preferences. It is also possible to directly apply more advanced and specialized algorithms, such as speech processing algorithms to transform speech signals into text, if we know the nature of the data the AmI system will have.

Signal processing functions, such as the Fourier transform, can be applied to highlight important information in the signal. However, this usually works with data with fixed sampling frequency, except that in our case, the data do not necessarily meet this condition.

To summarize, if the AmI system has no idea of the data to be processed, discretizing the categorical values according to their basic variations seems to be a good solution, because it can remain understandable for the user, while not requiring any expert knowledge to be given *a priori*. If the nature of the data can be known in advance, for example temperature, value ranges discretization is possible, which is easier to understand. Pre-processing functions can also be applied to simplify discretization, and fine-tuning can be done to improve the discretization over time.

2.3.2.4 Metadata and data overload

It is possible to have additional data related to those returned by the objects, called metadata. For example, the location of objects, whether GPS data or the room in which the object is located. This data can either be returned by the object or defined by the user when installing the object at home.

The problem of data overload, which has already been encountered in section 2.3.2.2, then arises. How can we reduce the size of the data to be processed to discover habits, in other words how to estimate the range of action of a habit? This is similar to the problem of the frame problem, expressed in section 1.3.2.2: how to define a search boundary field in data or objects? In other words, can we group objects according to their location or their share of common habits, and thus, make the search for habits faster? It is easy to imagine, for example, that an object connected in an attic will contribute little or nothing to habits in the kitchen, for example.

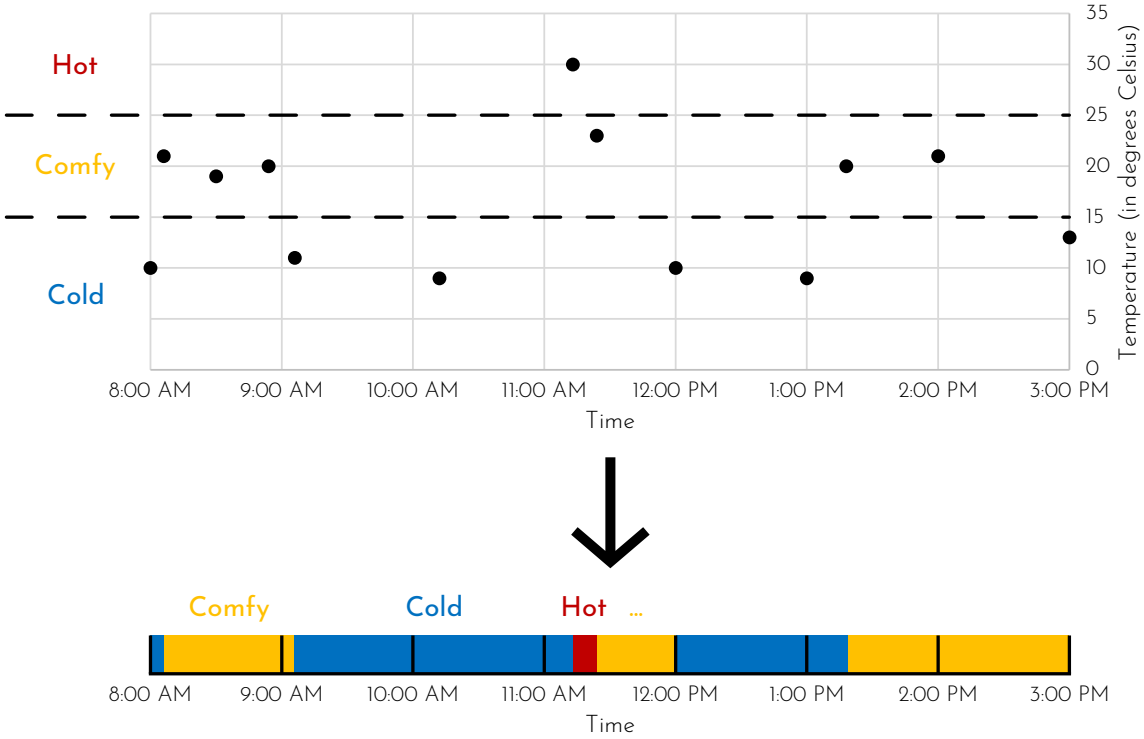


Figure 2.7: Discretization of quantitative data with value ranges

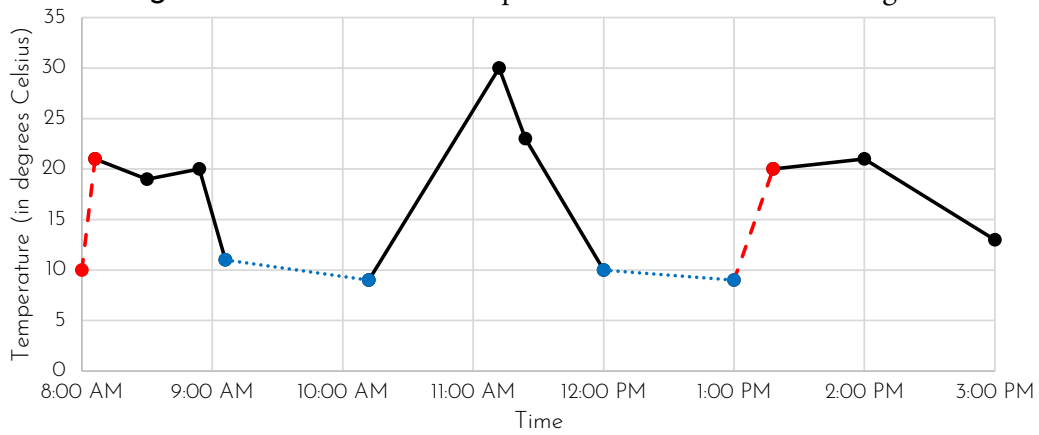


Figure 2.8: Discretization of quantitative data with variations. Dashed variations are considered similar, as are dotted ones.

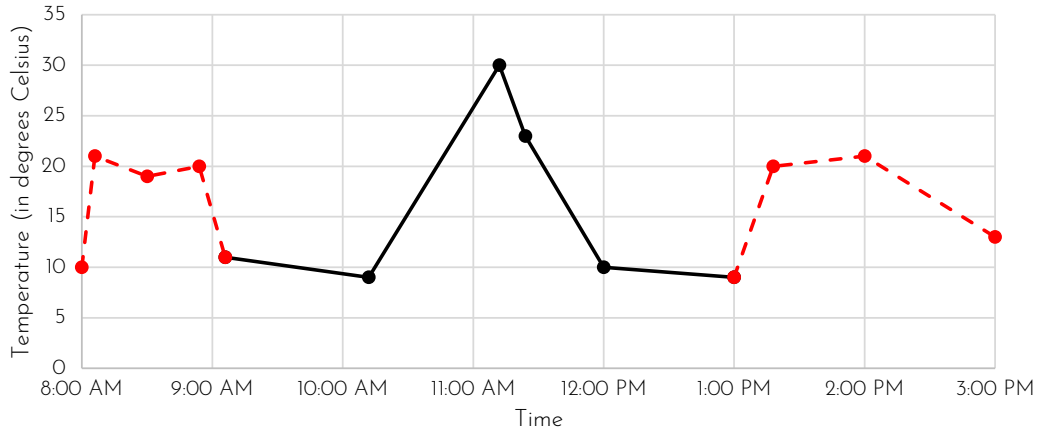


Figure 2.9: Discretization of quantitative data with patterns. Dashed variations are considered similar.

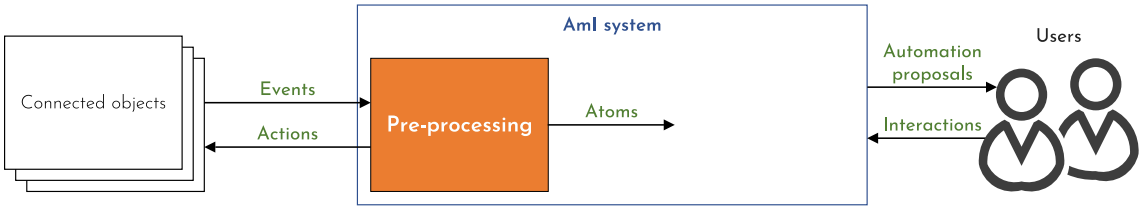


Figure 2.10: Abstract view of the system. The AmI system will need pre-processing algorithms to clean and take into account the different kinds of events sent by the connected objects

[Cumin et al., 2017b] dealt with this problem by grouping the connected objects according to their location. The AmI system processed the data for each room separately, to reduce computation time and make it easier to find habits. Unfortunately, this technique requires *a priori* data, which can be given by the user or by the objects themselves, and does not allow to find habits that take place in several places at the same time, such as the passageway from the entrance to the living room, etc.

2.3.3 Summary

We see, in the quantitative and categorical data, that **pre-processing** will be present. For quantitative data, a discretization will be necessary to be able to treat them equally with the categorical data. For categorical data, a cleaning of duplicates is necessary, and other processing would be useful to take into account context data for example. Thus, the raw data from the sensors will be different from those available after pre-processing. We distinguish these new processed data from the original events by giving them a new name: **atoms**. An atom is a categorical data describing the properties of a variation. In the same way that a variation can be observed several times, an atom can also be instantiated several times. These atoms form the basic material in search of habits, that is why the name "atom" was chosen. The set of all atoms is noted A.

Thus, at the end of pre-processing, we can assume that, to simplify, we have a time series of atoms, whose atoms come from all connected objects in the environment. This time series can be formalized as $TS{=}\langle(t_1, I_1), ..., (t_n, I_n)\rangle$, $I_1, ..., I_n \subseteq A$, where:

- t_i is a **time stamp**. It defines the time coordinates of the occurrence.
- $I_i \subseteq A$ is called an **itemset**. It is the set of individual atoms of A which are observed at time stamp t_i.

These time series of atoms will be the data analyzed by the habit search algorithm. Thus, we can evolve figure 2.3 into figure 2.10, by specifying a little more the architecture of the AmI system.

Also, two major issues are apparent in data processing: how to avoid having too much data, and how to discretize quantitative data in a way that is understandable to the user? We have identified several ways of processing quantitative and categorical data, and these two issues will guide the choices that will be expressed in the next chapter.

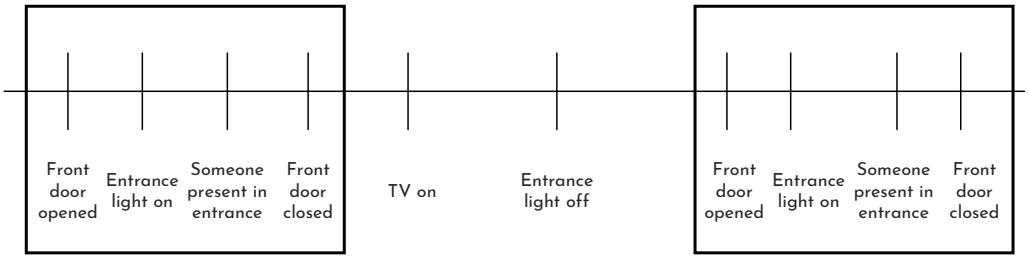


Figure 2.11: Example of a frequent sequence in a time series

2.4 Automation

In section 2.2, we identified that the AmI system would make automation proposals that are understandable to users, because they are the ones who will validate them. It is important to note that not all habits will necessarily lead to automation proposals. For example, "every day at 11pm, the user is present in his room" cannot lead to automation because it does not involve any action to be taken on the connected objects.

Thus, the main goals regarding automation are:

- Recognize the habits of users involving actions made on connected objects, which are potential candidates for automation
- Provide automation in a way that is understandable to users, i.e. make a transcription of the habits found by the system into an intelligible form

The purpose of this part is therefore to identify the structure of the automations to be proposed, and to find an algorithm to search for the habits that can lead to automations.

In what form should these habits be stored, so that they are understandable to users? Of all the possible data structures to represent habits, three are the most commonly used: patterns, sequences and prediction rules.

2.4.1 Possible structures

2.4.1.1 Sequences and Patterns

A **sequence** is an ordered series of atoms. It is noted $\langle a, b, c \rangle$, which means that a comes first, then b, and finally c. A sequence can have one, two or more atoms, up to infinity. If we were to apply this structure to our case, the AmI system would have to search for **frequent sequences**, i.e. sequences of atoms that have been seen many times in the input time series. It is possible to add a notion of duration to these sequences, in order to have an estimate of the time a habit takes.

Here is an example (figure 2.11): ⟨the front door opens, the entrance light turns on, someone is present in the entrance, the front door closes⟩. With this example, it is easy to see that a frequent sequence can characterize a common situation, in this case, the entry of a user into his or her home. Many frequent sequence search algorithms exist, such as PrefixSpan, [Pei et al., 2001], SPADE [Zaki, 2001], and IncSpan [Cheng et al., 2004; Nguyen et al., 2005]. An overview of these algorithms can be found here: [Fournier-Viger and Lin, 2017].

A **pattern**, on the other hand, is a set of data that is almost identical to a frequent sequence, except that it is not ordered. It is noted $s = \{a, b, c\}$. In our case, patterns can also be useful in characterizing a situation. Let us take the example of someone who cooks. Knowing in

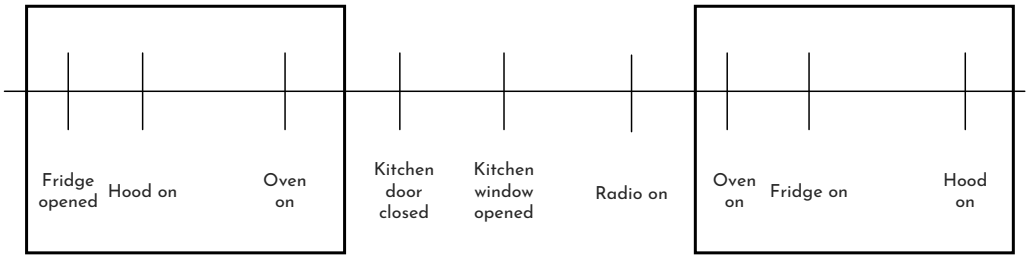


Figure 2.12: Example of a frequent pattern in a time series

which order he opened the fridge, turned on the oven, or the hood doesn't really matter to characterize the cooking situation, as seen in figure 2.12. This situation can thus be described as {fridge opened, hood on, oven on}. Many pattern mining algorithms exist [Jin and Agrawal, 2007; Bouakkaz et al., 2017; Zarrouk and Gouider, 2012; Lee and Yun, 2017; Spiegel et al., 2011; Galbrun et al., 2018; Tanbeer et al., 2009]. An overview of pattern mining can be found here: [Aggarwal et al., 2014; Chee et al., 2019]. Overall, patterns describe situations in a more general way. To describe situations for which the order of events does not matter, it would therefore be interesting to use this structure.

However, we must not lose sight of the objective of our system: to provide automation to users. Patterns and sequences therefore describe common situations, but there is a fundamental notion missing: that of prediction. Finding a habit leading to automation means finding a **rule** that successfully predicts certain user actions based on a set of observations. They may therefore represent a step towards creating rules, but they do not represent the final form of automation proposals. To describe the automations to be discovered, it is necessary to look at another data structure: **prediction rules**.

2.4.1.2 Prediction rules

A **prediction rule** is composed of a condition part and a prediction part. It describes that if the condition is observed, the prediction part will be observed after a certain time. Thus, this structure is well suited to our case, in the sense that we are trying to predict actions to be taken from a situation. If we imagine automation proposals, we can distinguish two main types:

- "If a person is cooking, then turn off the light in the living room." This is a **situation rule**. If a situation occurs, then do a series of actions.

- "Every day at 6pm, turn on the radio." This is a **periodic rule**. They represent a habit that takes place at a fixed time indicator, which can be an hour, a day of the week, etc.

Of course, rules with time indicators and situations may exist. For example: If a person is in his car and it is 6pm, then put the news channel on the radio. Thus, several types of rules can be found, which would require one or several search algorithms.

Referring to the problem of intelligibility, expressed in section 2.2, the description of these rules could be of a high level, as in the previous examples, so that the rules are easily understandable. However, since the system must manage low-level data from connected objects, it should be possible to obtain a semantic layer that transcribes the data to a high-level representation. Since the system is agnostic, and the high-level layer depends on the users, interactions will be necessary to build the necessary semantic layer.

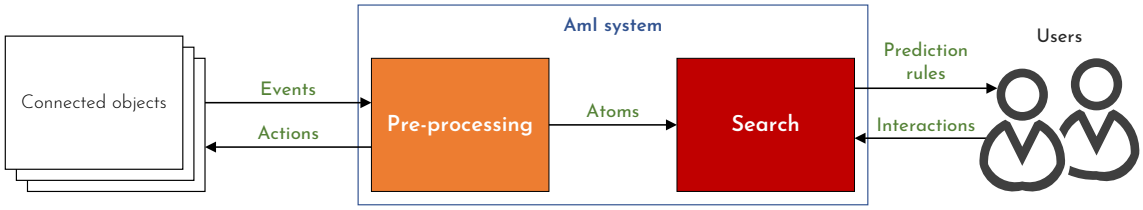


Figure 2.13: Abstract view of the system. This diagram presents the functional architecture of the AmI system, allowing to search for habits.

2.4.2 Summary

After pre-processing, the AmI system will have to look for habits. Prediction rules and periodic rules are complementary structures to represent these habits. Algorithms searching for these rules from the time series of atoms will have to be determined in order to accomplish this task. Also, it will be necessary to determine whether the order of events is important to characterize a situation.

2.5 Continuous improvement of the system

So far, we have been able to detail the primary features of the system, shown in figure 2.13. Now, let us recall two essential objectives of the system: to be useful and personalized. In order to best achieve these objectives, it would be interesting for the AmI system to take into account the feedback from its users to improve its internal functioning.

That is why we have envisaged multiple improvement processes coming from two sources, which can be seen on figure 2.14:

- On the one hand, feedback coming from the user, which can be done in several ways. Its most basic feedback is to accept that certain habits are automations. This provides a small evaluation of the system, to measure its usefulness. We can also imagine that more advanced interactions, allowing a more complete feedback, are possible. For example, users could guide the search, such as asking to search for habits only in certain rooms for example, or not taking into account certain objects. This evaluation, which we call external feedback, aims to improve the building blocks of the system in order to make it more efficient, and more useful.

- On the other hand, we can also imagine internal evaluations, specific to the building blocks, to optimize them. This is internal feedback, which does not come from the users. It is therefore complementary feedback to the first one.

This feedback would allow us to have a system that is constantly evolving, that responds to user requirements, so that it is useful to them. However, designing these different feedback processes of the desired AmI system is a complex issue. We will see in chapter 3 that the building blocks were created with these feedbacks in mind. However, this thesis focused on the search for habits, and therefore the feedbacks have not been implemented.

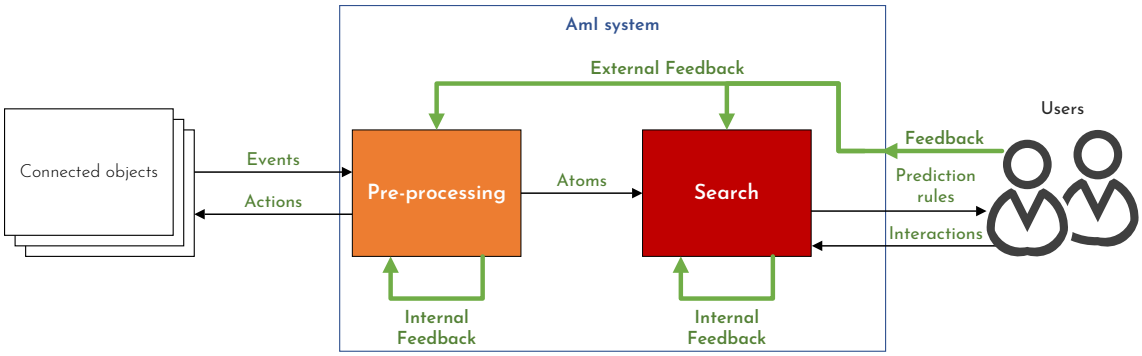


Figure 2.14: Abstract view of the system. This diagram outlines the architecture of the AmI system, including the feedbacks.

2.6 Conclusion

In this chapter, by identifying the main issues and problems, we were able to specify the functionalities of the desired system, illustrated in figure 2.14.

Here is a summary:

- To avoid possible dangerous situations, the users will need to keep control of the system, which will propose automations.
- To address confidentiality issues, it will act locally, via a machine integrated into the environment, and will only process data from a single environment.
- The connected objects are separated into three categories: sensors, interfaces, and actuators. The objects that send more complex data, like video cameras are not considered here.
- The AmI system will pre-process the items sent by the connected objects. This pre-treatment will be different depending on the nature of the events, i.e. quantitative or categorical. The pre-processed data is called **atoms**.
- The instantiations of these atoms will be represented in a time series, which will be the entry of an algorithm for searching prediction rules. These rules will form the automation proposals to users.
- User feedback will be taken into account, and internal evaluations will be made, to improve the system and make it more useful.

These precisions also bring their share of problems, which are summarized here:

- How to take into account changes in user habits?
- How to take into account changes in connected objects, i.e. additions, removals, failures?
- How to take into account the desires of users on such a system, to avoid their lack of interest?
- How to translate prediction rules in a way that makes it understandable to users?

- What types of prediction rules, i.e. situation or periodic, should the system find? Does the system need to find both types of rules, or is one of these types less relevant?
- How to find prediction rules on a time series?
- What interactions should the AmI system have with its users?
- How to design the different feedback processes?

The following chapters therefore explain how the desired AmI system works. We will first detail the architecture in chapter 3 and the algorithms in chapters 4 and 5. They allow to answer some of these problems, and to propose a valid implementation of the AmI system within the constraints set here.

Chapter 3

Architecture

3.1 Introduction

With this chapter, we start to attack the substance of the thesis subject. Here is detailed the architecture of the desired AmI system.

First, the AmI system proposed in the thesis can be seen as a complement to another research system developed within Orange, called "IoT Mashups" [Orange, 2016]. This system proposes to manually create automations in a connected environment via a simplified web interface, as shown in figure 3.1. An example of a mashup could be "If the front door opens and I am present at the entrance, then switch on the entrance light and turn on the radio". Here, all actions on objects are named *a priori*, as well as the data that it is possible to have from the sensors, to make it easier for users to understand. IoT Mashups is intended as an evolution of tools such as IFTTT, which was mentioned in section 1.2.4, because it can take into account several conditions and make several actions within the same automation. Even if it has been developed from scratch, the AmI system proposed in the thesis takes the principles of IoT Mashups, bringing automation proposals to facilitate the user in the use of this tool. This inspiration guided the creation of the architecture.

To explain this architecture, we follow the same path as the data, coming from the connected objects, and going towards intelligible automation proposals, then towards effective automations. Then, we answer why this architecture can respond to the problems and issues

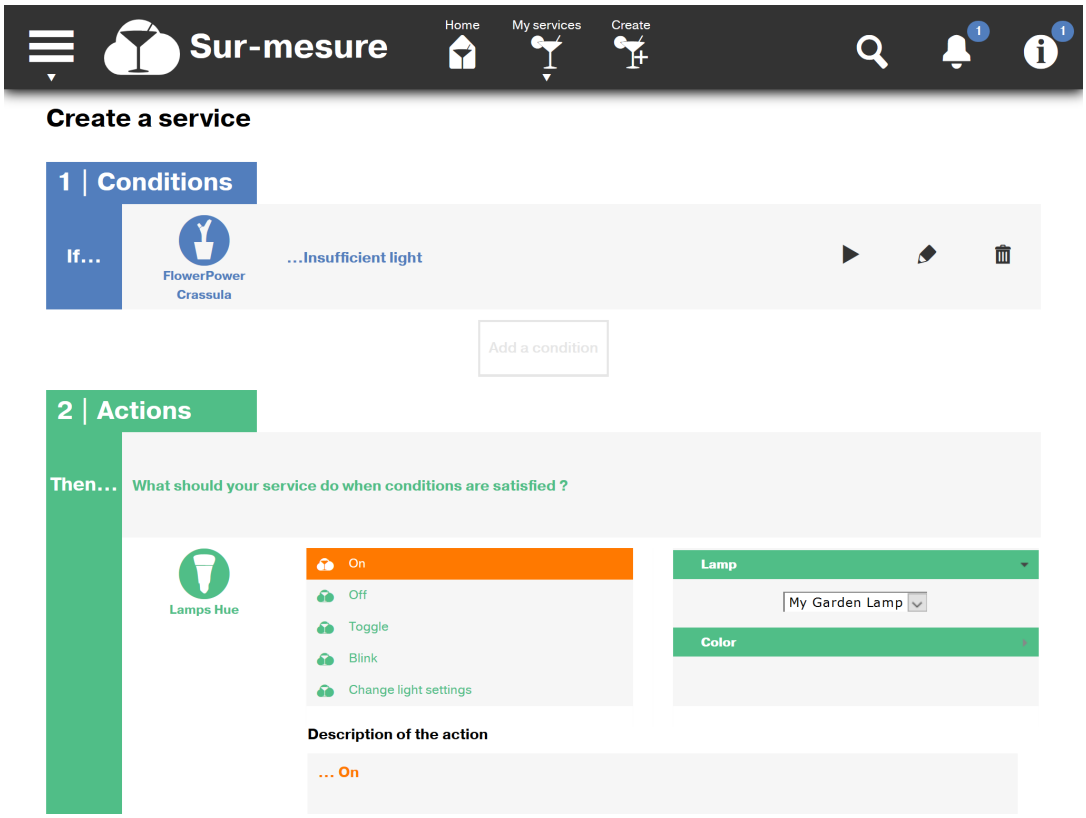


Figure 3.1: Creation of an automation within the IoT Mashup interface

mentioned in the previous chapter, like the adaptation to changes, taking the users into account, and the optimization of computation time. Also, we focused on some building blocks of this system during this thesis. Thus, we detail which building blocks we have implemented, and what remains to be designed.

3.2 Architecture

In this section, we detail the system architecture, taking as a starting point the events sent by the objects, and then going on to automation proposals, then active automations, via a user feedback that evaluates the AmI system in order to improve it. Figure 3.2 outlines this architecture, and represents a visual support complementary to this section. As seen in chapter 1, the proposed system is at the crossroads of several paths: AI with activity discovery, UbiComp with connected objects, and ergonomics with the user perspective.

This architecture differs from those present in the state of the art. Indeed, most of the architectures presented in the field of ambient intelligence do not focus on the discovery of activities, but rather on optimization problems, such as in Ambient Assisted Living or the reduction of energy consumption or the recognition of predefined activities. Other systems, such as [Meurer et al., 2018] or [Jakkula and Cook, 2007], use neural networks, which makes it impossible to understand why the system has made a decision, and therefore to know how to control its actions. [Mazac et al., 2014] may be subject to the same criticism, as it uses a multi-agent system whose operation can be difficult to understand. The AmI architecture presented here discovers new activities in a connected environment and proposes a concrete application case, that of making automation proposals based on observed habits. Moreover, it is designed

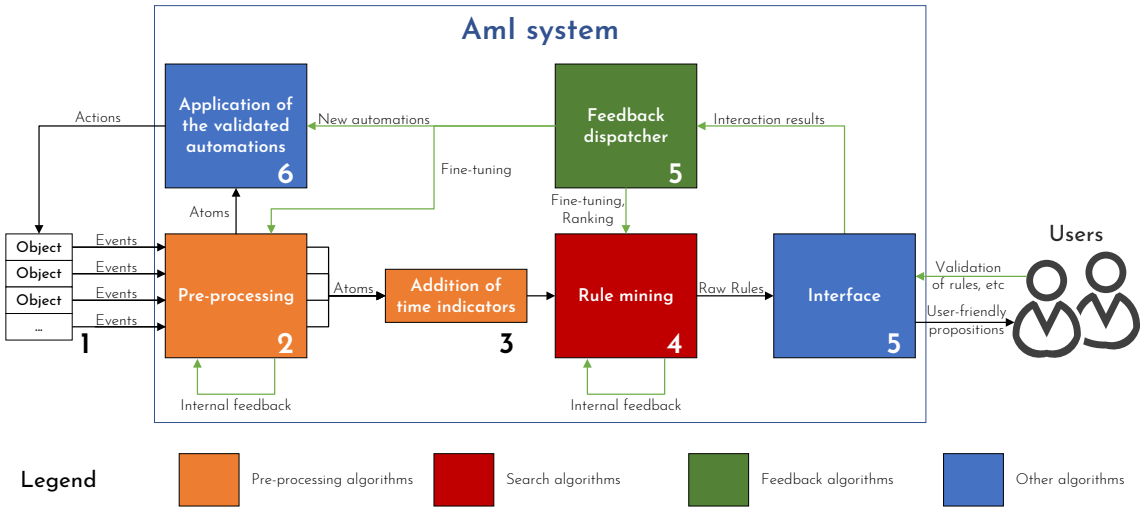


Figure 3.2: Architecture of the proposed AmI system

to promote user interactions, with interpretable results, and the taking into account of user feedback.

3.2.1 From events...

As a reminder, we have connected objects[1] sending quantitative or categorical events (identified by block 1 in figure 3.2). An object is formalized as follows: $o = \{category, datatype\}$, where:

- $category$ represents the category of the object, i.e. a sensor, interface, or actuator.
- $datatype$ is the type of data sent, either quantitative or categorical.
- The set of all objects in the environment is named O.

Here, the function of the object, e.g. a temperature sensor or switch, is not mentioned. This is because the system is agnostic, and it knows nothing about the object other than its unique identifier, its category, i.e. sensor, interface or actuator, and the type of data sent by the object.

Events have a different structure depending on the communication protocol used, such as Bluetooth, Zigbee, or LoRA, which are not compatible with each other (section 1.2.4). In our case, we determine that a sent event is composed of the identifier of the object sending it and the value of the event.

We will therefore formalize an event as: $e = \{o, value\}$, where:

- o is the object that sends it.
- $value$ is the value of the event.
- An example of an event would be $e = \{temp62, 21\}$.
- The set of all events coming from an object o is named E_o.

[1] To simplify the following formalizations, we will define an object as elementary, not composite, i.e. it is composed of only one sensor, interface or actuator (section 2.3.1). Thus, according to this definition, a composite object is simply the combination of elementary objects

It is possible to obtain further information about the event from the identifier of the object sending it, namely the category of the object, i.e. sensor, actuator or interface, and the type of data, i.e. quantitative or categorical.

Thus, we consider that each object sends a time series of events either quantitative or categorical. To formalize, an object o sends over time a time series $TS_o = \langle (t_1^o, e_1^o), ..., (t_n^o, e_n^o) \rangle$, $e_1^o, ..., e_n^o \in E_o$, which we will simplify into $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle$, $e_1, ..., e_n \in E_o$ for greater generality, where:

- t_i is a time stamp, i.e. the time coordinates of the occurrence.
- $e_i \in E_o$ is an event which is observed at time stamp t_i. In this thesis, we consider that a connected object cannot send two different events at the same time. Thus, each timestamp is different from the others.
- E_o is the total set of events that the object o can send. It is either a set of quantitative or categorical events, not both.
- TS is ordered by the time stamps.

Thus, for n connected objects, the pre-processing part of the AmI system (block 2 in figure 3.2) must process n time series. In our architecture, each time series is treated **independently**. This allows adaptation to the input data. Indeed, temperature data do not have the same range of values, nor the same variations, as electrical data for example. Processing these time series independently therefore makes it possible to adapt to the specificities of each time series. Pre-processing acts as follows:

- It discretizes quantitative events by identifying their basic variations (section 2.3.2.3). This represents a good balance between intelligibility and ease of implementation without any data preconceptions, as stated in that section.
- It cleans categorical events by removing duplicates (section 2.3.2.2).

Pre-processing is an essential component of the system. As seen in figure 3.2, it provides a time series of atoms that is used for two other parts of the system:

- The rule mining algorithm, to discover habits from the data history of connected objects
- The application of the rules validated by the user, which must react quickly in case of the identification of a situation leading to an automation

In order for the automation to be effective, the data must be taken into account and pre-processed as they arrive. The pre-processing must therefore work online, i.e. it should return results as the data comes in, not process everything at once or periodically. Also, pre-processing algorithms must be defined by **few parameters**. Indeed, as the feedback part (5 in figure 3.2) will have to optimize these parameters, having as few as possible will facilitate this optimization. Those parameters are first predetermined, or chosen according to the input events, and can be modified at any time. In addition, if an object is deleted or added, the system only needs to delete or add a process.

These time series of events, being modified by pre-processing, become time series of atoms. The distinction between events and atoms is intended to emphasize the modification of these data by pre-processing.

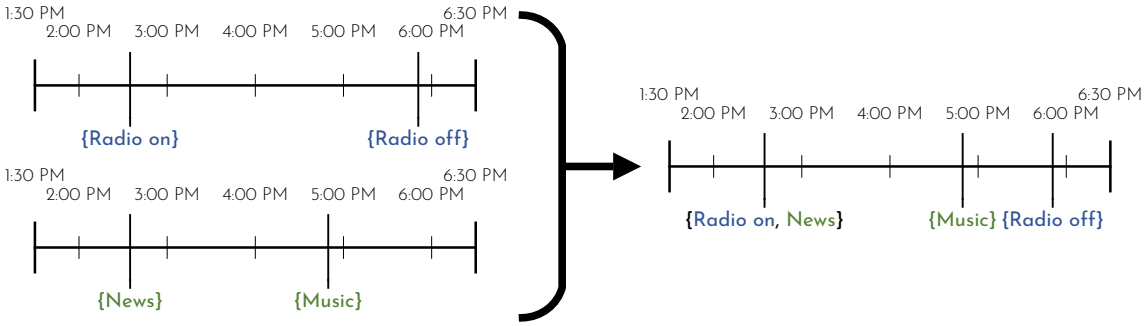


Figure 3.3: Example of merging two time series based on the time of arrival of events

3.2.2 ... to atoms...

As a reminder, an atom comes from one or more events from an object that have been modified by pre-processing algorithms. An atom is formalized as $a = \{o, desc\}$, where:

- o is the object from which the atom has been made:
  - It is used, among other things, to know in which category of object the atom comes from, i.e. sensor, interface, actuator. This will be useful when searching for rules.
- $desc$ is a set of descriptors of the values represented by the atom:
  - If the atom comes from categorical events, $desc$ will be the end value of a variation (section 2.3.2.2).
  - If the atom comes from quantitative events, $desc$ will be a set of indicators describing the variation.
- An atom has a unique identifier, in the form of a number. This is used for the rule search algorithm presented in chapter 5.
- An example of an atom would be $a = \{Radio, ON\}$.
- The set of all atoms is named A.

As a reminder, at the end of the pre-processing, there are several time series of atoms, one for each object. All the time series of atoms are then merged into a single one, as illustrated in the figure 3.3, which will be the input for the prediction rule search algorithm.
$TS = \langle (t_1, I_1), ..., (t_n, I_n) \rangle$, $I_1, ..., I_n \subseteq A$, where:

- t_i is a time stamp, i.e. the time coordinates of the occurrence.
- $I_i \subseteq A$ is an itemset, i.e. the set of individual atoms of A which are observed at time stamp t_i, and are sent by one or several objects.
- A is the set of all atoms.
- TS is ordered by the time stamps.

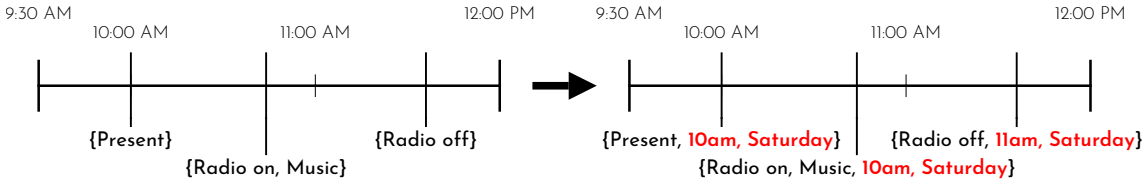


Figure 3.4: Example of adding time indicators to a time series. Here, hour and weekday indicators are added.

Then, an algorithm adds time indicators to each itemset of the time series (block 3 in figure 3.2). This process is illustrated in figure 3.4. Coming from timestamps, they will make it easier to find periodic rules. Examples of time indicators can be the hour of the day, day of the week, month, season of the year, etc. They therefore represent context data, and exist as atoms. To unify this new kinds of atoms with the other ones, the system assigns a fictional sensor to it, and represents the indicator variable, i.e. time, day of the week, etc.

Thus, this algorithm adds these time indicators for each itemset of the time series in the form of atoms, then the prediction search can be done afterwards.

3.2.3 ... to prediction rules...

A prediction rule is noted $r : A_c \Rightarrow A_p$, where A_c is the condition, and A_p is the prediction of the rule. R describes that if A_c is observed, A_p will be observed after a certain time. This global structure corresponds to what we want to achieve in our AmI system. Indeed, in our case, we are trying to propose automation in this form: If a set of conditions is recognized, then do this set of actions. Thus, the prediction part of the rules to be found must be restricted to observations of data from actuators only.

We could also look for rules of type $A_p \Leftarrow A_c$, where it is possible to make predictions from more recent observations. An example would be: if the user turned on the bathroom light, then he entered the bathroom first. However, these rules cannot lead to automation, so we prefer to focus on the first type of rules.

The rule search algorithm (block 4 in figure 3.2) searches for two types of rules: situation rules and periodical rules, using the time indicators mentioned above. This search does not need to process the data online. This is different from pre-processing, where the results are used to initiate automation. This part only serves to find new habits, which can take time, and it can therefore run periodically, taking into account part of the data via a buffer system.

Several prediction rule structures are possible, which cover both situation and periodic rules. Here are the two most common ones [Fournier-Viger et al., 2015]:

- **Fully-ordered sequential rules**, where the condition A_c and the prediction A_p are sequences of atoms
- **Partially-ordered sequential rules** [Fournier-Viger et al., 2015], where A_c and A_p are both unordered. This remains a prediction rule, because A_p comes after A_c. Two mathematical structures are possible for A_c and A_p:
  - **Sets** of atoms, where an atom can only appear once
  - **Multisets** of atoms, where multiple instances of atoms are allowed. The number of instances of an atom in the multiset is called the **multiplicity**. For example, the multiplicity of the atom x in the multiset $\{x, x, y\}$ is 2.

Partially ordered prediction rules seem more appropriate to our case, because they are more general, in the same way as the patterns in section 2.4.1.1. Describing a situation does not necessarily require an order, but the multiplicity of an atom can be important. To explain this choice, we can take the example of a sound detection lamp: when one claps twice, i.e. when one makes the same sound twice, the lamp lights up. Of course, other rule structures may be relevant, such as fully ordered rules, to discover habits where order is important, and the system proposed in this thesis may be adapted to take them into account, via a modification of the rule search algorithm, or a completely new algorithm.

Temporal data are affixed to these rules, improving their understanding: first, the time delay between condition A_c and prediction A_p, the execution time of A_c and that of A_p. This allows to visualize the time that a habit takes. In addition, over time, user habits will evolve, and the rule search algorithm will be updated to adapt to these changes.

Thus, the search algorithm looks for partially ordered rules containing multisets, from a time series of atoms. Now that we have the structure of the rules to search for, let us define the data that will be in these rules. As mentioned above, atoms come from three types of objects: sensors, actuators, and interfaces. **The atoms present in condition A_c come only from sensors and actuators, and those of A_p only from actuators.**

This allows to search only for habits that can lead to automation, i.e. actions to be performed on connected objects. It should also be noted that interfaces are not taken into account at all. Indeed, interfaces, by definition, are in a way duplicates of the actuators on which they act. To speed up the calculation time, and avoid finding trivial rules, such as the action an interface has on its actuator, or duplicate rules, they are therefore not taken into account.

3.2.4 ... to user-friendly automation propositions...

The way in which prediction rules will be proposed to users is a key component of the user interface (block 5 in figure 3.2). The intelligibility of the proposed rules is therefore essential for users to interact with the system. In this thesis, we focused on rule search algorithms, not on this interface. However, a first step to improve the presentation of the rules has been made. In our implementation, the rules are presented in the form of a sentence, broken down as follows:

- “If ”
- Condition of the rule, presented as follows. For each atom present in the condition:
  - Name of the connected object
  - Value of the atom, using the present (example: the door opens, the light turns off)
  - “ and ”, if other atoms remain in the condition
- “, then ”
- Prediction of the rule
  - Same as for the condition, except that the imperative is used to indicate actions (example: open the door, turn on the light)

A priori data were used to be able to make this syntax, especially to simplify object names, and use the imperative for prediction actions. In addition, for the purposes of the experiments,

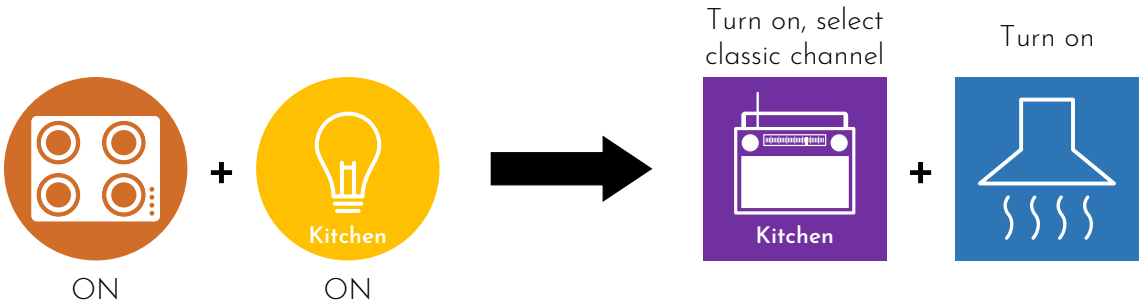


Figure 3.5: Mockup of a graphical presentation of a rule

a French translation has been made, as this is the mother tongue of the people participating in the experiment. It should be noted that this representation remains possible to implement if the user, or the object manufacturers, provide detailed information on the connected objects, as well as on the possible actions to be taken on them. Atoms from quantitative events are a particularly difficult issue when it comes to expressing rules in an intelligible way. Indeed, we have seen in section 2.3.2.3 that several discretizations are possible, which it would be interesting to test and compare for such a system, as each one brings its own set of advantages and disadvantages in terms of intelligibility and ease of implementation.

Other representations are possible, such as a visual one, representing objects as icons (figure 3.5). This allows a more expressive and pictorial view of the rule, in relation to a sentence.

There are many possibilities for simplification, especially through the metadata that can be retrieved from objects. The location of objects, for example, not only makes it easier to visualize where a rule is located, but also to simplify it. For example: "In the living room, if the light comes on and someone is present, then turn on the television" can be a good simplification. No need to add on each object name it is in the living room, this is summarized at the beginning of the rule.

Finally, to simplify a rule, it would be preferable to have a higher level representation, i.e. more abstract and more understandable for users, and less close to the atoms. An example would be "When I cook, then turn on the hood". This requires having a base of habitual situations, gathering a set of atoms to a semantics. In the previous example, this would be a "Cooking" situation associated with opening the fridge, turning on the hob, turning on the oven, etc. As we have seen previously, the thesis is at the crossroads of several disciplines, and a simple building block of the proposed architecture, such as the presentation of rules to users, which appears anodyne, can lead to a number of studies.

3.2.5 ... to feedbacks...

As mentioned above, users will have interactions with the AmI system, especially to accept or not the proposed rules. Indeed, it is the users who will evaluate the output rules, i.e. accept or not them as automations.

It is possible to imagine many interactions with users. This has not been the focus of this thesis, but we can already indicate some ways forward:

- Give full control to users, regarding:
  - Adding or deleting objects
  - Setting operating time slots, or common situations that the system should not observe to find prediction rules. Indeed, the system is at the service of users. They must be able to define observation limits, in order to be comfortable with it

  - Adding manual rules, and deleting all possible rules
  - Modifications on all the building blocks of the system, for example reset the pre-processing, or change the algorithms of this part, no longer take into account time indicators, etc.
  - Manual actions on actuators or interfaces in the environment
  - Monitoring on sensors of the environment
  - Shut down the system at any time
  - Reset the system at any time

- Have a simple presentation of the automation propositions, and to give users the possibility to interact in order to have a more precise, but more complex representation of them.

- Have several means of interaction, on several devices, in order to avoid having a central control point, and rather to materialize the fact that the system is really ambient.

To summarize, these interactions form a feedback (block 6 in figure 3.2), and provide a kind of evaluation of the system. Thus, the architecture presented here also aims to capitalize on this user feedback to improve the different algorithms used by the system. A building block of the AmI system, called "Feedback dispatcher", is therefore intended to evaluate the different building blocks involved in the automation proposals:

- Discretization, by fine-tuning the parameters or by putting several algorithms in competition

- Rule mining, by building a proximity map between objects, and a usefulness function, to make search faster. This will be discussed in chapter 6.

- In addition to providing an evaluation, it will also be used to send the new accepted rules to the algorithm applying the automations (block 6 in figure 3.2), for the effective deployment of these automations.

In addition to this feedback from users, self-evaluation processes are present for rule pre-processing and rule mining. Also, it is very likely that a lot of results can be found by the rule search algorithm. Of course, users will not be able to evaluate all of them, and these rules should be given a usefulness rating based on past interactions with these users. Thus, this notion of usefulness can make it possible to optimize the display of results according to users' tastes, but also to optimize the search for rules, to make it faster. In addition, this algorithm could make a proximity map between connected objects, also to optimize the search. It would be based on the rules found previously.

3.2.6 ... to active automation

For the system to be complete, it is necessary not only to offer automations, but also to apply those that are accepted. The building block for applying the rules (block 7 in figure 3.2) takes as entries:

- Rules accepted by users

- The time series of atoms sent by the pre-processing algorithms

As soon as the condition part of one of the accepted rules is seen on the atomic time series, the algorithm will apply the rule, i.e. make actions on the objects corresponding to the prediction part of the rule by respecting the time between condition and prediction.

3.3 Answers to the problems of the previous chapter

Now that we have defined the global architecture of the AmI system, let us look at how to solve the issues raised in the previous chapter.

3.3.1 Adaptation

First, this architecture is intended to be adaptive to changes in user habits. Indeed, the rule mining algorithm will periodically search for new habits, and updates its results. Then, additions and deletions of connected objects are supported by the architecture. Each object having its own pre-processing process, adding or deleting an object is like adding or deleting a process. If an object fails, and returns inconsistent data, no new prediction rule could take it into account, and this object would be ignored by the AmI system. In addition, rules found with this object would be deleted.

3.3.2 Taking users into account

The entire architecture is focused on users. First, pre-processing unifies categorical and quantitative data, so that these data can be understood. Second, the system returns prediction rules that are intended to be intelligible to users, with an explicitly defined condition and prediction part, and related information such as the time between condition and prediction. This makes it possible to give all the necessary information so that users can validate a rule or not in full awareness. Then, the users' wishes are taken into account. The order in which the rules are displayed, as well as the search for them, is adapted to interactions with users, through the usefulness measure: it will be able to determine whether a rule is considered useful for the user. The more useful it is, the more likely it will appear as an automation proposal.

Privacy is also taken into account. Since this AmI system only processes data from one environment, it can be implemented in a local machine, present in the environment (section 2.2.2.2).

3.3.3 Optimization of computation time

As we have seen in section 2.3.2, the system must balance between having as much information as possible to best characterize the rules to be found, and having too much data to process, making the execution time too long. In the algorithms used in the current implementation, decisions have been made to address this issue. They will be explained in the following chapters, alongside with the algorithms themselves. Finally, the system aims to improve over time, by optimizing rule search and pre-processing, based on user feedback.

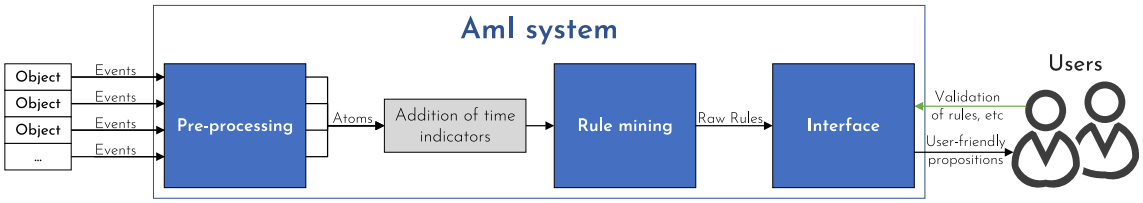


Figure 3.6: Implementation of the AmI system. The system is not adaptive as it stands, but the main components are present

3.4 Current implementation

As part of this thesis, a primary but functional implementation of the AmI system was made (figure 3.6). During this thesis, we focused on the search for prediction rules. Thus, feedback mechanisms have not been implemented yet, as well as the notion of usefulness, making the system non-adaptive to users and the environment. In addition, some parameters in the rule search and in the addition of time indicators are defined by hand. However, the current implementation allows, from raw data of connected objects, to find situational prediction rules and some periodicals. Details of the algorithms used and created for this system are given in the next chapter. Moreover, since the architecture is detailed in this manuscript, it can be taken over and modified, as well as its algorithms, and can thus evolve into a final form corresponding to the original ambition.

3.5 Conclusion

This is one of the contributions of the thesis: an AmI architecture adaptive to changes in objects and habits. This AmI system is scalable because it takes into account user feedback to optimize its algorithms. The architecture is clear, including building blocks whose functions are understandable to users, who may want to know how this system works. Finally, this AmI system respects the privacy of users through local data processing and emphasizes the notion of usefulness of the proposed automations, justifying its existence to users. This architecture is fundamental to this thesis, because it highlights a possible way to build an AmI system that meets all these criteria.

During the thesis, we focused on some of the fundamental building blocks of the system, which are pre-processing, in chapter 4, and rule research, in chapter 5. Then, in chapter 6, we will share our evolutionary perspectives, which may be useful to researchers in the field.

Chapter 4

Pre-processing

4.1 Introduction

This chapter, as well as the next one, presents the algorithms used and created for the main building blocks of the AmI system. This chapter introduces an essential component of the system: pre-processing, which consists in unifying quantitative and categorical data from various objects, considering only variations in their data. There are several issues to be considered in this chapter:

- The pre-processing must be able to adapt to the wide variety of data that can be returned by the connected objects. This includes the type of events, i.e. quantitative or categorical, and, in the case of categorical events, the value range, and the variations usually encountered in the data returned by the object.

- As mentioned in section 3.2.1, we need to have as few parameters as possible. This allows the feedback part to optimize the pre-processing more easily if needed.

- The result of the algorithm, i.e. the atoms, must be at least understandable by a human being, so that the automation proposals returned by the system can be understood by the users.

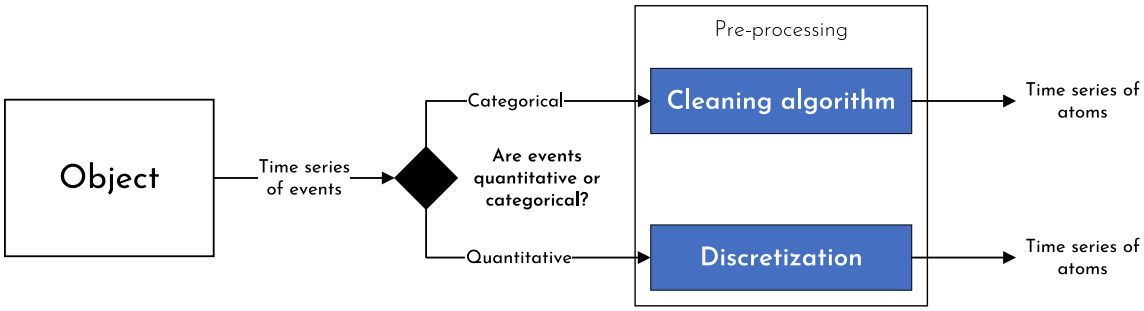


Figure 4.1: Architecture of the pre-processing part of the AmI system

In this chapter, we introduce a pre-processing architecture designed with the above mentioned objectives in mind. We detail the algorithms used, and show some results to illustrate its operation.

Here, each connected object is treated independently of each other, allowing adaptation to the characteristics of the objects. Also, the pre-processing algorithms will be different depending on the nature of the data, i.e. whether they are quantitative or categorical. Indeed, as shown in section 3.2.1, quantitative events are discretized to unify the data. This is what is applied in this architecture (Figure 4.1):

- For categorical events, a simple algorithm for cleaning redundant data is used, to keep only events which represent a category change in the time series.
- For quantitative events, a discretization process is implemented, adapted to each time series. This allows each time series to be treated in a unique way.

At the end, the modified data resulting from these algorithms are called atoms, and are the basic elements of the rule search presented in chapter 5.

4.2 Categorical events

Suppose we have an object o, which sends categorical events. We therefore have a time series of the form $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle, e_1, ..., e_n \in E_o$. The algorithm only keeps the value changes:

- Input:
  - $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle, e_1, ..., e_n \in E_o$: Time series of quantitative events
- Output:
  - $TS_a = \langle (t_1, a_1), ..., (t_n, a_{n_a}) \rangle, a_1, ..., a_{n_a} \in A_o$: Time series of atoms

The goal here is simply to remove duplicates in the data, and to keep only the value changes, so that a time series like that of figure 4.2 becomes like that of figure 4.3.

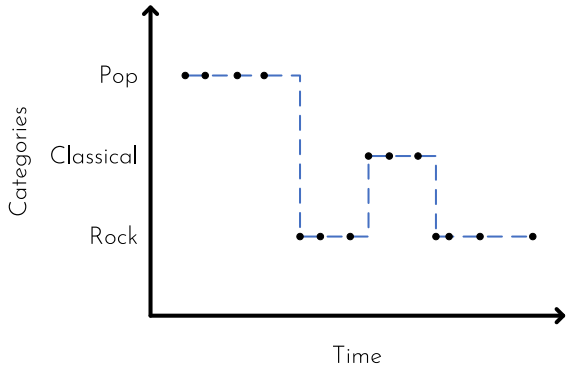


Figure 4.2: Example of a time series of categorical events. It is possible to draw a curve (dashed on the figure) on the basis that as long as there is no new event, the selected category does not change

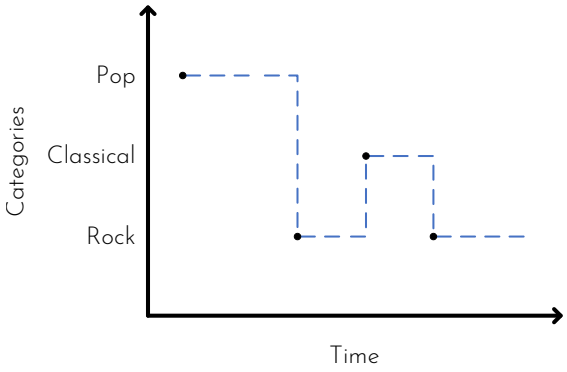


Figure 4.3: Same time series as in figure 4.2, but with redundancies removed. Note that the variations, and therefore the curve, of the time series have not changed

Algorithm 1: Pre-processing algorithm for categorical events

Data: $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle$, $e_1, ..., e_n \in E_o$: time series
`// Initialization`
1 $TS_a \leftarrow \langle\rangle$;
`// Main loop`
2 **foreach** $(t_i, e_i) \in TS, i > 0$ **do**
  `// If the event is different from the previous one`
3 **if** $e_i \neq e_{i-1}$ **then**
    `// Create an Atom from this event and add it in the new and cleaned time series`
4    $a \leftarrow$ new Atom($o = e_i.o, value = e_i.value$);
5   Add (t_i, a) at the end of TS_a;
6 **Return** TS_a;

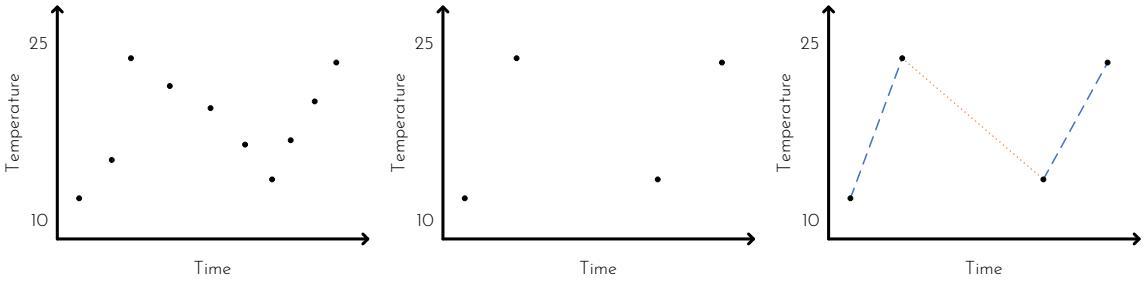
As can be seen, this algorithm works very simply. The time series of atoms contains all changes in the values returned by the connected object.

4.3 Quantitative events

As specified in section 2.3.2.3, we have chosen to discretize the quantitative data. But how to do this? Do we observe the variations, or rather the values, or make a profile of the signal, as explained in section 2.3.2.3? Everything rests on a balance between two constraints that we have set on this part: the lack of expert data on the data and the need to obtain intelligible results. In this system, we have therefore chosen to simplify and classify the variations in the data, which represents a good compromise for these two constraints. We therefore propose, in this section, algorithms that discretize a signal according to its variations.

It should be noted that it is difficult, if not impossible, to evaluate the pre-processing of quantitative data itself. Indeed, this evaluation depends entirely on the use that will be made of the pre-processed data. In our case, the evaluation will therefore depend on the prediction rules found by the entire AmI system.

Due to this lack of evaluation criteria, the discretization of quantitative elements was



(a) Unprocessed time series

(b) Same time series but keeping only the major breaking points, showing the macroscopic variations

(c) Same time series but where similar variations are clustered. Here, the dashed variations are considered identical, and different to the dotted one

Figure 4.4: Example of the pre-processing a time series of quantitative events

designed with the following objectives:

- The identification of macroscopic variations, and the grouping together of similar variations.
- The adaptation to the characteristics of a signal.
- The reduction of the number of parameters used by the algorithms.
- The need to have intelligible results.
- The need to run online, i.e. the algorithm pre-processes the data as it arrives.

To achieve these goals, two successive algorithms are used: Segmentation and Clustering. Let us take a time series example shown in figure 4.4a.

- **Segmentation** first simplifies the time series of events by keeping only the major breaking points (figure 4.4b). Those are the couples (timestamp, event) that are describing the macroscopic variations of the variable, hence leaving microscopic variations. To do it, it applies the "Sliding Window Algorithm". [Lovrić et al., 2014], known in the time series segmentation domain. We use this algorithm because it runs online, identifying major variations as soon as they end. Other algorithms in this domain do not run online, such as the Top-Down algorithm, also known as the Douglas-Peucker algorithm [Douglas and Peucker, 1973], and the Bottom-Up algorithm [Keogh and Smyth, 1997]. Others can, but through a buffer, like the Sliding Window And Bottom–Up algorithm [Keogh et al., 2001].

  Finally, it outputs a simpler time series of events, where the **segments**, i.e. the lines formed by two consecutive points of the time series, form the macroscopic variations of the variable.

- **Clustering** groups similar segments into atoms (figure 4.4c). It uses a clustering technique called Hierarchical Clustering [Johnson, 1967] that can also be run online [Chen et al., 2002; Widyantoro et al., 2002]. This process then creates a time series of atoms.

It is important to note that Segmentation and Clustering both use a threshold: θ_{seg} for Segmentation, θ_{clu} for Clustering, whose purposes will be explained in the next part. To adapt to the input data, these two algorithms observe the signals during a time period noted δ_t. For each time series of quantitative values, two thresholds have to be fixed for the discretization to operate, in addition to the time period δ_t.

Also, even if the algorithms were chosen for their ability to run online, the implementation made in this thesis does not run online. Thus, the experiments will be done offline, to give an idea of the returned results and the execution time of these algorithms according to the volume of processed data.

We illustrate here how these algorithms work by presenting their results.

4.3.1 Segmentation

4.3.1.1 Goal

The main goal of Segmentation is to clean the microscopic variations, while keeping the macroscopic ones, by removing the points that are not used to form these macroscopic variations. The challenge of this algorithm is to identify what a macroscopic variation is, whereas it does not know in advance the range of the values of the events sent by the object.

4.3.1.2 Methodology

First, let us look at the algorithm used: the Sliding Window Algorithm. The algorithm splits the times series in intervals on which the data points can reasonably be fitted by a linear function. It takes into account a time series, and has as parameter a threshold named ϵ. Figure 4.5 is a representation of this, and a complete pseudocode can be found in appendix A, at algorithm 7.

- It starts from the first point of the time series (step 2 in figure 4.5), then creates a segment between the first point and the third. This segment represents an attempt to simplify the time series, as it aims to check whether the second point can be removed.
- If the second point in the time series has a vertical distance from the segment below the threshold (symbolized in dark red in the second picture of the figure 4.5), it also tries to delete the third point, by creating a new segment, this time between the first and fourth points (step 3 in figure 4.5), and so on, by comparing the vertical distances of all the points potentially to be deleted from the segment.
- As soon as one of the points is too far from the segment (i.e. the vertical distance between the point and the segment is greater than the threshold), then the segment that was previously created is validated, and the points it replaces are permanently deleted (steps 3 and 4 in figure 4.5).
- Finally, the end point of the segment becomes the new starting point of the algorithm, which iterates over the previous steps, until it reaches the end of the time series.

To express the algorithmic complexity of segmentation, let us take the example of a segment, composed of two points, replacing k points in the original signal. To achieve this result, the algorithm tries to replace 3 points, then 4, up to $k+1$ points. Therefore, it calculates the vertical distance of 1 point, then 2, up to $k-1$ points, and does $1+2+...+k-1$ calculations, that is, $\frac{(k-1)*k}{2}$. For a signal composed of n points, the worst case is to replace

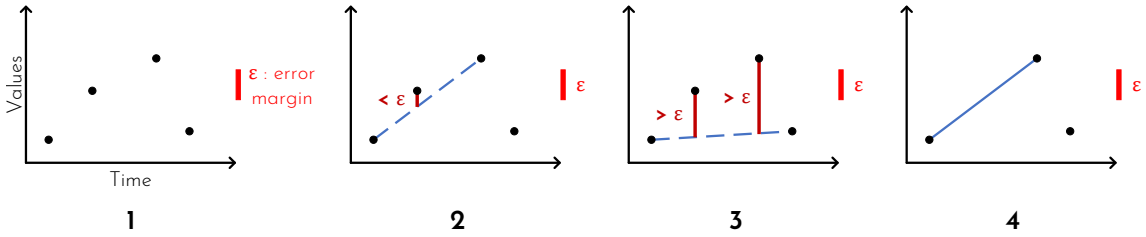


Figure 4.5: Sliding Window Algorithm Example. Here, the second point is removed, but the macroscopic variations remain

the whole signal by a single segment, i.e. make $\frac{(n-1)*n}{2}$ calculations. Thus, the algorithm has a time complexity of $\mathcal{O}(n^2)$.

As we have seen, to distinguish microscopic variations from macroscopic variations, the Sliding Window Algorithm has as parameter an ϵ threshold. It is obvious that ϵ depends on the range of values of the events sent by the object. The segmentation part therefore observes the time series during a δ_t period to estimate this range, which is simply the difference between the maximum and the minimum values observed during δ_t, and stores it as Δ_t. Once Δ_t is estimated, ϵ is calculated as a fraction of this range, more precisely:

$$\epsilon = \theta_{seg} * \Delta_t, 0 < \theta_{seg} < 1 \tag{4.1}$$

θ_{seg} is defined in our implementation as 0.1, and is intended to be editable by the feedback part of the AmI system, described in section 3.2.5. Also, the observation time δ_t can be modified by the feedback, and the feedback can make a new estimation of the range of values Δ_t.

Then, a representative time series will be built, which will be used by the clustering algorithm, to have an estimate of the macroscopic variations characteristics to be expected in the time series. Derived from the δ_t observation period, this time series is noted TS^{ref}. In our current implementation, TS^{ref} is no longer updated once created. Indeed, the main problem of updating it is that it can change the clustering, and therefore the resulting atoms. Mechanisms can be designed to overcome this limitation, such as an alert if the signal differs too much from its reference signal TS^{ref}, in which case TS^{ref} could be updated. If TS^{ref} must be updated, an algorithm is needed to switch to these new atoms without losing the rules found with the old ones.

4.3.1.3 Data

- Input:
  - $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle, e_1, ..., e_n \in E_o$: Time series of quantitative events
  - $\theta_{seg} \in [0, 1]$: Segmentation threshold, in the shape of an error margin (percentage). Here, set to 0.1.
  - δ_t: Data observation period. Here, set to 24 hours.
- Output:
  - $TS^{sim} = \langle (t_1, e_1), ..., (t_{n_{sim}}, e_{n_{sim}}) \rangle, e_1, ..., e_{n_{sim}} \in E_o$: Time series of quantitative events
  - $TS^{ref} = \langle (t_1, e_1), ..., (t_{n_{ref}}, e_{n_{ref}}) \rangle, t_{n_{ref}} \leq (t_1 + \delta_t) < t_{n_{ref}+1}$: Representative time series

Algorithm 2: Segmentation algorithm

Data: Time series of quantitative events $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle$, Threshold θ_{seg}, Observation time frame δ_t

Result: Time series of quantitative events TS^{sim}, time series of quantitative events TS^{ref}

```
// Get all the values observed after t_1 and during δ_t
```

$\Delta_t \leftarrow \{e_i.value, (e_i, t_i) \in TS | t_1 \leq t_i \leq t_1 + \delta_t\}$;

```
// Compute the error margin used in the Sliding Window
   Algorithm
```

$\epsilon = \leftarrow \theta_{seg} * (\max(\Delta_t) - \min(\Delta_t))$;

```
// If the time series has no variation, we choose not to take
   it into account
```

if $\epsilon = 0$ **then**

  return Error, no observed variation;

```
// Create the simplified time series using the Sliding Window
   Algorithm (algorithm 7)
```

$TS^{sim} \leftarrow$ Sliding Window Algorithm(data: TS, threshold: ϵ);

```
// Create the representative time series (cut the simplified
   time series according to δ_t)
```

$TS^{ref} \leftarrow \langle (e_i, t_i) \rangle | (e_i, t_i) \in TS^{sim}, t_1 \leq t_i \leq t_1 + \delta_t$;

return TS^{sim} and TS^{ref}

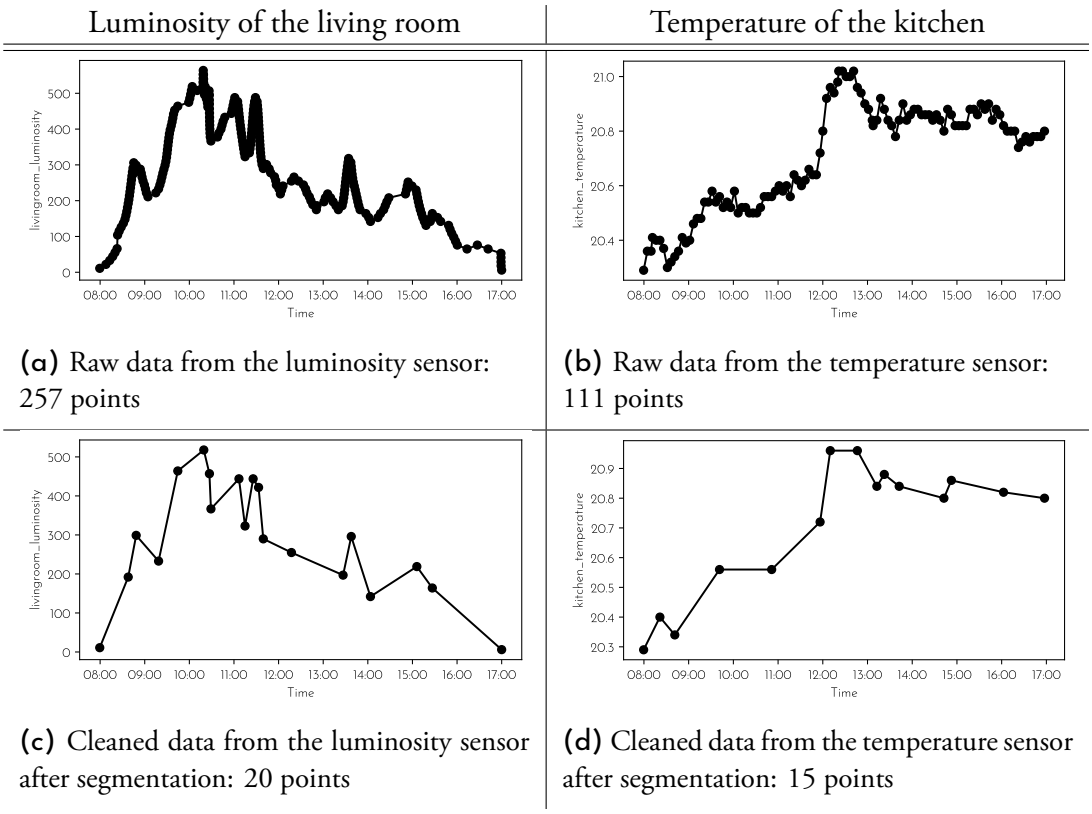
4.3.1.4 Experimentation

On quantitative data, this algorithm allows to simplify the general signal to keep only the macroscopic variations. To illustrate this point, here are two signals from the Orange4Home database [Cumin et al., 2017a], recorded on January 30, 2017 by a luminosity sensor in the living room (figure 4.6a) and a temperature sensor (figure 4.6b) in the kitchen. Figures 4.6c and 4.6d show these same signals simplified by the segmentation algorithm, with the parameters described above. If θ_{seg} increases, the signal rendered by the algorithm will contain even fewer points and will be even simpler, and if θ_{seg} decreases, the signal will contain more points of the original signal. As said before, θ_{seg} has been empirically set to 0.1, which simplifies the signal already, even if θ_{seg} remains primarily a parameter to be modified by the feedback part of the AmI system. Here, more than 92% of the points have been removed for the luminosity sensor and 86% for the temperature sensor.

However, we can already observe a limitation to this pre-processing. Figure 4.7a shows data from a voltage sensor for the kitchen oven. On most current sensors, the data varies a lot during the day, and this one is no exception. Thus, the algorithm provides less interesting results here, as shown in figure 4.7b because only 41% of the original signal points are removed. In this case, two options are possible:

- The observation period does not reflect the entire signal, i.e. the signal has even greater variations outside the observation period. Thus, the variations observed previously would be minor, and could be simplified by segmentation. In this case, a process extending the observation period should be put in place.

- The observation period reflects the entire signal. In this case, it is unlikely that rules can be found using this object, due to the high frequency of the observed variations, unless



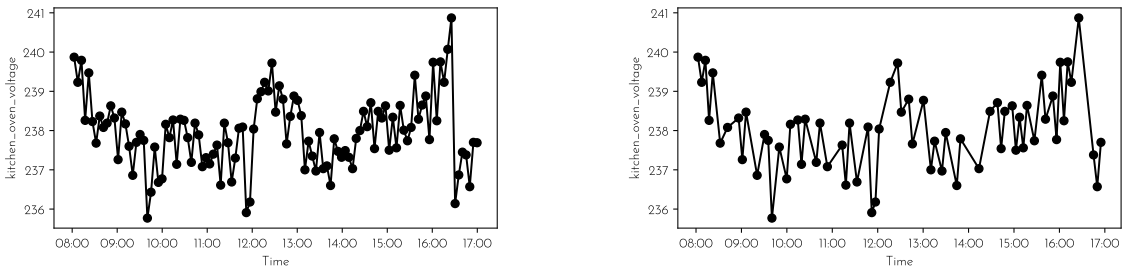
(a) Raw data from the luminosity sensor: 257 points

(b) Raw data from the temperature sensor: 111 points

(c) Cleaned data from the luminosity sensor after segmentation: 20 points

(d) Cleaned data from the temperature sensor after segmentation: 15 points

Figure 4.6: Result of the segmentation algorithm on quantitative data. Illustration with events sent by two sensors registered on January 30, 2017, Orange4Home database [Cumin et al., 2017a]



(a) Raw data from the voltage sensor: 111 points

(b) Cleaned data from the voltage sensor after segmentation: 66 points

Figure 4.7: Result of the segmentation on data from a sensor with high variations, where the cleaning is not as effective as in figure 4.6. Data from the voltage sensor monitoring the kitchen oven, January 30, 2017, Orange4Home database [Cumin et al., 2017a].

that frequency is an important information factor of the signal. In this case, it would be interesting to use signal processing functions, as mentioned in section 2.3.2.3, to transform the signal and bring out information from other indicators such as frequency.

We can assume that this is a special situation, but it seems important to see how this preprocessing can be put at fault. Indeed, a discretization technique alone cannot provide the keys needed to understand all signals, and must therefore be enhanced by other algorithms in some cases, or by user feedback.

4.3.2 Conversion into time series of segments

As mentioned above, the time series TS^{sim} is now simplified, and its segments, i.e. the consecutive pairs of points $(t_i, e_i), (t_{i+1}, e_{i+1})$, represent macroscopic variations. We formalize a segment as: $s = \{mean, variation, duration\}$. The three characteristics of the segment completely describe its linear variation, independently of the time of its appearance in the series:

- The mean: $mean = \frac{e_i.value + e_{i+1}.value}{2}$
- The variation: $variation = e_{i+1}.value - e_i.value$
- The duration: $duration = t_{i+1} - t_i$

To simplify the rest of the clustering part, TS^{sim} and TS^{ref} are converted into time series of segments using algorithm 3.

Other characteristics are possible to describe a segment. For example, we had previously considered the start value, end value and duration. However, we think that defining a segment by its linear variation allows us to get more information on the segment. For example, for a temperature sensor, it would be useful to know that it has increased a lot in a short period of time, or that it is stable for a few hours with an average of 20 degrees Celsius. This information is made available with the mean, variation and duration. We have chosen to use these characteristics in this system, but other choices remain possible.

As TS^{sim} and TS^{ref} are time series, it is necessary to attach a timestamp to the segments they contain. To do this, we choose to use the timestamp of the second point of the segment, i.e. the end timestamp. To explain this, let us go back to the building of the segments and their use: first, macroscopic variations are identified, and these variations are converted into segments. Then, similar segments form groups that are represented by atoms. Finally, atoms can be part of prediction rules, which can be validated by users, to become an effective automation. Thus, a macroscopic variation can be part of prediction rules, but also of effective automations. However, it can only be identified after it has taken place, and not at the beginning of its appearance. Using the end timestamp is therefore more relevant for the correct recognition of the variation.

4.3.3 Clustering

The main goal of Clustering is to identify groups of similar segments, each of which will define an atom, to create a time series of atoms. To do this, a distance measure is created to estimate the proximity between two segments, and then a clustering algorithm is applied. The main issue of this part is also the lack of *a priori* information on input data, i.e. variations. Indeed, having a set of segments representative of the time series makes it possible to estimate the

Algorithm 3: Translation of a time series of events into a time series of segments

Data: Time series of quantitative events $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle$
Result: Time series of segments $TS_s = \langle (t_1, s_1), ..., (t_{n_s}, s_{n_s}) \rangle$
// Initialization
$TS_s \leftarrow \langle \rangle$;
foreach *consecutive pair of points* $(t_i, e_i), (t_{i+1}, e_{i+1})$ **do**
  $mea \leftarrow \frac{e_i.value + e_{i+1}.value}{2}$;
  $var \leftarrow e_{i+1}.value - e_i.value$;
  $dur \leftarrow t_{i+1} - t_i$;
  $s \leftarrow \{mea, var, dur\}$;
  Add (t_{i+1}, s) at the end of TS_s;
return TS_s;

range of values of the characteristics of the segments, i.e. their mean, their variation, and their duration. The range of these three characteristics allows, as it will be explained here, to treat them fairly to measure the distance between two segments. For this reason, the segmentation algorithm produces a representative time series TS^{ref}, to estimate the variations that can be obtained from the connected object o.

- Input:
  - $TS^{sim} = \langle (t_1, e_1), ..., (t_{n_{sim}}, e_{n_{sim}}) \rangle, e_1, ..., e_{n_{sim}} \in E_o$: Time series of quantitative events
  - $TS^{ref} = \langle (t_1, e_1), ..., (t_{n_{ref}}, e_{n_{ref}}) \rangle, t_{n_{ref}} \leq (t_1 + \delta_t) < t_{n_{ref}+1} \leq t_{n_{sim}}$: Representative time series
  - $\theta_{clu} \in [0, 1]$: Clustering threshold. Here, it is estimated by the clustering algorithm itself.
- Output:
  - $TS_a = \langle (t_1, a_1), ..., (t_n, a_{n_a}) \rangle, a_1, ..., a_{n_a} \in A_o$: Time series of atoms

To analyze the similarities between segments, the clustering algorithm needs a distance measure.

4.3.3.1 Distance measure

The distance measure aims to assess the similarity between two segments by comparing their characteristics: averages, variations, and durations. This measure makes it easy to define whether two segments are similar or not, and will be useful for grouping similar segments.

As a reminder, a segment is formed by two consecutive points in the time series, and formalized as $s = \{mean, variation, duration\}$.

The distance measure will be built upon the means, variations, and durations of the segments. It must take those characteristics as evenly as possible.

This is a complex issue. How can we make a distance measurement that takes into account each characteristic equitably, knowing that we do not know in advance the value space of these characteristics?

We first thought of putting a weight on each characteristic in the distance measurement, each weight would be equal to the value space of the characteristic. We defined the weight by observing the segments produced during observation period δ_t. This allowed us to estimate the value space of each characteristic, and thus the weights.

However, another problem arose. Let us imagine that, due to lack of battery, for example, a temperature sensor stops sending data for two hours, whereas it usually sends data about every minute. Thus, of all the segments observed, only one will have a duration of two hours. This is therefore an outlier, because one of its characteristics differs too much from the most observed segments.

Taking this outlier into account changes the distance measurement. Indeed, as the duration distance between the outlier and another segment is very large, the distance between two segments that are not outliers will be comparatively very small. Thus, in this case, the duration would not be taken into account in the distance measurement because of the outlier.

To take into account the outliers, we first tried to estimate the data density using the Kernel Density Estimation tool, but without any convincing result. In the end, we decided not to take into account this type of segments. To identify outliers, the three characteristics of the segments are first normalized by their standard score, also known as z-score (x_z in equation (4.2)), obtained by the mean μ and the standard deviation σ observed in all the segments of the representative time series TS^{ref}. The original value can be retrieved from its normalization, as shown in equation (4.3).

$$x_z = \frac{x - \mu}{\sigma} \tag{4.2}$$

$$x = x_z * \sigma + \mu \tag{4.3}$$

Then, any segment that has at least one characteristic not bound between -3 and 3 is considered as an outlier, because one of its characteristics differs too much from the average of the segments observed. Therefore, this segment is not taken into account in the rest of the AmI system. This rule is usually applied to normally distributed data [Gorrie, 2016; Frost, 2019], which is not necessarily the case here. However, this method is an easy and efficient way to standardize segment characteristics while identifying outliers.

In addition, to take the mean into account as evenly as the variation and the duration, we use the Manhattan distance, also known as the Minkowski's L1 distance, city block distance, or taxi cab metric [Black and Pieterse, 2006]. The distance measure is defined as follows:

For two segments $s_1 = \{mean_1, variation_1, duration_1\}$ and $s_2 = \{mean_2, variation_2, duration_2\}$,

$$d(s_1, s_2) = \frac{|mean_2 - mean_1| + |variation_2 - variation_1| + |duration_2 - duration_1|}{18} \tag{4.4}$$

We wanted this distance to be limited between 0 and 1, since each characteristic is limited between -3 and 3, the basic distance is bounded between 0 and a maximum of $3 * 6 = 18$. To have this distance bounded as desired, we therefore divide it by 18.

4.3.3.2 Clustering Methodology

Now that we have a distance measure, we have to define a proximity metric to group similar segments into atoms. As a reminder, we have a major constraint: the system has no *a priori* knowledge of the input data, except for the fact that they are quantitative or categorical, where

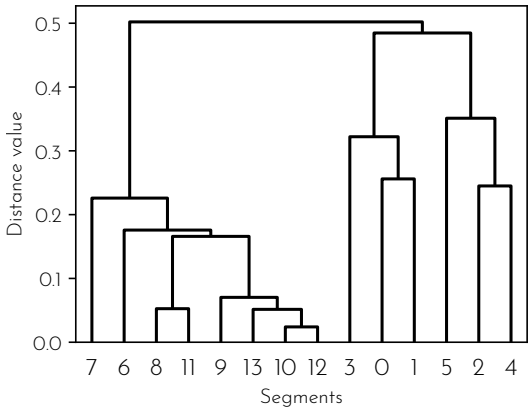


Figure 4.8: Representation of a dendrogram, result of the hierarchical clustering of the segments shown in figure 4.6d

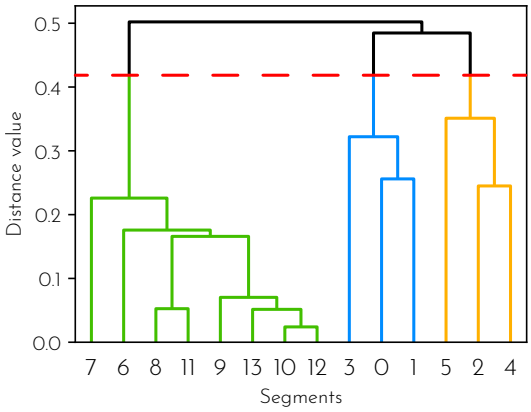


Figure 4.9: Cutting of the dendrogram at 0.479, resulting in new groups of segments that will form atoms

on the latter clustering is not applied. Thus, the value range, and the types of variations are not known in advance. Thus, defining a number of groups in advance may not be a good solution for this case, which is required by clustering algorithms such as k-means [MacQueen, 1967].

For this task, we have chosen to apply an agglomerative **hierarchical clustering** on all segments, based on the distance measure defined above. In this algorithm, each segment starts in its own group, and pairs of groups are merged according to their distance value, until all groups are merged. This makes it possible to estimate all the groups that can be built, over several granularities. These granularities range from the most precise, where the groups are composed of only one segment, to the most global, where one group contains all the segments. The result of this clustering can be visualized as a **dendrogram** (figure 4.8).

In this figure, the elements of the x-axis are the segments, numbered from 0 to 13. The y-axis represents a distance. We can see that segments 10 and 12 are very close to each other, because they join in the same group at a low distance. Segments 1 and 5, on the other hand, are further away. Thus, the dendrogram illustrates the merge of these segments into groups. It also illustrates the merge of groups among themselves, indicating at what distance they can merge.

To estimate the distance between a segment group and a segment, or between two segment groups, several linkage criteria are possible. For two groups of segments S_1 and S_2, we can have, among others [Tan et al., 2018]:

- Complete-linkage: $d(S_1, S_2) = max\{d(s_1 \in S1, s_2 \in S_2)\}$
- Single-linkage: $d(S_1, S_2) = min\{d(s_1 \in S1, s_2 \in S_2)\}$
- Mean-linkage: $d(S_1, S_2) = d(s_1, s_2)$, where s_1 and s_2 are the means of all the segments of S_1 and S_2 respectively

In our implementation, we have chosen mean-linkage clustering, because it allows us to obtain a segment representing the group, which will become an atom. The time complexity for this algorithm is $\mathcal{O}(n^2 log_n)$ [Manning et al., 2008]. Here, the representative segment of a group are formed by the average of the characteristics of the segments of the group, namely the mean, variation and duration. This is a proposal, and other representative segments are possible, such as the segment of the group closest to this average.

To get the desired atoms, a granularity must be chosen to obtain the segment groups. This can be expressed as a cut of the dendrogram. In the example shown in figure 4.9, the dendrogram is cut at a distance of 0.479, symbolized by the horizontal dashed line. This leads to the creation of three groups of segments, identified by colors: one with segments 6 through 13, one with segments 0, 1 and 5, and one with segments 2, 4 and 5. Here, the optimal granularity is defined by a known measure in Data Mining, called the **silhouette** [Rousseeuw, 1987], which is estimated from the time series representative of the data.

4.3.3.3 Silhouette

The silhouette criterion estimates the quality of the groups formed. To do this, it assumes that a good clustering forms groups that are far from each other, and that the elements within a group are close to each other. Thus, two indicators are considered in this measure:

- The inter-group distance, i.e. the distance between the groups formed, which must be maximized
- The intra-group distance, i.e. the distance between elements of the same group, which must be minimized

To explain in more detail, let us imagine that we have a set of clusters named S_1, S_2, ... up to S_n. For each segment s_i of S_i, it is possible to determine the average distance between this segment and the other segments of its cluster, S_i in this form (here, $|S_i|$ is the number of elements present in the cluster S_i):

$$a(s_i) = \frac{1}{|S_i| - 1} \sum_{s_j \in S_i, i \neq j} d(s_i, s_j) \tag{4.5}$$

It is also possible to define the smallest distance between this segment and the segments belonging to the other clusters:

$$b(s_i) = \min_{k \neq i} \frac{1}{|S_k|} \sum_{s_k \in S_k} d(s_i, s_k) \tag{4.6}$$

The silhouette value for this particular point is defined as follows:

$$silhouette(s_i) = \frac{b(s_i) - a(s_i)}{\max\{a(s_i), b(s_i)\}} \text{ if } |S_i| > 1, silhouette(s_i) = 0 \text{ if } |S_i| = 1 \tag{4.7}$$

This measure can be simplified as:

$$silhouette(s_i) = \begin{cases} 1 - a(s_i)/b(s_i), & \text{if } a(s_i) < b(s_i) \\ 0, & \text{if } a(s_i) = b(s_i) \\ b(s_i)/a(s_i) - 1, & \text{if } a(s_i) > b(s_i) \end{cases} \tag{4.8}$$

Finally, the silhouette measurement of a set of clusters is the average of the silhouette of all segments.

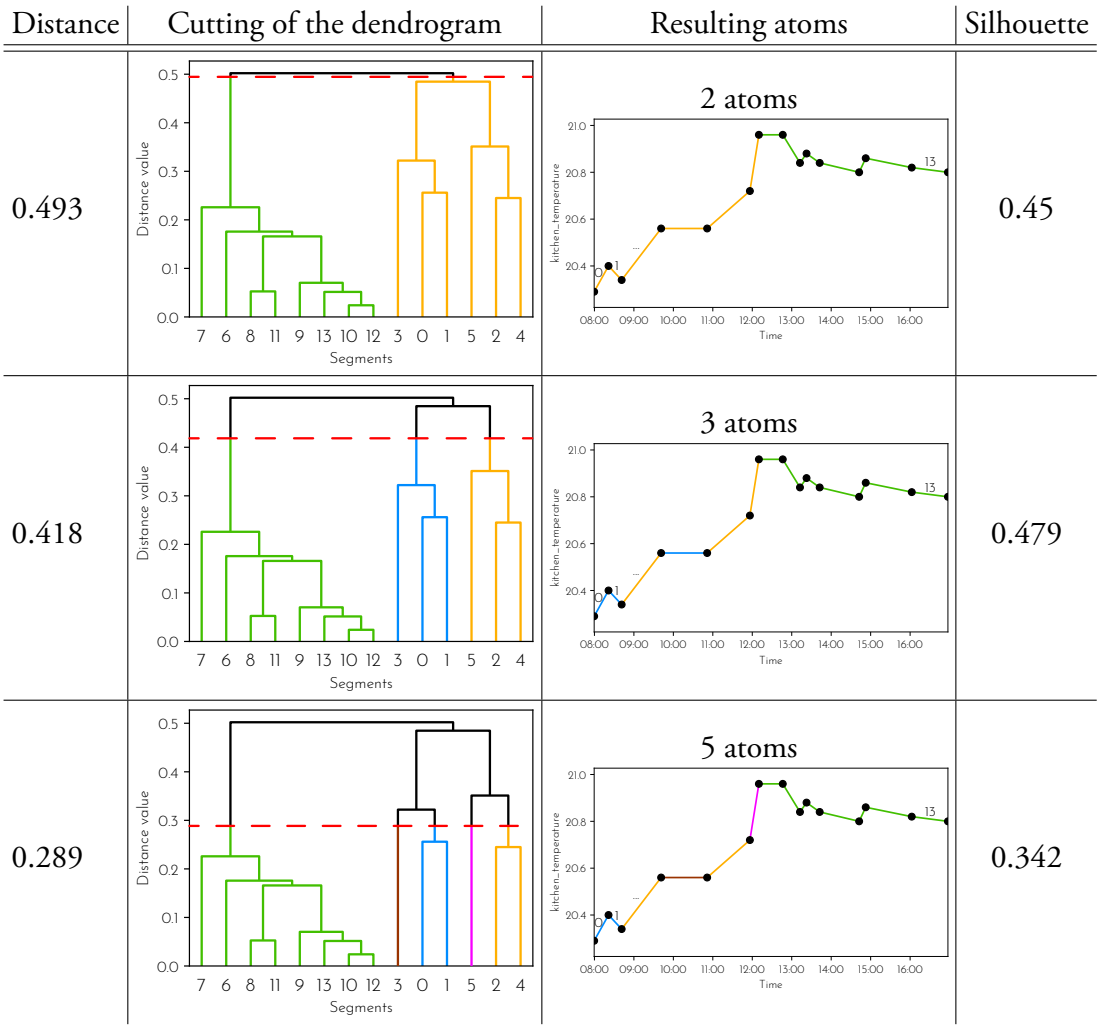


Figure 4.10: Result of the choice of the distance value on the formed atoms and the silhouette. Illustration with data from the kitchen temperature sensor taken on January 30, 2017, Orange4Home database [Cumin et al., 2017a]

4.3.3.4 Hierarchical Clustering

As said before, we apply a hierarchical clustering with the distance measure mentioned in this section. We use this clustering technique mostly because we can define the cluster configuration dynamically after the algorithm is run. Indeed, hierarchical clustering produces a dendrogram, a tree representing the hierarchy of possible clusters. In this tree, each hierarchy level is tied to a clusters' granularity. A hierarchy level is defined by the maximum distance that members of a same cluster can have. The AmI system chooses the clusters's granularity with a distance threshold (here, θ_{clu}), that can be changed anytime.

Furthermore, incremental hierarchical clustering algorithms exist that can update the cluster hierarchy when new segments are built from Segmentation [Chen et al., 2002; Widyantoro et al., 2002]. In our implementation, the AmI system runs in batch mode, i.e. it processes all the data at once. As said before, we use a mean linkage clustering [Tan et al., 2018] here, but other methods can be applied, as long as they can be run online.

As said before, this hierarchical clustering produces a dendrogram. θ_{clu} defines which set of clusters are kept. To choose θ_{clu}, the pre-processing does the following:

- A dendrogram is made from the set of all the segments of TS^{ref}.

- This dendrogram is cut on all distances values between two aggregations of segment groups, as can be seen in figure 4.10. For example, if an aggregation occurs at a distance of 0.5 and the following one at 0.7, the cut will be made at 0.6. This avoids making a cut too close to an aggregation.
- For every cut made, the silhouette value is computed according to the atoms built from the cut.
- θ_{clu} is the distance value of the cut that has the best silhouette value.

Then θ_{clu} does not change, as the segments are added to the clustering. Indeed, changing this value would change the structure of the end atoms. It is therefore necessary to use an algorithm that transforms the old atoms into the new representation, and to adapt the prediction rules accordingly. The formed groups can accommodate new segments, and new groups can be formed in addition to the old ones.

Then, an atom is tied to each of these groups, and comes from the segment representing the cluster. Thus, it has as information the averages of the means, variations, and durations of all the segments of the group to which the atom is linked. This atom is the representative of the group, and therefore of all the segments of the group.

The atoms built in the end represent a group of similar segments. They store the three indicators of the group of segments it is representing: the average of the means, variations and durations of all the segments from that group. In the end, a time series of atom is produced, each segment being replaced by the atom of the cluster in which it is located: $TS_a\langle(t_1, a_1), ..., (t_{n_a}, a_{n_a})\rangle, a_1, ..., a_{n_a} \in A_o$. For more details about hierarchical clustering, a complete pseudocode can be found in appendix A at algorithm 8.

4.3.3.5 Experimentation

As with segmentation, it is difficult to evaluate clustering itself. The characteristics used for distance measurement, the clustering algorithm and the threshold used to define the atoms depend entirely on the desired goal. However, we can show some results on the signals shown in section 4.3.1.4, namely the temperature sensor and the brightness sensor in the Orange4Home database [Cumin et al., 2017a].

In figure 4.10, we can observe the effects of the choice of the cutting distance on the created atoms. The greater the cutting distance, the fewer atoms there are, and these atoms are more generalized, i.e. they represent more different groups of segments. Having too high a cutting distance makes the atoms too generalized, and no longer accurately describes variations in the original signal. Conversely, having the cutting distance too low produces too many atoms, and similar variations may not be gathered within the same atom. Even knowing these extremes, it is difficult to choose the optimal distance. This is why the silhouette measure, recognized in the field of clustering, was chosen to obtain the optimal distance. In figure 4.10, the optimal distance is 0.418, which produces 3 atoms, because the silhouette measurement is at the highest.

4.4 Conclusion

In this chapter, we have presented a contribution of the thesis by proposing a solution to one of the building blocks of the AmI architecture. The main purpose of pre-processing is to take into account only variations, whether the events are quantitative or categorical. This goal is

achieved by cleaning duplicates for categorical data and discretization for quantitative data. Discretization comes to life with state-of-the-art algorithms in the fields of segmentation and clustering. This is therefore a first basis on which the AmI system can be based, and which can evolve, as we will discuss it in chapter 6. This pre-processing will be used in the evaluation of the system in section 5.5.

With these algorithms, the data is now unified. The system now processes atoms, which are categorical data describing variations. These time series of atoms are the basic material of the algorithm presented in the next chapter: prediction rule mining.

Chapter 5

Rule mining

5.1 Introduction

To finish the explanation of the AmI system, we focus on the major contribution of the thesis: the algorithm for searching prediction rules, which is essential for identifying automation proposals.

First, through the state of the art, we highlight the problems encountered in current rule search algorithms, which are not in line with our objective. Even trying to adapt our framework to one of these algorithms creates its own set of problems.

For this reason, a new rule search algorithm has been created: TSRuleGrowth. Based on the principles of TRuleGrowth present in the state of the art [Fournier-Viger et al., 2015], TSRuleGrowth is fully adapted to time series. This chapter details the algorithm, its principles, and its advantages. But also, it describes the performances of this algorithm on several real databases, coming from connected environments. These results validate the approach envisaged with this algorithm, as it allows to discover habits that were not known in advance.

5.2 Background

As said in section 3.2.3, this algorithm must find partially ordered prediction rules containing multisets. This means that the condition and prediction of these rules are multi-sets of atoms, i.e. unordered sets where an atom can appear several times. For the rest of this chapter, it is important to clarify a few concepts.

5.2.1 Input: a time series of atoms

As an input to the rule search algorithm, we have a time series of atoms from all connected objects. It is formalized as $TS = \langle (t_1, I_1), ..., (t_n, I_n) \rangle$, $I_1, ..., I_n \subseteq A$, where:

- t_i is a time stamp, i.e. the time coordinates of the occurrence.
- $I_i \subseteq A$ is an itemset, i.e. the set of individual atoms of A which are observed at time stamp t_i, and are sent by one or several objects.
- A is the set of all atoms.
- TS is ordered by the time stamps.

5.2.2 Output: prediction rules

A prediction rule is noted $r : A_c \Rightarrow A_p$, where A_c is the condition, and A_p is the prediction of the rule. r describes that if A_c is observed, A_p will be observed after a certain time.

In our case, we want to look for rules that can propose several actions based on several observations. As seen in section 5.2.1, we do not consider an order within the condition or prediction, but we consider the number of appearances of atoms. This is why A_c and A_p are both multisets, i.e. sets of atoms where an atom can be present several times. This means that we want the rule search algorithm to be able to find rules with several atoms in the condition and prediction, not just one.

In our case, two points should be considered in the rules that the AmI system should search for:

- The condition part of a rule contains only atoms from sensors, actuators, or time indicators. As stated in 3.2.3, interfaces are not considered.
- The prediction part of a rule contains only atoms from actuators. Indeed, we are looking for automation proposals, which means having only actions at the end.

5.2.3 Data structures in rule mining

Prediction rules can be searched on several data structures. The two best known structures are:

- A set of **transactions**, which are simply sequences, i.e. ordered lists of elements, and in our case atoms. Here, no notion of time, but the order of instantiation of the atoms is preserved. Transactions are most commonly used, among other things, to find rules on online shopping histories. Here, a transaction represents a shopping list made by a customer, where the order of adding products to the shopping cart is kept. All transactions are generally all the lists of purchases made by the buyer. The prediction rules

are intended to determine which next product the customer will add to his list, from the products present in his current shopping list.

- A **time series**, which is the structure used in this thesis. It is described in section 3.2.2. Unlike transactions, we have only one time series, not several, and the time series keeps the notion of time in atom instantiations, which is useful for our automations.

5.2.4 Validation of a rule

For a prediction rule to be validated, it must be:

- **Frequent**, i.e. it happens a fairly high number of times
- **Reliable**, i.e. the prediction must happen after the condition in almost all cases. Indeed, depending on the use case, a percentage of failures can be tolerated

5.2.4.1 Support

In rule mining, to check that a rule is frequent, its **support** is calculated. The notion of support depends on the structure of the input, but can be applied to a rule, a set of atoms, or an atom, to estimate their frequencies. To simplify the following, we consider the case for an atom named x, but this is also applicable for multisets, and rules. Two types of supports exist:

- The absolute support, called sup, which determines the absolute number of times x is encountered, as an integer.
- The relative support, called $relSup$, bounded between 0 and 1, determines the frequency of appearance of x.

To illustrate, let us take the case of transactions. Here, the most common support measure is constructed as follows: imagine a set of transactions named TR, containing transactions named $tr_1, tr_2, ..., tr_n$. The absolute support of x is the number of transactions where x appears.

$$sup(x) = |tr_i, x \in tr_i \wedge tr_i \in TR| \tag{5.1}$$

The relative support of x is equal to its absolute support, divided by the number of transactions in TR.

$$relSup(x) = \frac{sup(x)}{|TR|} \tag{5.2}$$

For time series, several support measures exist, which will be described in section 5.3.

5.2.4.2 Interest

To ensure that a rule is reliable, its **interest**, named int, is calculated. Several measures can estimate the interest of a rule. These measures depend on the supports of the rule r, the condition A_c and the prediction A_p.

The best known is confidence [Azevedo and Jorge, 2007], formalized as:

$$confidence(r : A_c \Rightarrow A_p) = \frac{relsup(r)}{relSup(A_c)} \tag{5.3}$$

The confidence measure is bounded between 0 and 1 and is widely used in rule searching. However, it does not test the independence between the occurrences of A_c and those of A_p. This is a problem, because in our case, we are trying to test this independence. For the following example, let us ignore the fact that a rule must have a prediction part composed only of actuators. Imagine a weather station that very often sends back the "sunny" and "cloudy" atoms, because the weather outside oscillates between the two. Let us also imagine a door opening sensor. During the day, a person can open the door several times. Thus, it is highly likely to see the "open door" $\Rightarrow$ "sunny" and "open door" $\Rightarrow$ "cloudy" rules. With the confidence measure, these rules are validated, because every time the door opens, the "sunny" and "cloudy" atoms are seen, which makes no sense. It is precisely by testing the independence of the occurrences of the condition and the prediction that we can avoid this case.

Known alternatives in the field of rule mining exist, such as conviction [Azevedo and Jorge, 2007], lift [Azevedo and Jorge, 2007], but they are not bounded, as their values can go to infinity, making it difficult to set a minimum value from which a rule can be considered reliable.

$$conviction(r : A_c \Rightarrow A_p) = \frac{1 - relSup(A_c)}{1 - confidence(r)} \tag{5.4}$$

$$lift(r : A_c \Rightarrow A_p) = \frac{confidence(r)}{relSup(A_c)} \tag{5.5}$$

It is therefore important to look for other, less known measures of interest that are bounded, and test the independence between the occurrences of A_c and those of A_p.

In the following section, we will review the existing algorithms that could solve our problem of finding partially-ordered prediction rules over time series. We will see that the existing algorithms do not fully address the problem, and that the proposed support measures, essential to identify the rules, are also not satisfactory.

5.3 State of the art

As said before, the AmI system needs an algorithm for mining partially-ordered prediction rules over a time series of atoms. Thus, this algorithm is at the crossroads of two research domains, which are rule mining on time series and partially-ordered rule mining. We will therefore study a state of the art in these two areas.

5.3.1 Rule mining on time series

[Das et al., 1998] proposes a system mining basic rules on a sequence of atoms, where one atom predicts another. Those atoms represent simple variations of stock market data. It can also search for more complex rules, where the condition is a sequence. This system therefore makes it possible to mine prediction rules over a time series. However, it seeks fully-ordered prediction rules, rather than partially-ordered ones. Also, the prediction part of the rules is limited to a single atom, a limitation that we want to avoid in our AmI system. [Schlüter and Conrad, 2011] can be considered as an improvement over [Das et al., 1998], because this system looks for rules where the prediction is not limited to a single atom. But, since it seeks fully-ordered rules, this system cannot be applied in our case.

[Mannila et al., 1997] introduces a notion of support for a time series, via a sliding window with a determined duration. The support of an atom, a set of atoms or a rule is the number

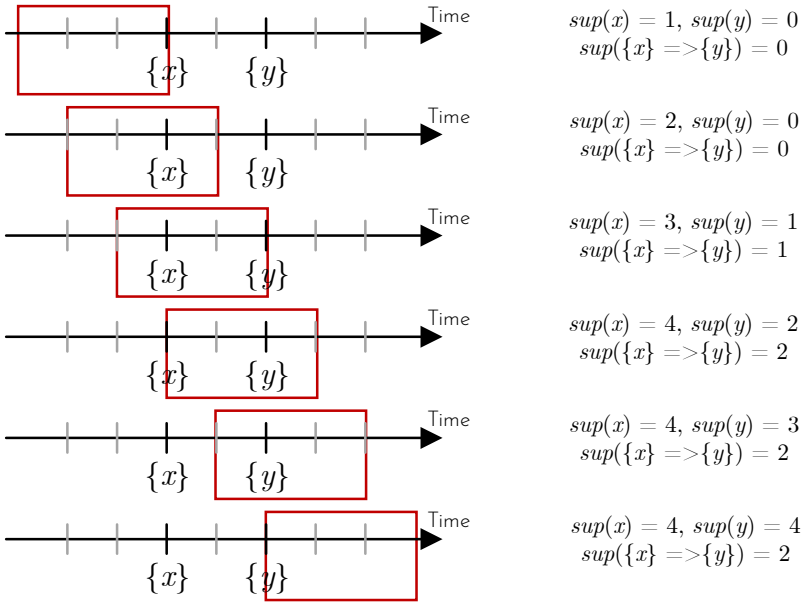


Figure 5.1: Example illustrating the problems of the notion of support on time series defined in [Mannila et al., 1997]

of windows in which this atom, set or rule appears. This algorithm finds partially-ordered rules, first finding sequences of atoms that are frequent, then dividing these sequences into two sub-sequences in any possible way to determine whether they form a prediction rule or not. Other algorithms use this notion of support, including [Deogun and Jiang, 2005] which finds rules which prediction is composed of one single atom. The algorithm presented in [Mannila et al., 1997] can therefore be applied in our case. But this support definition can be problematic: the atoms of A_p being strictly later than A_c, the number of windows covering the rule r will be strictly lower than the number covering A_c. Even if A_p always appears after A_c, the support of the rule will be lower than that of A_c, reducing its interest.

In the example in figure 5.1, it is easy to visualize the problem. Here, the sliding window advances at regular time steps, but the same problem would arise if this window advanced one itemset at a time. Here, the sliding window covers the rule less often than it does the atoms that compose it, which is normal, because these atoms are distant from each other. Thus, in view of the different support values, the rule will not be validated. For example, the confidence measure of this rule would be 0.5, i.e. the rule would be visible only once out of two occurrences of x. However, this is not true in this example.

Furthermore, since the search is structured in two steps in [Mannila et al., 1997] and [Deogun and Jiang, 2005] (mine frequent sets, then search for rules), those algorithms are not fully efficient.

As we can see, there is no algorithm to solve our problem with the fixed constraints. In addition, the notions of support in time series also have problems inherent in their definition. Let us now look at the state of the art of partially-ordered rule mining, which does not necessarily take into account a time series, to see if it is possible to find a source of inspiration.

5.3.2 Partially-ordered rule mining

We have already discussed [Mannila et al., 1997] before, and shown the problems of its notion of support, and in its two-step rule search. To our knowledge, few other algorithms of partially-ordered sequential rule mining exist. The most known are RuleGrowth [Fournier-Viger et al.,

2015], and its variations, TRuleGrowth [Fournier-Viger et al., 2015] and ERMiner [Fournier-Viger et al., 2014]. These algorithms take as input a set of transactions.

RuleGrowth directly searches for prediction rules, unlike [Mannila et al., 1997] that searches for frequent itemsets and then searches for rules on these itemsets. In addition, the incremental architecture of RuleGrowth allows to limit the size of the searched rules, and to limit the atoms in which rules are searched.

RuleGrowth allows this limitation directly during the search, reducing the total computation time. TRuleGrowth is an extension of RuleGrowth that accepts the constraint of a sliding window, determined by a number of consecutive itemsets. It allows to limit the search to rules that can only occur in this window. ERMiner is presented as a more efficient version of RuleGrowth, but without an extension that accepts a sliding window. However, those algorithms have a major problem in the proposed use case: they take transactions instead of a time series. The notion of support depends directly on the structure of transactions, and cannot be straightforwardly applied on a time series. Despite the advantages of these algorithms, they cannot be applied directly to our input data.

5.3.3 Scientific problems

To our knowledge, the state-of-the-art algorithms are not satisfactory enough to solve the initial problem. Two major issues need to be solved:

1. How to define the support of a rule in a time series that avoids the problem encountered in section 5.3.1, preventing the validation of certain rules?

2. How to build a rule mining algorithm upon this new support measure?

In addition, this algorithm must address the following:

3. How to limit the duration of the found rules?

4. How to limit the search to certain atoms in the condition or prediction?

5. How to avoid that a rule is found twice?

RuleGrowth answers points 4 and 5, but only takes transactions as input. Its extension, TRuleGrowth, uses a sliding window that can be used to answer to the third problem with some modifications. The following section describes an adaptation of the AmI data to be accepted by TRuleGrowth, and raises limitations of this adaptation. After, we describe our algorithm: TSRuleGrowth. It uses the principles of TRuleGrowth, but applies them to time series, to deal with the first two problems.

5.3.4 Adapting time series to TRuleGrowth

To solve the problem of the input data of these algorithms, one can simply convert the time series into a list of transactions, as in figure 5.2. To do this, this time series (step 1 in the figure) is divided into limited subsets with a defined duration noted Δ_{tr} (steps 2 and 3 in the figure). Then the timestamps of the small time series are removed, to keep only the order of appearance of the atoms (step 4 in the figure). Without this notion of time, these are no longer time series, but rather sequences of atoms, in other words, transactions.

But the main problem of this implementation is the calculation of the support of a rule. Let us take the following example with three transactions:

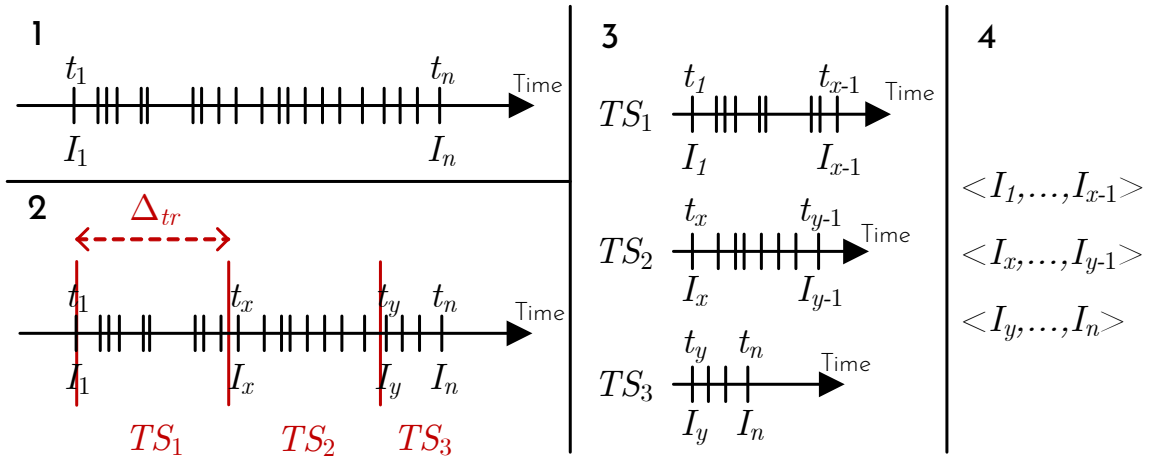


Figure 5.2: Example of conversion of a time series into transactions

$$\langle\{x\},\{x\},\{y\},\{x\},\{x\}\rangle$$
$$\langle\{x\},\{y\},\{x\},\{x\},\{x\}\rangle$$
$$\langle\{x\},\{x\},\{y\},\{x\},\{x\}\rangle$$

Here, $\{x\} \Rightarrow \{y\}$ is considered valid, because its support is 3, the same as x and y. As a reminder, in the context of transactions, the support of x is the number of transactions in which x appears, as stated in equation (5.1). As long as a rule has only been seen once in a transaction, it is considered present throughout that transaction, even if it could have been invalidated, as in the example: x can be seen without y after, in all the transactions. Cutting a time series into transactions can lead to rules that are validated by mistake. There are other problems, inherent in the choice of the value of Δ_{tr}. Having a small Δ_{tr} can increase the risk of a rule being "split in two", i.e. whose occurrence is separated between two transactions, which reduces interest. Having a large Δ_{tr}, over a time series, reduces the absolute support of the rules the system is looking for, because fewer transactions are generated from the time series.

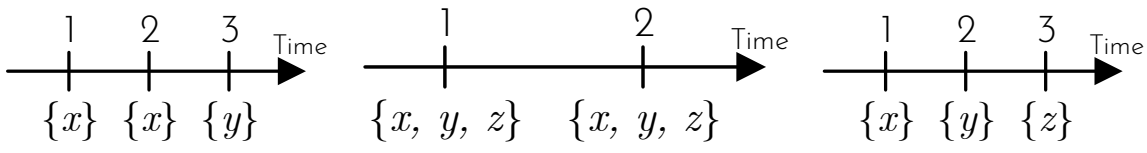
Converting a time series into a set of transactions can be applied in the proposed use case. However, the above limitations have led us to create a new algorithm, inspired by TRuleGrowth, which is fully adapted to time series.

5.4 TSRuleGrowth

5.4.1 Inputs, Outputs

This section outlines the proposed rule mining algorithm on a time series of atoms: TSRuleGrowth, for "Time Series RuleGrowth". This algorithm is incremental, and can limit the search to certain atoms in the condition and prediction. TSRuleGrowth takes as inputs:

- $TS = \langle(t_1, I_1), ..., (t_n, I_n)\rangle, I_1, ..., I_n \subseteq A$: A time series of atoms, where:
  - t_i is a time stamp, i.e. the time coordinates of the occurrence.
  - $I_i \subseteq A$ is an itemset, i.e. the set of individual atoms of A which are observed at time stamp t_i, and are sent by one or several objects.
  - A is the set of all atoms.



(a) Example 1 (b) Example 2 (c) Example 3

Figure 5.3: Examples of time series

  - TS is ordered by the time stamps.

- min_{sup}: The minimum absolute support for a rule to be frequent
- min_{int}: The minimum interest for a rule to be reliable
- $window$: A time frame in which the rules must occur

TSRuleGrowth produces partially-ordered prediction rules using multisets, detailed in section 3.2.3. In the proposed use case, the prediction part of the rules is only composed of atoms coming from actuators. Since TSRuleGrowth takes a time series as input instead of a list of transactions, some notions need to be redefined: the support, the interest, and how to record the occurrences of a rule.

5.4.2 Metrics

5.4.2.1 Support

Distinct or possible occurrences? As stated in section 5.3.1, the current notions of support on time series do not suit our problem. We want to propose a new notion of support, which avoids the problems encountered in the state of the art.

This is an issue that provoked a considerable amount of debate during the course of the thesis, which can be summarized in the following question. In figure 5.3a, are there 1 or 2 occurrences of $\{x, y\}$? In other words, should we count the distinct occurrences of $\{x, y\}$, i.e. 1 because there is only one y, or the possible occurrences, i.e. 2, located at the timestamps [1, 3] and [2, 3]? From this simple question flows the whole notion of support, so it is crucial.

So, should we choose distinct occurrences or possible occurrences? To answer this question, let us try to define the relative support, which must be bounded between 0 and 1, as stated in section 5.2.4.1:

- For distinct occurrences, it is possible to divide the absolute support by the total number of itemsets. This would remain limited between 0 and 1, as in the example in figure 5.3b. For an atom, multiset, or rule a:

$$relSup(a) = \frac{sup(a)}{|(t_z, I_z) \in TS|} \tag{5.6}$$

- For possible occurrences, dividing the absolute support by the total number of itemsets will not work. In figure 5.3b, there are 8 possible occurrences of $\{x, y, z\}$, and there are only 2 itemsets present in the time series. Thus, the relative support, which must be bounded between 0 and 1, cannot be equal to $8/2 = 4$.

  The maximum number of possible occurrences for an atom, multiset, or rule a, is $|(t_z, I_z) \in TS|^{|a|}$, i.e. the number of itemsets in the time series to the power of the

number of atoms inside a. The relative support should therefore be defined as follows. For an atom, multiset, or rule a:

$$relSup(a) = \frac{sup(a)}{|(t_z, I_z) \in TS|^{|a|}} \tag{5.7}$$

Now that we have an idea of the notions of absolute and relative supports in both cases, let us compare them in a simple example. In figure 5.3c, let us compute the relative supports of x, $\{x, y\}$ and $\{x, y, z\}$, simple multisets.

- With the distinct occurrences:
  - $sup(x) = 1$, so $relSup(x) = 1/3$
  - $sup(\{x, y\}) = 1$, so $relSup(\{x, y\}) = 1/3$
  - $sup(\{x, y, z\}) = 1$, so $relSup(\{x, y, z\}) = 1/3$
  - Which is normal after all, because we only see these multisets once. Regardless of their size, the notion of support does not change.
- With the possible occurrences:
  - $sup(x) = 1$, so $relSup(x) = 1/3^1 = 1/3$
  - $sup(\{x, y\}) = 1$, so $relSup(\{x, y\}) = 1/3^2 = 1/9$
  - $sup(\{x, y, z\}) = 1$, so $relSup(\{x, y, z\}) = 1/3^3 = 1/27$
  - Here, we see that the relative support decreases with the size of the multiset.

It is for this reason that we have decided to choose the distinct occurrences, to avoid the difference in treatment of multisets according to their size. Now that we have answered this question, the next paragraph will define the notion of support that we have built.

Definition of the new support metric For a time series TS noted $\langle (t_1, I_1), ..., (t_{n_s}, I_{n_s}) \rangle$ where I_i is an itemset and t_i is an associated timestamp, the support of an atom a, noted $sup(a)$, is defined as the number of itemsets containing a (equation (5.8)).

$$sup(a) = \left|(t_i, I_i) \in TS | a \in I_i\right| \tag{5.8}$$

The absolute support of a multiset of atoms A_m is the number of distinct occurrences of all atoms of A_m within the time window. If an occurrence of an atom of A_m has contributed to an occurrence of the multiset A_m, it can no longer contribute to other occurrences of A_m. The examples in figure 5.4 can help to understand this concept more easily.

The support counting algorithm, Count, scrolls a window on the time series. A simplified pseudocode can be found in algorithm 4, and a more detailed one in appendix B, namely algorithm 9. If all atoms of A_m are seen, their occurrences will be blacklisted to prevent them from being involved in another occurrence of A_m. This ensures that the definition of the support is respected. If several occurrences of the same atom of A_m are seen in the same window, only the earliest ones are blacklisted. It leaves to newer ones the possibility to contribute to a future occurrence of A_m. The absolute support of $r : A_c \Rightarrow A_p$ is the distinct number of occurrences where all the atoms of A_c are observed, followed by all the atoms of

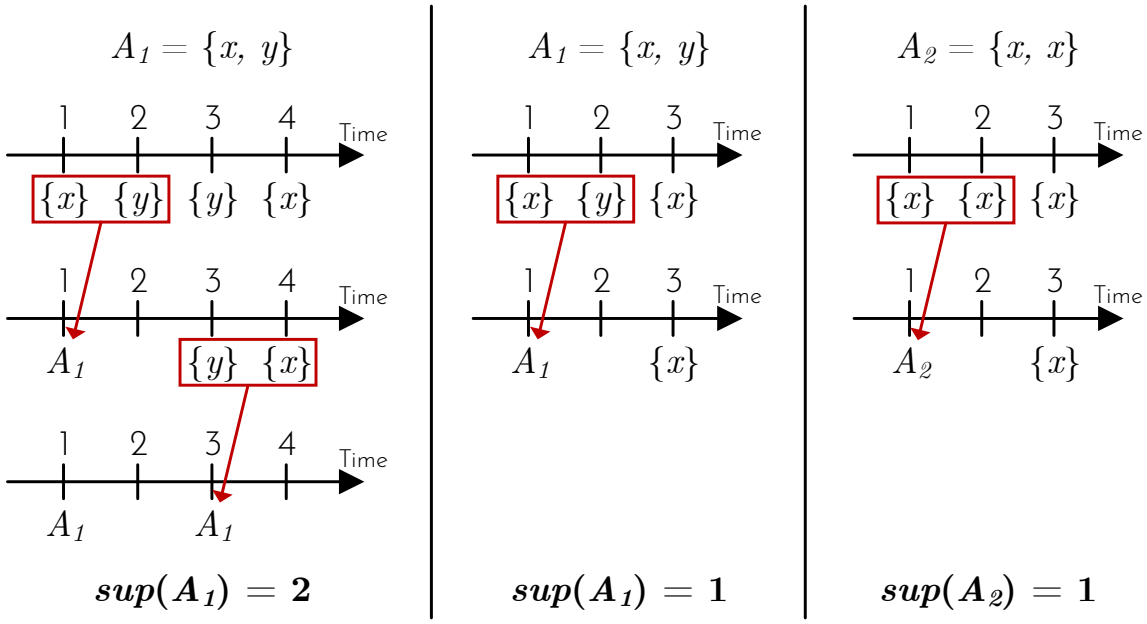


Figure 5.4: Support calculation examples. Each column represents a step-by-step example of support calculation

Algorithm 4: Count : support counting algorithm

Data: A_m: multiset, $TS = \langle (t_1, I_1), ..., (t_n, I_n) \rangle$, $I_1, ..., I_n \subseteq A$: time series, $window$: duration

`// Initialization`

Assign an empty blacklist $b(a)$ to every unique atom $a \in A_m$;

$sup(A_m) \leftarrow 0$; `// Support of` A_m

`// Sliding window through the time series`

while *the window has not reached the end of* TS **do**

  found $\leftarrow$ True;

  Scan the window, record the timestamps of $a \in A_m$ in $T(a)$;

  foreach *atom* $a \in A_m$ **do**

    $T(a) \leftarrow T(a) \setminus b(a)$;

    if $|T(a)| <$ *multiplicity of* a *in* A_m **then**

      found $\leftarrow$ False ; `// No distinct occurrence`

  if *found is True* **then**

    $sup(A_m)$ += 1;

    foreach *atom* $a \in A_m$ **do**

      `// Add the earliest timestamps of` $T(a)$ `to the blacklist of` a

      $m \leftarrow$ multiplicity of a in A_m;

      $b(a) \leftarrow b(a) \cup m$ earliest timestamps of $T(a)$;

  Slide the window by one itemset;

Return $sup(A_m)$;

A_p. The atoms of A_c and A_p also have blacklists, grouped into two sets: one for the atoms of A_c, and one for those of A_p.

The relative support of an atom a, a multiset A_m or a rule r, noted $relSup$, is its absolute support divided by the total number of itemsets in the time series (equation (5.9)). This support can be applied to partially-ordered rules, unlike the one detailed in [Das et al., 1998; Schlüter and Conrad, 2011], which search for fully-ordered rules. In addition, it avoids the defect in the support detailed in [Mannila et al., 1997], which is explained in section 5.3.1.

$$relSup(r) = \frac{sup(r)}{|(t_z, I_z) \in TS|} \tag{5.9}$$

5.4.2.2 Interest

In TSRuleGrowth, one can compute the interest of a rule through its confidence, conviction or lift as mentioned in section 5.2.4.2. In the proposed use case, we chose a metric that is not widespread and rarely mentioned in the field: netconf [Ahn and Kim, 2004]. Unlike confidence, netconf tests the independence between occurrences of A_c and those of A_p. Also, unlike conviction and lift, it is bounded between -1 et 1, 1 showing that A_p has a high chance of appearing after A_c, -1 that A_p has a high chance of not appearing after A_c, and 0 that this chance is unknown. It is for these two reasons that we have chosen this metric.

For a rule $r : A_c \Rightarrow A_p$:

$$netconf(r) = \frac{relSup(r) - relSup(A_c) \times relSup(A_p)}{relSup(A_c) \times (1 - relSup(A_c))} \tag{5.10}$$

$netconf(r)$ can also be written as $netconf(relSup(A_c), relSup(A_p), relSup(r))$.

5.4.3 Recording of rule occurrences

An occurrence of r is decomposed as the occurrence of A_c and A_p. Indeed, an atom can be found in both A_c and A_p, and it is necessary to distinguish the occurrences of this atom in A_c from those in A_p. An occurrence of a multiset is recorded in an associative array, which is a collection of pairs of (key, value), where each key is unique. Here, the keys are the distinct atoms of the multiset, and their values are the set of timestamps where the atoms are observed.

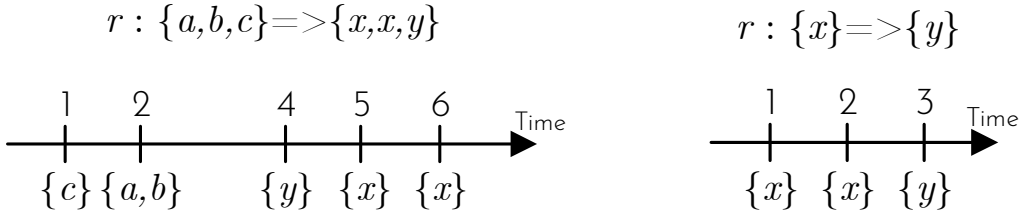
Let us take the example of $r : \{a, b, c\} \Rightarrow \{x, x, y\}$ in figure 5.5a. Here, the occurrence of A_c is $\{a$:$\{2\}$, b:$\{2\}$, c:$\{1\}\}$ and the occurrence of A_p is $\{x$:$\{5, 6\}$, y:$\{4\}\}$. Two timestamps are recorded for x, because it is present twice in A_p.

To reduce memory use, an occurrence of a multiset can also be stored as a list of timestamps, provided that the multiset is ordered. Within the list, the index of a timestamp is the same one as the index of the linked atom in the multiset. In the previous example, the occurrence of $A_p = \{x, x, y\}$ is recorded as [5,6,4]. Multiple occurrences of a multiset are recorded as a list of these structures. All occurrences of the rule are recorded in two lists, for A_c and A_p.

5.4.4 Principles

5.4.4.1 Principles shared with TRuleGrowth

TSRuleGrowth takes the principles of TRuleGrowth and applies them to time series. The algorithm uses a sliding window, to limit the search. But, unlike TRuleGrowth where the



(a) Example 1 (b) Example 2

Figure 5.5: Examples of rules and time series

window is a number of consecutive itemsets, TSRuleGrowth has a time sliding window. It allows to restrict the search, and to have an estimate of the lifetime of a rule.

Furthermore, this algorithm first finds basic rules, where one atom can predict another. Then, recursively, it extends them, by adding an atom in A_c or A_p, via ExpandCondition and ExpandPrediction. This mechanism allows, if necessary, to limit the maximum length of the rules to be searched, i.e. the maximum number of atoms in A_c and A_p. Then, TSRuleGrowth applies two principles of TRuleGrowth to avoid finding duplicate rules. First, ExpandPrediction cannot be called by ExpandCondition. Second, ExpandCondition and ExpandPrediction can add an atom only if its identifier is larger than those of all the atoms of A_c or A_p. Indeed, as mentioned in section 3.2.2, each atom has a unique identifier, in the form of a number. In our implementation, the order between atoms is simply the numerical order. For this algorithm, the order between the atoms can also be defined manually, or according to another method.

5.4.4.2 New Principles

Let us take the example in figure 5.5b. For this rule r, even if $sup(r) = 1$, two occurrences of the rule are possible: $\{x{:}\{1\}, y{:}\{3\}\}$ and $\{x{:}\{2\}, y{:}\{3\}\}$. This problem is inherent in time series: we cannot know *a priori* which occurrence will be useful for an extension of this rule. To do this, TSRuleGrowth tries to extend all of the seen occurrences of this rule. In addition, TSRuleGrowth does not use the same rule structure as TRuleGrowth: instead of being sets, A_c and A_p are multisets. Therefore, a principle coming from TRuleGrowth needs to be modified: ExpandCondition and ExpandPrediction can add an atom if its identifier is greater than those of all the atoms of A_c or A_p, but also if it is equal to the greatest atom of A_c or A_p, according to the numerical order between atoms identifiers. But a new problem of duplication arises. In figure 5.5b, if we try to grow $\{x\}\Rightarrow\{y\}$ to $\{x, x\} \Rightarrow \{y\}$, the same occurrence will be found twice. $\{x{:}\{1\}, y{:}\{3\}\}$ will extend to $\{x{:}\{1, \mathbf{2}\}, y{:}\{3\}\}$, by adding the timestamp 2, and $\{x{:}\{2\}, y{:}\{3\}\}$ will extend to $\{x{:}\{\mathbf{1}, 2\}, y{:}\{3\}\}$, by adding the timestamp 1. To avoid this, TSRuleGrowth does the following: if the rule extends to the greatest atom of A_c or A_p, it should only record the timestamps of that atom that occur **strictly later** than the last timestamp of that atom in the base rule. Thus, in the previous example, the first occurrence is recorded, not the second.

5.4.5 Algorithm

5.4.5.1 Main loop

Like TRuleGrowth, the main loop tries to find basic rules, i.e. rules which conditions and predictions are composed of only one atom. To do this, it computes the support for all

basic rules that can be created in the time series. If one of these rules has a support higher than min_{sup}, it tries to grow it, by adding an atom in A_c (ExpandCondition), then in A_p (ExpandPrediction). Finally, it computes its interest for validation. As mentioned earlier, the algorithm computes all distinct occurrences of the rule for its support, but also all possible occurrences for the expansion of the rule. To do this, TSRuleGrowth uses a blacklist system to discern occurrences. Multiprocessing can be added to TSRuleGrowth, by treating all basic rules and expansions in parallel, to reduce the execution time. A simplified pseudocode can be found in algorithm 5, and a more detailed one in appendix B, namely algorithm 10.

Algorithm 5: TSRuleGrowth

Data: TS: time series, min_{sup}: minimum support, min_{int}: minimum interest, $window$: duration

Scan TS once. For each atom a found, record the timestamps of the itemsets that contains a in $T(a)$;
`// Creation of basic rules`
foreach *pair of atoms i, j* **do**
  $sup(i \Rightarrow j) \leftarrow 0$; `// Support of the rule`
  $O_c(i \Rightarrow j), O_p(i \Rightarrow j) \leftarrow []$; `// Occurrences of the condition and the prediction`
  $b(i), b(j) \leftarrow \emptyset$; `// Blacklists`
  foreach t_i *in T(i)* **do**
    foreach t_j *in T(j)* **do**
      if $0 < t_j - t_i \leq window$ **then**
        `// New occurrence of the rule`
        Add t_i to $O_c(i \Rightarrow j)$;
        Add t_j to $O_p(i \Rightarrow j)$;
        if $t_i \notin b(i)$ *and* $t_j \notin b(j)$ **then**
          `// New distinct occurrence`
          $sup(i \Rightarrow j)$ += 1;
          $b(i) \leftarrow b(i) \cup \{t_i\}$;
          $b(j) \leftarrow b(j) \cup \{t_j\}$;
  `// Growth of basic rules`
  if $sup(i \Rightarrow j) \geq min_{sup}$ **then**
    if *netconf*$(\frac{|T(i)|}{|TS|}, \frac{|T(j)|}{|TS|}, \frac{sup(i \Rightarrow j)}{|TS|}) \geq min_{int}$ **then**
      Output rule;
    Run ExpandCondition and ExpandPrediction on the rule $i \Rightarrow j$;

5.4.5.2 Expanding the rules

ExpandCondition (algorithm 6) tries to expand a rule by adding an atom to its condition. It goes through all the possible occurrences of the rule, from the earliest to the most recent. To respect the time constraint imposed by $window$, the condition of a rule can only expand between two timestamps, noted $start$ and end, as seen in the figure 5.6. As for ExpandCondition, ExpandPrediction searches for new atoms for A_p in the area described in figure 5.7. After having found new rules, ExpandCondition and ExpandPrediction compute their frequency and their interest, and output them if they are frequent and reliable. Finally, they try

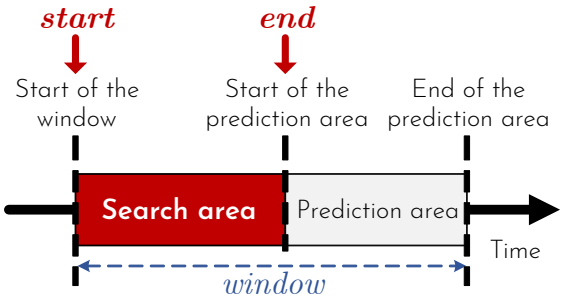


Figure 5.6: ExpandCondition Search Area

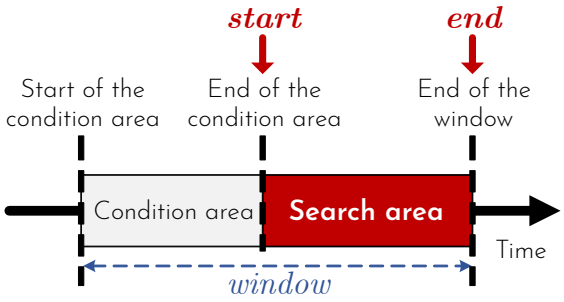


Figure 5.7: ExpandPrediction Search Area

to grow those rules again if they are frequent. Here, the simplified pseudocodes of TSRuleGrowth and ExpandPrediction are described. In appendix B, more detailed pseudocodes of ExpandCondition (algorithm 11) and ExpandPrediction (algorithm 12) are presented.

Algorithm 6: ExpandPrediction

Data: TS: time series, $A_c \Rightarrow A_p$: rule, $sup(A_c)$, occurrences of $A_c \Rightarrow A_p$, min_{sup}: minimum support, min_{int}: minimum interest, $window$: duration

`// Growth of the original rule` $A_c \Rightarrow A_p$

1 **for** *each occurrence of the rule* $A_c \Rightarrow A_p$ **do**
2 **foreach** *atom* k *seen in the search area, between* $start$ *and* end **do**
3 **if** k *has never been seen before, i.e.* $A_c \Rightarrow A_{pk}$ *does not exist* **then**
4 Create a new rule $A_c \Rightarrow A_{pk}$, its lists of occurrences and its blacklists;
5 $sup(A_c \Rightarrow A_{pk}) \leftarrow 0$;
6 **foreach** *timestamp of* k t_k *inside the window (ascending order)* **do**
7 **if** $k > \max(e), e \in A_p$ *or* $t_k >$ *all occurrences of* k *in the prediction part of the rule* **then**
8 Create a new occurrence of $A_c \Rightarrow A_{pk}$;
9 **if** *timestamps are not in the blacklists* **then**
10 $sup(A_c \Rightarrow A_{pk})$ += 1;
11 Add the timestamps to the blacklists;

`// Growth of the new rules found`

12 **foreach** *item* k *where* $sup(A_c \Rightarrow A_{pk}) \geq min_{sup}$ **do**
13 $sup(A_{pk}) \leftarrow Count(A_{pk}, TS, window)$;
14 **if** $netconf(\frac{sup(A_c)}{|TS|}, \frac{sup(A_{pk})}{|TS|}, \frac{sup(A_c \Rightarrow A_{pk})}{|TS|}) \geq min_{int}$ **then**
15 Output rule;
16 Run ExpandCondition and ExpandPrediction;

5.4.5.3 Multiprocessing

TSRuleGrowth's architecture allows multiprocessing to be implemented. In the main algorithm, TSRuleGrowth takes pairs of atoms to try to find primary rules, and then extend them. Here, multiprocessing is implemented as follows: when TSRuleGrowth is started, several parallel processes are created, up to one per logical core, and an ordered list of all possible atom pairs is created. Each process has a copy of the time series and of the list of atom pairs, and all processes share a common iterator, called $iter$. When a process is started, it does the following loop: it locks $iter$, takes the couple of atoms defined by $iter$, increments $iter$, and

Table 5.1: Database characteristics, and parameters applied to TSRuleGrowth

	ContextAct@A4H	**Orange4Home**
Recording period	7 days in July and 21 days in November	4 consecutive weeks
Number of connected objects	213	222
Data records	35634	746745
Pre-processing parameters, outlined in section 4.3		
δ_t	24 hours	24 hours
θ_{seg}	0.1	0.1
θ_{clu}	Defined by the maximum silhouette	
TSRuleGrowth parameters, outlined in section 5.4.1		
min_{sup}	7	20
min_{int}	0.9	0.9
$window$ (in seconds)	1,2,5,20,40,60,80, 100,120,140,160,180	1,2,5,10,15,20,25,30

unlocks $iter$. Finally, it searches if this pair of atoms can lead to rules. Once completed, it loops back to search on other pairs of atoms. This allows for dynamic resource management, and processes interfere with each other as little as possible, as they share only one variable.

5.5 Experiments and results

In this section, we detail experiments with the implementation of the AmI system. The experiments focused on the two sets of algorithms detailed in this chapter: pre-processing and rule research. At the time of these experiments, time indicators had not been added. In addition, the concept of interfaces had not been created, and interfaces were part of the actuators at the time. It should be noted that the discretization of quantitative data has run alongside with the rule search, but the results will mainly focus on TSRuleGrowth. Further developments regarding pre-processing, more precisely discretization, and the rest of the implementation will be expressed in the next chapter.

5.5.1 Results of TSRuleGrowth on two databases

We tested this algorithm on two databases: ContextAct@A4H [Lago et al., 2017] and Orange4Home [Cumin et al., 2017a]. Both databases contain daily activities of a single occupant. The characteristics of these databases and the parameters applied to TSRuleGrowth are described in table 5.1. The ContextAct@A4H database is located on the same physical location as Orange4Home. But it has significant differences: the objects, as well as their names, are not the same. In addition, the observed person is different, as is the observation period. Thus, the observed habits are different from one database to another.

For the purpose of the experiment, some objects were specified manually to be actuators: shutters, doors, and lights for example. In addition, the pre-processing part described in chapter 4 was used for these experiments. Also, the timestamps have been rounded to the nearest second on both databases. TSRuleGrowth has been implemented in Python with

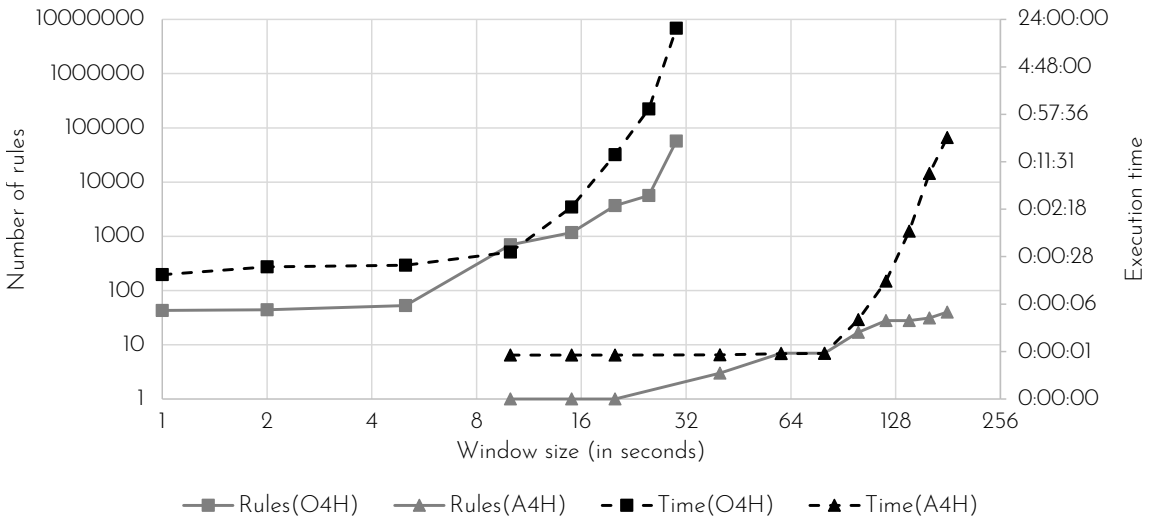


Figure 5.8: Number of rules and execution time, TSRuleGrowth on Orange4Home (O4H) and ContextAct@A4H (A4H)

multiprocessing[1].

5.5.1.1 Cross-cutting results

First, let us look at TSRuleGrowth's results on the two databases. In general, two aspects of the algorithm are evident in figure 5.8.

First, the execution time increases exponentially as a function of the window size. Indeed, when the window is large, so is the search space; it takes into account more atoms and results in more possible combinations for rule creation.

Secondly, the number of rules found is also increasing exponentially. For example, on the Orange4Home database, 43 rules are found on a one-second window, and 57103 on a 30-second window. This is explained by two complementary reasons:

- In a connected environment, several connected objects can be used to characterize a situation. For example, a person's entry into his or her home can be observed by a presence, noise, or door opening sensor. Thus, the rules found can be formed from a combination of atoms of these three objects. The larger a window is, the more combinations are possible, thus increasing the number of found rules and the computation time.

- Also, the rules found on a given window will, for the most part, be found again on larger windows.

It is mentioned "for the most part" in the previous sentence, as some rules can be invalidated from one window to another. The invalidation of these rules does not come from the support of the rule, which can only increase from one window to another, but rather from its interest. Indeed, the interest of a rule is calculated according to the support of the rule, and the support of its components. In some cases, the support of the components may increase more than that of the rule, reducing the interest enough to invalidate the rule. Let us take the example of the rule {'bathroom light1:on, bathroom switch top left:off'} ⇒ {'bathroom door:closed'}:

[1]CPU: Intel(R) Xeon(R) Gold 5118 @ 2.30GHz, RAM: 128GiB, Ubuntu 18.04.2 LTS

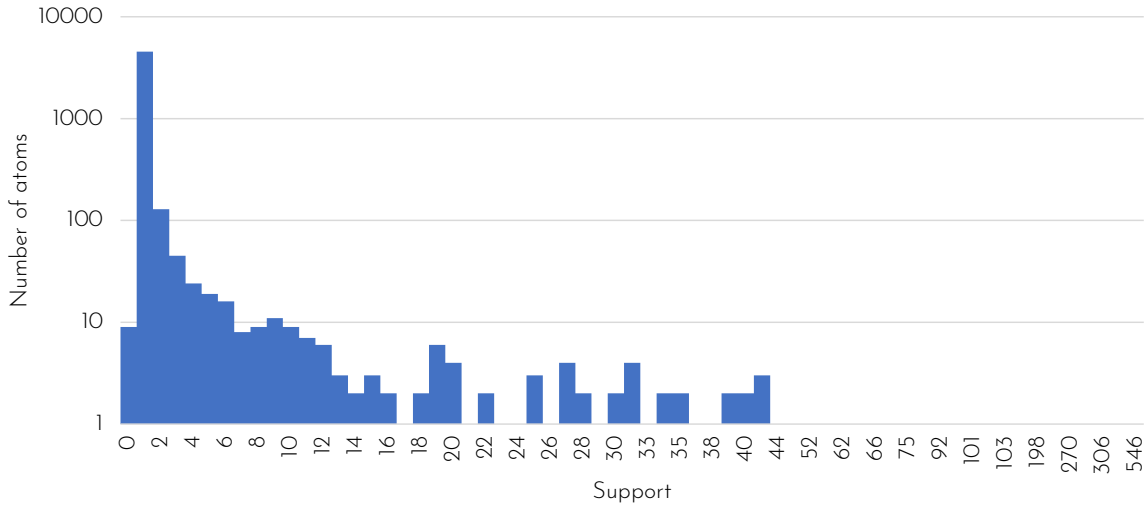


Figure 5.9: Histogram of the atoms grouped by their support in ContextAct@A4H

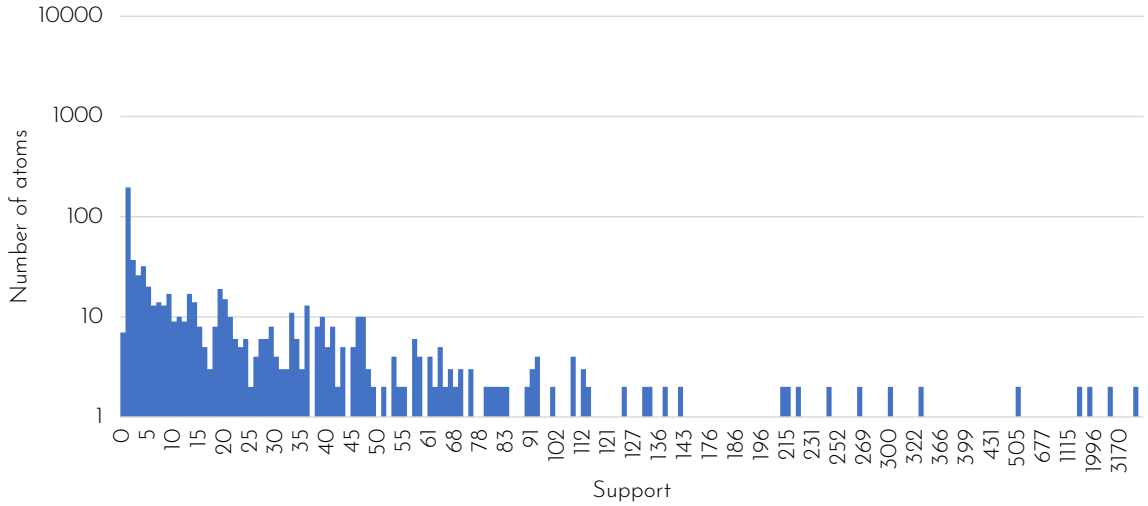


Figure 5.10: Histogram of the atoms grouped by their support in Orange4Home

- With a 5 second window, this rule is validated. The support of the rule is 38, the support of the condition is 91 and the support of the prediction is 42. Its interest is therefore 0,904. Using a 10 second window, the support is still at 38 for the rule and 91 for the prediction, but changes to 44 for the condition. This is typically the case explained above, the support of the rule increases less than that of its components. As a result, its interest drops to 0.863, invalidating the rule for this window.

- These invalidated rules represent only a fraction of the total number of rules found. Indeed, if we compare the results obtained with 10 and 15 second window lengths, 3 rules were invalidated, while 694 rules were found on the window of 10 seconds, and 1170 rules were found on the window of 15 seconds.

5.5.1.2 Difference in results between the two databases

In figure 5.8, we have explained the overall result curve of TSRuleGrowth. However, there is a very clear difference in results between Orange4Home and ContextAct@A4H. Why is that so? After all, the physical environment and most of the connected objects are the same

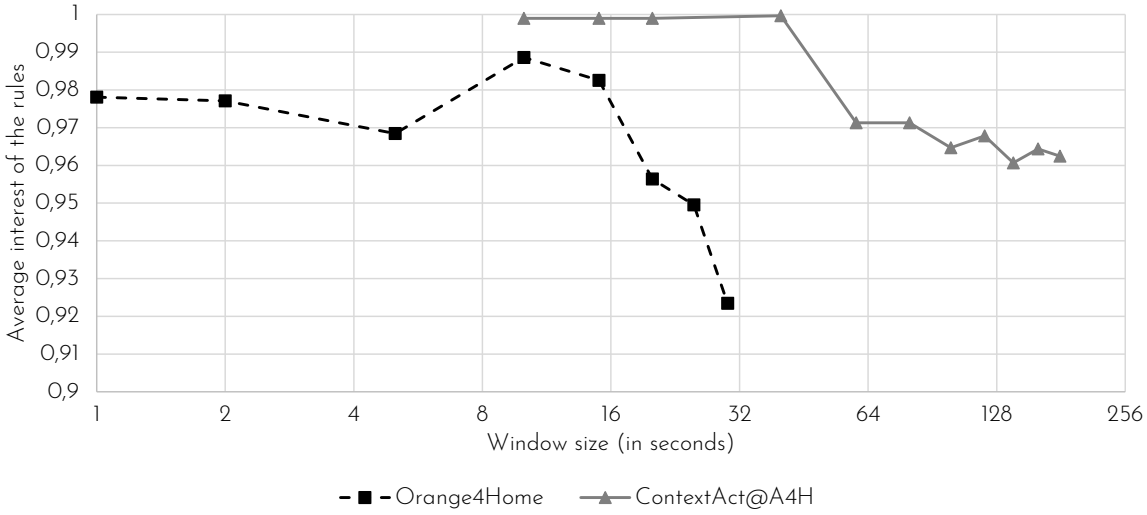


Figure 5.11: Evolution of the average interest of the rules found by TSRuleGrowth, according to the size of the window

between these two databases. The explanation is based on two factors. First, there is much less input data in ContextAct@A4H than Orange4Home (35634 vs 746745). The less input data there is, the lower the probability of finding rules. Second, of all the atoms coming from ContextAct@A4H, very few are frequent. It can be observed by comparing figure 5.9 to figure 5.10. Indeed, on this database, there are 63 atoms with an absolute support larger or equal to 20, unlike 395 on Orange4Home. That's why in our experiments, we lowered the minimum support to 7 on ContextAct@A4H, to have enough frequent atoms (here, 132). This is reflected in the results reported by TSRuleGrowth: even with a minimum support lowered to 7, far fewer rules are found on ContextAct@A4H than on Orange4Home (1 vs 3689 for a 20 second window), and the execution time is also much lower (1 second vs 14 minutes for a 20 second window).

5.5.1.3 Focus on the discovered prediction rules

Let us now look at the rules themselves, first from the Orange4Home database. On small windows (less than 5 seconds), straightforward rules are found, mostly the actions of switches in the environment. For example, {'bedroom switch bottom left:on'} ⇒ {'bedroom shutter1:closed', {'bedroom shutter2: closed'} and {'bedroom switch top right:on'} ⇒ {'bedroom light1:off', {' bedroom light2:off'}, seen in a 1 second window, indicate the different functions of the connected switches of the room. Then, by increasing the window size, more complex rules are observed, characterizing the user's usual situations. {'office door:open','office presence:on','office switch left:on'} ⇒ {'office door:closed'}, seen in a 30 second window, indicates the user's entry into his office. It should be noted that this rule takes into account several different connected objects. For the ContextAct@A4H database, many fewer rules are found in general, but some interesting rules are emerging. For example, {'fridge door:open'} ⇒ {'fridge door:closed'} describes that the fridge door will be closed within 40 seconds of being opened. As users, this rule may seem trivial to us. However, it should be remembered that the system has no preconceptions about the objects to which it is connected. With TSRuleGrowth, the system learns the rules that govern the environment, and the habits of users.

This corresponds well to the results found in figure 5.11, where we observe that the average interest rate of the rules decreases, as the observation window increases: initially, we observe

straightforward rules, with a very high measure of interest, because it only involves the action of one object on another.

Furthermore, these results highlighted the fact that some objects were intended to act on other objects. Observing these results led us to design the notion of interfaces, a new category different from actuators and sensors, for this type of object. The system finds rules involving these interfaces, and others involving the objects controlled by these interfaces. These rules are basically duplicates. In addition, the system also finds rules describing the actions of the interfaces on the objects they control. Although these rules may be interesting, they are not relevant in the case of use, i.e. finding automation to do. Indeed, these rules are already effective automations. This is why the notion of interface has been created, and why objects of this category are no longer taken into account.

Then, rules involving the user are observed, making interest lower, because these habits are less certain than the action of a switch on a light for example.

For Orange4Home as for ContextAct@A4H, TSRuleGrowth is able to discover relevant prediction rules, in the desired format, describing the actions of switches in the room, which was not detailed in the database documentation, and user habits. TSRuleGrowth can therefore predict actions in the environment, and the rules found can therefore be proposed to users. Let us now compare these results with those of TRuleGrowth.

5.5.2 Comparison between TRuleGrowth and TSRuleGrowth

Let us compare TSRuleGrowth and TRuleGrowth. To do this, we use the same input databases, Orange4Home and ContextAct@A4H. These data have been converted into transactions through the process detailed in section 5.3.4. Three sets of transactions were made, with a Δ_{tr} duration of one minute, one hour and one day. As a reminder, Δ_{tr} sets the duration for splitting the time series into transactions, introduced in section 5.3.4. The same parameters were applied between TRuleGrowth and TSRuleGrowth for min_{sup}, min_{int}, and $window$ sizes. TRuleGrowth uses netconf as a measure of interest, but no other changes are made to this algorithm: the window used is still a consecutive number of itemsets, instead of a duration for TSRuleGrowth.

As a reminder, for TRuleGrowth, this window is defined as a fixed number of consecutive itemsets, regardless of the time difference between these itemsets. On the other hand, for TSRuleGrowth, the window is defined by a duration, regardless of the number of itemsets that may be in that window. This difference implies that for TRuleGrowth, it is possible to find rules which duration can go up to Δ_{tr}. This explains why TRuleGrowth can find more rules than TSRuleGrowth in some cases. For example, on Orange4Home, for a window of 25 itemsets/seconds, and with a Δ_{tr} of 1 hour, TRuleGrowth finds 267007 rules, and TSRuleGrowth only 5677.

Figure 5.12 shows that, like TSRuleGrowth, TRuleGrowth finds more rules exponentially as the window expands. However, this figure also clearly shows the impact that the size of Δ_{tr} has on the number of rules found. On Orange4Home, and for a window of 25 consecutive itemsets, TRuleGrowth finds 8028 rules if Δ_{tr} = 1 minute, 7052216 rules if Δ_{tr} = 1 hour, and 9851 rules if Δ_{tr} = 1 day. These results can be interpreted as follows: when Δ_{tr}=1 minute, the number of rules is limited by the short duration of the transactions. When Δ_{tr} =1 day, less transactions are created from the time series, because the time series is split into transactions with a longer duration. The number of transactions being smaller, the absolute support of the rules found is therefore also lower. As min_{sup} does not change, fewer rules are found. Figure 5.12 shows that the number of rules made with Δ_{tr} =1 day catches up with that of

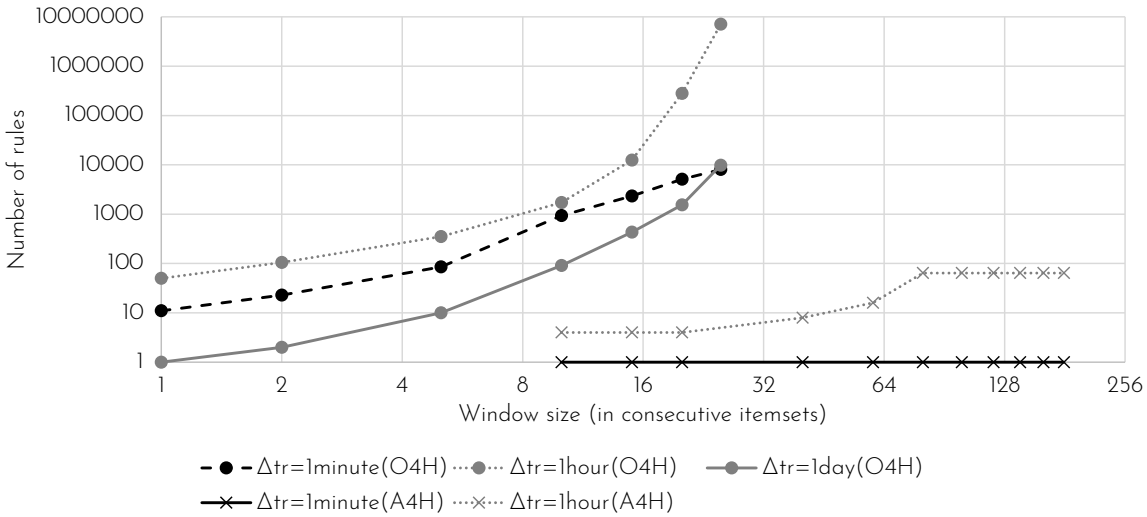


Figure 5.12: Number of rules for TRuleGrowth on Orange4Home (O4H) and ContextAct@A4H (A4H)

Δ_{tr} =1 minute as the window grows, until it exceeds it when window = 25.

It should be noted that many identical rules are found by both TRuleGrowth and TSRuleGrowth. For a 1 second window/itemset, and a 1-hour Δ_{tr}, 42 rules are found by these two algorithms. This represents 84% of the rules found by TRuleGrowth and 98% of TSRuleGrowth's rules. For the same Δ_{tr}, and a window of 10 itemsets/15 seconds, 1000 rules are common, i.e. 73% of the TRuleGrowth rules, and 85% of the TSRuleGrowth rules. But Δ_{tr} can also limit the number of rules found by TRuleGrowth and TSRuleGrowth. Of all the possible window combinations, only 194 common rules are found at most for Δ_{tr} = 1 day, 2061 for Δ_{tr} = 1 minute, and 8806 for Δ_{tr} = 1 hour.

Why does Δ_{tr} influence these results so much? We explain this by the principles cited in section 5.3.4. The rules found with Δ_{tr} = 1 minute are limited by the size of the transactions, while those found with Δ_{tr} = 1 day are limited by their minimal support. In addition, some rules can be validated by mistake. This is even more visible when Δ_{tr} is large, judging by the small number of common rules for any window. For example, the rule {'staircase switch left:on'} $\Rightarrow$ {'walkway light:off'} is seen with Δ_{tr} of 1 hour and 1 day, but not on Δ_{tr} = 1 minute. Also, it has not been found by TSRuleGrowth. Instead, the rule {'walkway switch2 top right:on'} $\Rightarrow$ {'walkway light:off'}, found by TSRuleGrowth, is more coherent, because the two involved objects are in the same room. Δ_{tr}'s limitations are even more visible on the ContextAct@A4H database. With min_{sup} = 7, and a Δ_{tr} of 1 hour or 1 day, TRuleGrowth does not find a rule, for any window. If Δ_{tr} = 1 minute, TRuleGrowth finds a single rule, which is also found by TSRuleGrowth. Thus, in the case of ContextAct@A4H, TRuleGrowth finds much fewer rules than TSRuleGrowth for the same minimum support.

By lowering min_{sup} to 6, TRuleGrowth finds more rules. If Δ_{tr} = 1 day, no rule found, if Δ_{tr} = 1 minute, only 1 rule found, also found by TSRuleGrowth. This number can be up to 64 if Δ_{tr} = 1 hour. It is higher than TSRuleGrowth can find (maximum 40), but few rules are common to both TRuleGrowth and TSRuleGrowth (maximum 16). This is explained by the difference in the window concept between TRuleGrowth and TSRuleGrowth, giving unique rules to TRuleGrowth, and the decrease in absolute support caused by the size of transactions, giving unique rules to TSRuleGrowth.

We can therefore confirm that converting the time series into transactions can severely limit the creation of rules and can create rules validated by mistake. TSRuleGrowth, taking

directly into account a time series, overcomes those shortcomings.

5.6 Conclusion

This chapter highlights the main contributions of the thesis, namely a prediction rule search algorithm called TSRuleGrowth and a notion of support on time series. The notion of support is freed from the limitations expressed in the state of the art. The algorithm also distinguishes itself by its features: first, an incremental architecture, inspired by TRuleGrowth, allowing to limit the search to certain atoms if necessary, as in the proposed use case; secondly, a sliding window, allowing to limit the duration of the searched rules; finally, the use of multisets in the rule structure, instead of sets in TRuleGrowth. Experiments carried out on two real databases validate the global approach, and show that TSRuleGrowth provides several advantages over state-of-the-art algorithms, including the possibility of limiting the search to several levels, either on the length of the rules, or on the atoms that compose them. These possibilities of limitation allow to avoid excessive calculation times.

This algorithm, as well as those detailed in the previous chapter, form the main foundation of the targeted AmI system, while allowing its evolution, through the implementation of complementary components, such as feedback, or the application of rules.

However, this AmI system has many points of evolution, particularly concerning the consideration of user feedback or the intelligibility of the system, more precisely that of automation proposals. The following chapter highlights the possibilities for further development of the system presented here, to get an idea of a complete, local, and useful AmI system for users.

Chapter 6

Evolutions

6.1 Introduction

In this thesis, we have described the architecture and some algorithms of a system that provides automation in a connected environment. This is a first step in the field of AmI, an area that will continue and evolve in the coming decades. In this chapter, we will therefore focus on the possible improvements of the proposed system. We have already detailed the possible major approaches of AmI in section 2.2.1, namely first leaving control to users, then interacting with them like a butler, and finally acting in the background, with few interactions.

In this chapter, we detail some ideas that have been imagined and sometimes tested in this thesis. First, we review the test databases and their shortcomings, and propose improvements to existing databases and new database designs. Then, we suggest improvement and alternatives concerning the two parts of the presented AmI system: pre-processing and rule mining. We focus on the interface that this AmI system can have, through the display of prediction rules, and user interactions. Finally, we discuss the feedback part, which is closely related to the user interactions.

6.2 Databases for AmI activity discovery

To understand the evolutions to be provided in the databases used to test activity discovery systems in connected environments, let us analyze those used to evaluate our system: Orange4Home [Cumin et al., 2017a] and ContextAct@A4H [Lago et al., 2017]. It is first of all very important to note that these databases were mainly made to test predefined activity recognition algorithms, presented in section 1.4.3.2, which falls within the scope of supervised learning. Those databases were not initially made for our use case, i.e. the discovery of new activities, an unsupervised learning domain described in section 1.4.3.4. Since these databases were made for a different purpose, it is normal to observe shortcomings and problems if we use these databases for our use case.

First, let us look at the relevant aspects of these databases, and then analyze their shortcomings.

6.2.1 Relevant aspects of existing databases

First of all, these databases have certain qualities that are advantageous to our use case. Indeed, the environment in which these databases were made is a **real and functional apartment**, with water, electricity, and appliances found in most apartments. **The data are real**, not simulated, and come from real connected objects monitoring real human beings. This distinction is crucial for the validation of an AmI system. Indeed, one specificity of AmI is to work in real physical environments, and testing an AmI system on synthetic data would mean missing this specificity. Also, simulated data may be biased in their creation, or may not faithfully represent all the characteristics of the real data. **The exact sending time of events** coming from connected objects **are recorded** in the database, which is perfect for time series rule mining. The provided documentation makes it easy to find which object returns categorical events, and which object returns quantitative events.

But the most interesting aspects of these bases are **the number of objects and their diversity**. Here, more than 200 objects are present in the environment. However, these objects are not very varied in their functions. To illustrate, we have grouped all the objects of the Orange4Home database according to their functionality, which we deduced from their names. The mapping table can be found in appendix C, table C.1. As shown in figure 6.1, the objects are grouped into 21 categories, 3 of which represent half of the connected objects in the base: many electrical sensors, switches, and also heaters, present in all the rooms and with multiple properties to be adjusted, like the temperature. On top of this, there are many objects that are identical between rooms, such as presence sensors, door opening sensors and water flow sensors.

The number and diversity of these objects is debatable. Regarding the number, we can judge that it is unrealistic to imagine 200 connected objects in the same environment. However, we can also say that there is no need to take them all into account for testing purposes. Moreover, this profusion of data allows a lot of experiments to be done. Finally, it is possible that this number will be reached in the environments of the years to come.

Finally, we can ask the question of the diversity of the objects contained in the database. Indeed, some users will buy objects of various kinds, because a connected object is interesting for them only if it is useful. Some objects can be highlighted by their originality combined with their functionality. Parrot's Flower Power, for example, was unique when it was released. It was one of the first connected objects in the garden to monitor a plant's water needs. Even if such basic objects as door opening sensors will be present, we can imagine more diversity in the

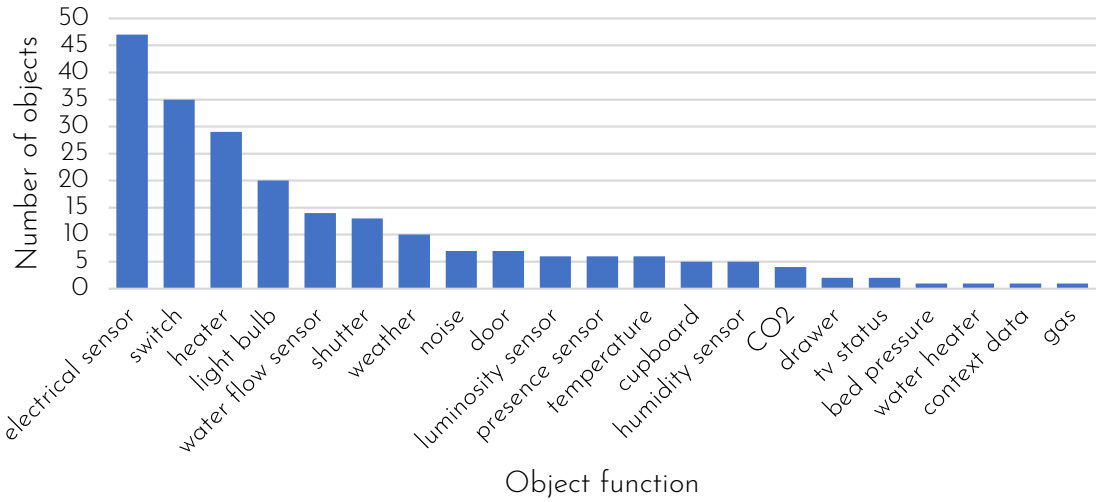


Figure 6.1: Diversity of the objects from the Orange4Home database [Cumin et al., 2017a]

objects than there is in the database. However, this diversity of objects in future environments is not yet certain. Indeed, we can imagine environments connected with basic objects, such as light bulbs, or door opening sensors, without more diversity. We can also think about the construction of new buildings, containing connected objects monitoring it, such as electrical sensors or water meters, without more diversity. Moreover, this lack of diversity also makes it easier to characterize the data, and to interpret the results.

There is therefore no right or wrong solution regarding the number of objects to be taken into account and their diversity. However, the precise purpose of the AmI system to be tested can help in making a choice. In our case for example, we are looking to monitor and act on objects in an environment to serve users in a personalized way. Thus, the environment on which to test the system should be as close as possible to future personal environments, such as connected homes. It would therefore be interesting to have a database containing fewer objects, but more diversified in nature, with potentially **more actuators**. Indeed, having more actuators means being able to act in more ways on the environment. So, with more actuators, it would be interesting to see how people use them, and therefore how the AmI system can automate them.

6.2.2 Shortcomings of existing databases

Orange4Home and ContextAct@A4H have shortcomings that need to be addressed in future databases.

First, with these databases, it was very difficult to interpret the prediction rules found by the algorithm. Generally speaking, we could only rely on the names given to the connected objects and the data sent to interpret a result. We could also rely on the name of the activity defined by hand in the schedule. In general, we would have liked to have **more descriptive data** about the environment, the objects, and why not about the users' actions. Thus, the data of the connected objects would be the input of the system, and the descriptive data would be used to interpret the results, and would not be processed by the AmI system. There is not as much descriptive data as we would need because our goal, which is activity discovery, is not the same as the one targeted by the database, which is the recognition of pre-established activities.

Indeed, the purpose of these databases is to recognize pre-established activities, which is supervised learning. Thus, to test a rule recognition algorithm, it is sufficient to check whether

the algorithm recognizes the correct activity or not. But we want to discover new activities, and we are not in a case of supervised learning. In our situation, we have to interpret the prediction rules returned by the algorithm. Thus, any descriptive data is important for this interpretation. What is particularly missing in these databases:

- The value range is not defined for connected objects, when the returned data is quantitative.
- The function of the connected objects is also not clearly defined. For example, a connected heating system exists in the apartment. It has several sensors and actuators, but they are not documented. There is also no descriptive data showing the dependency relationships between objects: e.g. switches associated with lights or shutters. Relying solely on the names of these sensors and actuators greatly limits the interpretation of results.
- The location of the objects is also unclear. The plans provided do not indicate where the connected objects are placed. Only the room can be deduced through the names given to these objects, but nothing more. Having a precise location of the objects would make it easier to interpret the found rules, and, for example, to trace the path a user has taken during a rule.
- Habits are documented in the same way, limited to a name, start times, and end times.

Also, the two databases both reflect the actions of **only one person**. To have a database that is representative of what is happening in a real environment, it would be necessary to be able to observe several people. If several people are in the same environment, it will complicate the search for rules, because these people do not necessarily have the same habits. Thus, it will be necessary to find out what data can distinguish one person from another, in order to make predictions. It would be interesting to see how rule finding would be able to find this data in this context.

Another problem encountered is the fact that **the observed period is relatively short.** For example, in the Orange4Home database, this is four consecutive weeks. This covers only a small part of the year, and does not include, for example, the change of seasons. For ContextAct@A4H, the observed period is split in two, in June and November. But here again, the total observed duration is shorter than Orange4Home.

In the Orange4Home database, the observed person is only present during daytime, not at night. This limits the number of possible rules to find.

Observing over a long period of time allows for habits changes, caused either by external events, such as changing seasons, or by the simple will of users. In addition, with a longer observation time, it is also possible to observe changes in connected objects, such as failures, additions or deletions of objects. As mentioned in the previous chapters, the system must adapt to these changes, so future databases will have to take this into account.

In these two databases, few activities were to be recognized. More specifically, Orange4Home has less than 20 activities, and ContextAct@A4H has only 8, according to their respective documentation, such as taking a shower, sleeping, eating, working, watching television, or going up or down stairs. Thus, **the actions undertaken were repetitive**, in order to respect the decided schedule. The schedule can also be the same from one day to the next, which is also why so many rules can be found in section 5.5. Take as an example the planning of the first week of the Orange4Home database in figure 6.2. It clearly shows the monotony between the activities between the days. We could say that this is ideal for our case, because

the habits are clearly defined. Only, with a schedule like that, where the days are homogeneous, everything becomes a habit. Not only does it remove realism from what happens in everyday life, with its diversity of actions, its share of unexpected events and its potentially sudden changes, but a prediction rule search algorithm can find a lot of rules, because of this pre-established planning.

In addition, **the suppliers of connected objects are not varied**, only one supplier per object type. This does not allow, for example, to compare the consideration of several objects with similar functions, but from different suppliers. For example, two temperature sensors can measure the temperature differently, and thus return dissimilar data. Assessing the impact of these differences would be important for the validation of a global AmI system.

Finally, **some, but very few, objects return data already processed by an algorithm.** These include the noise sensors in the Orange4Home database, which are microphone data processed by an algorithm that is not documented, to have a noise level that is not interpretable, making the object useless in our case. This may be due to confidentiality issues, but we have not had confirmation of this. Although some objects may have this type of behavior, it would be useful to document in detail the algorithms used, or at least the data returned even if the overall concept is understood, in order to be able to interpret the results.

6.2.3 Summary

Future databases must keep the interesting aspects of Orange4Home and ContextAct@A4H: the fact of having real data, coming from a functional physical environment populated with connected objects, and whose exact sending times are recorded.

The future databases must also avoid the shortcomings mentioned above, to serve as a master standard for verifying an activity discovery algorithm in an AmI context. A much more complete documentation will have to be provided, as well as more metadata. Needless to say, it will be impossible to make databases covering all the environments that can be analyzed. However, we can imagine databases of various environments, such as a house in the country, an apartment in the city, or a work environment. It would be interesting to see how the same AmI system adapts to all these different environments.

In addition, the environment must contain a variety of objects, from several suppliers, that are representative of the ones used today in connected houses, or what will be found in a few years' time. Activities in the environment must be documented, but above all they must be varied and avoid following a pre-determined schedule. The goal is to observe moments of life for long periods of time, day and night, interpret the results to validate or not an AmI system, and to see what is at stake in everyday life. Speaking of situations, they should also involve several users, even if only at certain times.

For an AmI system to be validated, it must be tested on real-life situations. In addition, it would be interesting to have real-time experiments to observe and understand the interactions that users have with the AmI system. However, building a database for activity discovery will also raise major issues in terms of privacy. Indeed, on the one hand, it would be necessary to have data from objects present in a real physical environment, and observing everyday life situations. For this, it is necessary to avoid cognitive biases in the subjects, as they are focused on the fact that they are in a test environment. But on the other hand, it is imperative to respect the privacy of these people. This issue will have to be addressed for future databases.

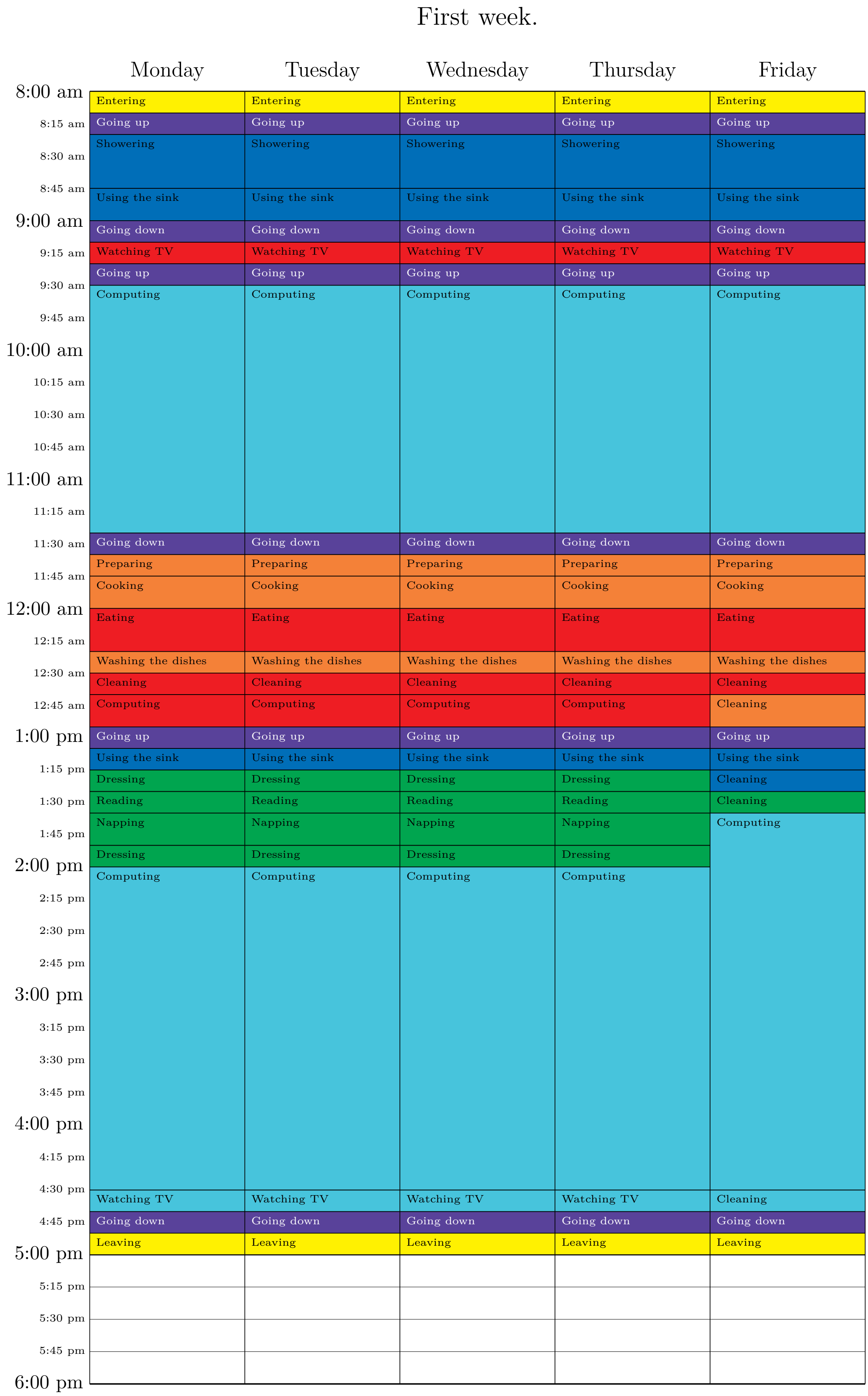


Figure 6.2: Planning of the first week of the Orange4Home database, excerpt from the Orange4Home documentation [Cumin et al., 2017a]

6.3 Pre-processing

In the implementation presented in chapter 4, pre-processing cleans the repetitions for categorical events, and discretizes quantitative events according to variations. Further development is possible.

6.3.1 Categorical events

First, regarding categorical events, it is possible to consider that some objects return notifications of state changes, instead of just returning the state. For example, it is easy to imagine a presence sensor that returns a notification only when someone is in a room, and nothing else. This was mentioned in section 2.3.2.2. It would thus be possible to take them into account, by not cleaning duplicates. On the other hand, it seems difficult to distinguish between objects returning changes and those returning states, without prior information. The objects themselves could provide this information.

It is also possible to take into account the stationary states of the connected objects, thus giving context information to the rules to be found. This could be done in the same way as adding time indicators, so that periodic rules can be found, as explained in section 3.2.2: stationary states would exist in the form of atoms, and would be added to every itemset of the time series. The danger of this method is to have much more data to process on the time series, as we discussed in section 2.3.2.2.

6.3.2 Quantitative events

For quantitative events, several ways of improvement are possible. First of all, it is possible, for some sensors, to make a discretization according to ranges of values, explained in section 2.3.2.3. This can be useful for temperature sensors, where only the “cold”, “comfy” and “warm” events could be defined. This requires expert data, but it would greatly simplify discretization, and improve the readability of the results, compared to a discretization taking into account variations. Also, signal processing algorithms could support the algorithms presented in this thesis. They could be applied as is, if the observed data have a fixed sampling rate. If this is not the case, resampling would become necessary. This could be useful when the original discretization algorithms have been put in fault, as in section 4.3.1.4. One could even imagine connecting dedicated systems, between the object data and the AmI system, to handle the discretization process and provide categorical events. These systems would be tailor-made for specific objects, and would work for a specific type of data. For example, a speech recognition algorithm for a microphone, an image recognition algorithm for a camera, or an algorithm that estimates the number of people in a room based on CO_2 levels. However, using dedicated systems may seem contradictory to the nature of the AmI system proposed in this thesis, which is intended to be learning by itself.

It would also be interesting to implement a self-evaluation of the pre-processing, for its optimization. Also, as pre-processing has parameters that differ depending on the object, this optimization would be done object by object. For that, it would be necessary to define performance indicators. These indicators should be created and evaluated to assess their relevance. They will allow pre-processing to optimize the segmentation parameter θ_{seg} and the clustering parameter θ_{clu} for a given object, as explained in section 4.3. For example, we could evaluate the segmentation through the ratio of deleted points, as shown in section 4.3.1.4, counterbalanced by a measure of distance between the original and the simplified signal over a given

period. Other indicators can come from rule mining, such as the number of rules found for a given object. Those indicators would represent an inner feedback for this algorithm. These settings could also be optimized based on user experience, through user feedback. These are just a few ideas, as these indicators need to be precisely defined, evaluated, and implemented.

As we see, many perspectives exist regarding pre-processing, both for quantitative and categorical events. These perspectives not only improve existing algorithms, but also provide additional information to discover not only more rules, but also more significant and meaningful rules.

6.4 Rule mining

In the implementation of the AmI system, we made precise choices regarding the prediction rules to be found: partially ordered rules containing multisets. In chapter 5, we developed an algorithm named TSRuleGrowth, looking for this type of rules, but which has several parameters to define: the size of the $window$, min_{sup}, and min_{int}.

6.4.1 Estimation of the parameters

It is precisely this first point that can be improved. It is possible to have an algorithm that estimates the parameters to be applied to TSRuleGrowth by observing certain characteristics of the environment, such as the number of objects, but also certain characteristics of the time series, such as the density of atoms, or according to objectives dictated by the user. For example, min_{int}, which estimates the minimum reliability of the rules to be found, can be set by hand at the beginning of the system launch, optimized according to certain parameters such as the number of found rules and the computation time. It can also be modified by the user via simple interfaces, like a slider, depending on whether he is looking for less reliable, or more reliable rules. The same applies to min_{sup} and $window$. We can imagine that min_{sup} and $window$ can be calculated according to certain characteristics of the time series, like the average of the absolute support of the atoms over the course of a week for min_{sup}, or by trial and error, by testing several parameters in parallel. The objective of the parameters to be defined is a balance between finding the maximum number of useful rules to the user, and not having too much computation time. Additional time series analyses and user experiences will therefore be required.

6.4.2 Alternatives in the rule structure

In the proposed system, we chose to use multisets because we considered that the multiplicity of an atom in a rule could be important. However, it would also be interesting, regarding the structure of the rules to be found, to test alternatives, and to compare the results in terms of found rules and computation time. First, sets could be used instead of multisets, which removes the multiplicity of elements within rules. This would be easy to implement, would make the TSRuleGrowth algorithm faster too, and lighter in memory, because the occurrence recording, explained in section 5.4.3, would use lists instead of associative arrays. Secondly, it would be interesting to look for rules containing sequences, i.e. ordered lists of atoms. For example, “If I open the door and then I am in the entrance”, the order of which indicates that the person has entered, not left, his or her home, “then turn on the light in the entrance and then the light in the living room”, which directly lights up the room the user is currently in,

and then lights up the room he or she is used to going to afterwards. One could imagine a rule whose condition is unordered and the prediction is a sequence. This type of rule would define actions to be taken, one after the other. For example, we could produce a rule like: "If the front door opens and the light at the entrance turns on, then run a bath and then turn on the bathroom and then play soft music". One could also imagine the opposite, i.e. a sequential condition and an unordered prediction. Or even a fully-ordered rule, defined in section 3.2.3, where condition and prediction are both sequences. All these types of rules would be interesting to observe, as some could be complementary to the partially ordered rules we have chosen. Indeed, some habits may only be defined by sequences.

Secondly, one aspect of TSRuleGrowth was not developed in the thesis, that of updating the rules over time. This is a major evolution to be made on this system: to be able to constantly update its results over time. Several ways are possible, such as simply deleting the old results and keeping only the most recent ones, or keeping the entire history of the rules to be found, to understand certain changes in habits, such as during the seasons, with the heating on and the shutters closing early in the day in winter. All this aspect updated over time is essential for our use case and must be considered.

6.4.3 Time indicators and other contextual information

In section 3.2.2, we mentioned the addition of time indicators in the time series to find periodic rules. We experimented this with TSRuleGrowth on the Orange4Home database, with the following parameters: min_{sup} = 10, min_{int} = 0.9, and $window$ = 1, 2, 5, and 10 seconds. We added indicators for the hour and the weekday to each item and time series, as explained in section 3.2.2. These indicators are treated in the same way as data from connected objects. Here are some of the obtained results, presented in a simplified syntax:

- "If the front door opens and it is 8am, then turn on first light in the entrance and turn on the light in the stairwell." This indicates the entry of the occupant into his home, between 8am and 9am which can easily be checked on the schedule shown in figure 6.2. It has been found on a 10 second $window$.

- "If the bedroom door opens and someone is present in the bedroom and it is 12am, then turn on third and fourth lights of the bedroom." This shows that the occupant is used to going into the bedroom between 12pm and 1pm, which is different from the schedule presented. However, after analysis of the data, this habit is verified. This rule has been found on a 5 second $window$.

This is very useful, and allows to have rules whose condition combines time information and data of connected objects. However, it is very difficult to find rules whose condition is composed only of time indicators. Indeed, they are repeated in each itemset of the input time series of TSRuleGrowth, and are therefore seen many more times than events from connected objects. Thus, if a rule of the type: "time indicator $\Rightarrow$ object events" is created, it will not be validated, because the occurrences of the indicator are much more frequent than those of the object events. To address this problem, new time indicators should be added, describing only temporal changes, such as changes in hours, for example. This new indicator would only be present once, and not repeatedly, in the time series.

In addition, other contextual information is possible, such as the stationary states of the objects mentioned in section 6.3. However, adding this data could make the time series heavier, lengthening the processing time of the rule mining algorithm. Experiments will be

necessary to determine the relevance of the rules found with this contextual information, and the impact this addition has on computation time.

6.4.4 Computational optimization

The last part of the evolution concerning rule mining is the computational optimization of TSRuleGrowth. Indeed, to offer the AmI system to as many people as possible, it is important to optimize the different algorithms that compose it. We have seen that the rule mining algorithm could take several hours to find results in figure 5.8. Of course, this execution time should be put into perspective with the large amount of data that was processed in the experiment. Nevertheless, optimizing TSRuleGrowth will not only allow to get results quickly, but also to offer the AmI system on less powerful machines, and thus to propose it to more people.

Let us start with one of the features of TSRuleGrowth, which is to test all the possibilities to extend rules. The aim here is to specify the search for rules based on the previously found ones, through an exploration and exploitation process. It is possible, for example, to make a map of the objects. The distance between two objects would be calculated according to the rules previously found involving these two objects. On the exploitation part, the search for rules would be limited according to the distance between objects. On the exploration part, to avoid too much specialization on data, the system could randomly grant the search for rules on objects with a large distance. There could also be an evaluation of this optimization by comparing the results obtained with and without this optimization. The evaluation criteria would still need to be defined. This optimization, aided by self-evaluation, would allow the algorithm to specialize on the environmental data in order to find rules more quickly, while remaining attentive to changes in habits and in the environment that may occur over time.

In parallel, this research could also be limited depending on the notion of usefulness defined in section 6.6. Indeed, this saves TSRuleGrowth from spending time searching rules that will not be wanted by the user anyway.

6.5 Display of automation proposals

In section 3.2.4, we mentioned the issue of providing automation proposals in a way that is understandable to users. We have attempted a first approach to this problem by proposing a sentence describing the rule, of the type: “If... then...”. The condition part describes the state changes of the objects, and the prediction part describes the actions to be done on them, using the imperative. Thus, the main problem is to move from an internal representation produced by the rule-finding algorithm to an intelligible representation, easily understandable by users.

Two major questions therefore arise. The first one is which representation to choose? And once a representation has been chosen, how to make the transcription between the internal rules of the system and this external representation?

6.5.1 Perspectives for the representation of automation proposals

6.5.1.1 Evolution of the textual representation

Firstly, adding delay information between the condition and the prediction could greatly improve the understanding of the rules. For example, “If the door opens, then in 5 seconds, turn on the light”, or, “If the oven turns on, then, in 10 minutes, turn on the radio”. This delay

can be calculated in the rule search algorithm, and we were able to integrate it and do some experimentation in this thesis. Here, the delay is calculated directly by TSRuleGrowth. As a reminder, in this algorithm, rule occurrences are recorded, as explained in section 5.4.3. In our experiment, we calculated the delay between the end of the condition and the beginning of the prediction on all possible occurrences of the rule, then we averaged these delays to obtain the one to display to the user. This information is easy to calculate in TSRuleGrowth, and it is possible to compute other indicators with a similar method, such as a more precise time of application of the rule, as well as for the condition or prediction.

For example, the rule "If the fridge door opens, then, after 8 seconds, close the fridge door" describes a common situation typical of a kitchen, which would be useful to be automated. Indeed, if the user forgets to close it or has closed it incorrectly, the system could close this door automatically, thus preserving the cold chain for the products, and avoiding an energy waste. This notion of delay is therefore crucial for this rule, and we can hardly imagine removing it, i.e. closing the door directly after it has been opened. It would also be interesting to see the duration of application of the condition, the prediction, and why not the entire rule.

6.5.1.2 Other means of representation

As mentioned in section 3.2.4, we can imagine other ways to show automation proposals to users, for example, a graphical user interface, shown in figure 3.5. But it is also possible to integrate it into a new means of interaction that have been widespread since 2011 with the introduction of Siri for the iPhone [Gross, 2011]: the voice assistant. Here, the voice assistant, present in the environment, could dialogue with the user, to propose rules, then the user, by voice, could validate them. Thus, the control of the system would no longer be done on a screen, and we would have a beginning of personification of the AmI system with a conversational agent, as explained in section 2.2.1. Several voice assistants currently exist, and this mode of interaction becomes familiar to most users. It would be an attractive means of interaction for an AmI system, where the user could control the system anywhere in the house, as long as a connected speaker is nearby. This has two major advantages:

- With this mode of interaction, an AmI system would be an integral part of the environment, and users would interact with it in a natural way, which is speech. This is in line with Mark Weiser's original vision, who imagined computing acting as a background in the environment.

- With this means of interaction, the transition to a more advanced AmI system acting as a butler, described in section 2.2.1, would be easier. Intelligence could be embodied, have a single voice, and interact with its users, in the same way that we speak.

6.5.2 Transcription of the rules

In parallel with the means of interaction, it is important to address the problem of translating raw rules into automation proposals that are easily understood by users. Indeed, the prediction rules found by the AmI system are very precise in their description, such as: "if fridge opens and the weather is rainy and the kitchen light is on and the oven is on then turn on the hood and turn off the television in the living room". These rules, as they stand, can be difficult to understand for the average user.

The problem is: how to transcribe these rules in a way that is understandable to users, in other words how to go from a raw rule into an easy to understand representation? This

problem is deeply tied to the one mentioned in the previous section. A workable solution could be done in two phases:

- The first phase consists in searching for common metadata affiliated to the atoms contained in a rule. For example, if we take the rule "If the television in the living room turns on, then turn off the light in the living room", it can be simplified to "In the living room, if the television turns on, then turn off the light".

- The second phase consists in introducing a higher level of semantics. For example, if we take the rule "If the front door opens, the light turns on, the door closes and it is 6pm, then turn on the oven", it is possible that the condition part is in itself a usual situation, shared by other automation proposals. Thus, through interactions with the user, it is possible to add a high level of semantics that makes sense to the user. In this case, the rule condition describes the arrival of a user at home after leaving work. The rule could therefore be simplified to "If the user comes home from work, then turn on the oven".

But one last point must be worked on and evolved regarding the display of the rules. In section 5.5, it should be noted that the TSRuleGrowth algorithm finds a huge amount of prediction rules. However, if we want to make proposals to users, we can only propose very few, less than a dozen. Thus, it is necessary to filter the rules to be displayed.

In some of our experiments, we have chosen to display only so-called closed rules. These are rules that do not have over-rules, i.e. rules with one more element in the condition or prediction, which have the same support as the original rule.

For example, the rule "If the door opens, then turn on the light" can be seen 5 times, but it has given rise to another rule, which is "If the door opens, then turn on the light and turn on the heater". This last rule is an over-rule, because it has all the elements of the previous rule, plus a new element. However, this last rule has only been seen 4 times, and there is no other overrule observed. Thus, the first rule is a closed rule.

But this is only one method of rule filtering. In our case of use, it should be possible to propose rules that the user would like to see: they should be selected and ordered according to a usefulness measure, which will have been determined on the basis of user interactions and feedback. It is precisely this notion that will be explained in the next section.

6.6 Interactions and user feedback

Mentioned in section 3.2.5, user interaction is an essential component of the AmI system. The primary interaction that the system has with its users is the proposal of automations, and the approval of these rules by these users. It is this basic interaction that allows to have an evaluation on the rules found and displayed. From these interactions come user feedback, which can be used to improve and customize the system. It is possible, for example, to imagine that the overall performance of the AmI system can be calculated as a balance between the ratio of accepted rules to the number of displayed rules, and the execution time of the rule search algorithm. This section is closely related to section 6.5, as not only user interaction, but also user feedback, are highly dependent on the way automation proposals are displayed.

A building block of the architecture, called the "Feedback Dispatcher" in section 2.5, was imagined to continuously improve the various components of the system to find the automations. In this block, user evaluation was materialized as a function, called a usefulness

function. The purpose of this function is to evaluate whether a rule can be considered useful to the user, and to what extent.

How can this function be defined? To get an idea of the parameters to be taken into account, we started to develop an experiment, where 22 people responded. In an online survey, we displayed a list of prediction rules, and asked the participants which ones they would approve, and why. A screenshot of the survey, in French, can be found in figure 6.3, and the translation of the complete survey can be found in appendix D. This experiment was done late in the thesis and was not very elaborate, so few conclusive results were drawn from it. However, a clear result was observed: 14 people clearly expressed that the main criterion for the choice of a rule is the nature of the actions present in it. For example, there were a few people who did not want doors in their house or apartment to close or open on their own. Thus, the function of objects will surely have a central place in the usefulness measure. It is also conceivable that other indicators, derived from the metadata associated with the rules, might be parameters of the usefulness function, such as the location of objects, or the time of day when the rule takes place. On the other hand, usefulness can also come from preferences that the user may express, such as saving energy. Needless to say, further experimentation with test subjects will be required to define this measure of usefulness.

We can also take into account other relationships between objects, which would depend on rules that have already been accepted. To illustrate this, let us imagine that a hotplate and a fridge have some rules in common, because we are in the context of cooking. So the link between the two objects would be strong. If a new rule is found that includes these two previous objects, it could therefore have a high measure of usefulness. In any case, this notion of usefulness must be defined through experiments involving human testers.

The building blocks of the system could be improved in this way:

- In pre-processing, the parameters to be set were few: for quantitative events, a segmentation threshold θ_{seg} and an observation duration δ_t, and no parameters for categorical events. The user feedback of this algorithm would in fact be an optimization problem, where θ_{seg} and δ_t must be chosen to get optimal feedback. We could imagine such feedback as the number of useful rules involving the pre-processed object. For the observation period, we could also imagine a reset mechanism, in order to obtain an updated representative time series. And, as we mentioned in section 6.3, several competing discretizations could run, maybe in parallel, to see which one would bring its highest number of accepted rules.

- In the rule search, it would avoid searching for rules that the user will not want, which reduces the execution time of the algorithm.

- In the presentation of the rules, an order could be established, according to this notion of usefulness. This maximizes the chances that the user can find the automation he/she wants.

We talked in section 6.5 about several kinds of interactions, including speech, to display and validate the rules. However, it is also possible to imagine other kinds of complementary interactions:

- Indirect interactions, which indicate that the user does not agree with an automation when it is applied. For example, opening a door directly after it has been closed by the system, or even preventing it from closing, shows that the user did not want the door to close in the first place.

Automatisation dans un environnement intelligent

Vous êtes, en ce moment-même, chez vous, dans un environnement intelligent. Cet environnement a détecté plusieurs habitudes que vous avez, et, à partir de celles-ci, vous propose d'automatiser certaines actions.
Par exemple, si on détecte que vous allumez la lumière après avoir ouvert une porte, on peut vous proposer que la lumière s'allume automatiquement à chaque fois que la porte s'ouvre.

La liste qui va suivre représente des habitudes observées chez vous, sous la forme de règles de prédiction.
Une règle de prédiction est de la forme "Si ..., alors ...". Il est à noter que les conditions des règles n'ont pas d'ordre.

Parmi les propositions suivantes, quelles sont celles que vous souhaiteriez automatiser ? Vous pouvez choisir autant de règles que vous voulez, et vous devrez expliquer vos choix globaux ensuite.

Merci !

* Obligatoire

A vous de choisir !

1. Quelles propositions accepteriez-vous ?

- ☐ Si la porte de la chambre s'ouvre et qu'il est 12 heures, alors fermer la porte de la chambre dans 15 secondes.
- ☐ Si la lumière du couloir s'allume et que nous sommes Jeudi, alors éteindre la lumière du couloir dans 30 secondes.
- ☐ Si le placard 1 de la cuisine s'ouvre, alors fermer le placard 1 de la cuisine dans 5 secondes.
- ☐ Si la porte de la chambre s'ouvre, alors fermer la porte de la chambre dans 15 secondes.
- ☐ Si la porte de la salle de bains s'ouvre et que quelqu'un est présent dans la salle de bains et que la lumière du couloir s'éteint, alors fermer la porte de la salle de bains dans 25 secondes.
- ☐ Si le tiroir 2 de la chambre s'ouvre, alors fermer le tiroir 2 de la chambre dans 20 secondes.
- ☐ Si la porte du bureau s'ouvre et que la lumière du bureau s'allume et que quelqu'un est présent dans le bureau, alors fermer la porte du bureau dans 25 secondes.
- ☐ Si la porte de la douche s'ouvre, alors fermer la porte de la douche dans 25 secondes.
- ☐ Si la porte d'entrée s'ouvre et que la lumière 1 de l'entrée s'éteint et que la lumière de l'escalier s'éteint, alors fermer la porte d'entrée dans 10 secondes.

Figure 6.3: Screenshot of the survey displaying automation proposals

- We can also imagine manually teaching a rule to the AmI system. Either via a dedicated interface, where the rule is described manually, or by asking the system to focus on actions taken by users during a given period. In this case, the user would repeat a series of actions, and the system would try, during this short period, to find the conditions that led to these actions. For example, a user can ask the system to focus for 5 minutes. During this period, he repeats the situation to be automated, such as opening the front door and then switching on the light, so that the system can then propose it because it only looked for a prediction rule in these 5 minutes.

- Finally, several privacy interactions are possible: asking not to operate, or not to observe rules during certain time slots of the day, or even certain whole days, not to take into account the data of certain objects, or of objects in certain locations. This can be dynamic, i.e. privacy proposals can be made in return for refusing a rule, or via a dedicated interface.

User interactions and feedback are perhaps the biggest areas for improvement in the AmI system. And this is quite normal: such a system is at the service of these users and must be customized. Many interactions are possible, and user feedback can be calculated in different ways. These aspects can thus be the subject of much work in AI, ergonomics, user interface, among others.

6.7 Conclusion

As we have seen in this chapter, there are perspectives at all levels of the AmI system: in algorithms, in interactions, in the user interface and even in test databases. As a reminder, this is a multidisciplinary subject. Thus, we can see that AmI will bring together scientists from several fields, such as ergonomists and computer scientists. In order to move forward on the subject, it would first be essential to work with ergonomists and test subjects, to study the user interface, the display of rules, and above all to design and implement the feedback.

Here is a summary of these improvement perspectives.

First, the user interface of an AmI system should be simple but should not prevent the user from having full control over the AmI system. The automation proposals must be designed to be easily understandable, while describing the original prediction rule accurately.

Regarding databases, it would be interesting to develop new ones specifically focused on the discovery of activities in an intelligent environment, still with real data, but with several observed people doing various activities, more documentation and more metadata on the objects.

Regarding the algorithms, it would be important to first study how the AmI system could evolve to update the rules over time, to make it adaptive to changes. Also, user feedback will allow for a system that is scalable and useful to the user. In addition, these exchanges with test subjects would be used to validate pre-processing approaches, either those implemented in the thesis or the alternatives presented in this chapter. Finally, an optimization of the different algorithms, and especially of the rule finding algorithm, must be done in order to be able to offer the AmI system to the largest number of people, without the need for a powerful machine.

Finally, in the longer term, it is easy to imagine that new techniques will lead to a smarter, more useful AmI system, as explained in section 2.2.1. Indeed, the purpose of AmI is to operate in the background in an environment, without putting too much strain on the user.

Thus, understanding the user's intent, and automate the environment in a useful way while minimizing interactions will be the major issue of AmI in the years to come. It will therefore require algorithms more oriented towards strong AI to address this problem.

Conclusion

Ambient intelligence is a very vast field, and we have only scratched the surface. Coming from ubiquitous computing theorized by Mark Weiser in the 1990s and artificial intelligence, imagined by John McCarthy, Marvin Minsky, Nathaniel Rochester and Claude Shannon in the 1950s, it is not a passing trend, but a fundamental one, designed to last and evolve. It is presented not only as an evolution of computing, but also and above all as an evolution of society and our environment, making the latter more connected, more intelligent, but most importantly able to help people in their daily lives.

This thesis presents a small step in this field. We started from a simple premise: connected objects sold and used today communicate little with each other. From this premise, we first imagined the global operation of a system that automatically orchestrates the services offered by these objects. Several scientific constraints, such as current techniques in artificial intelligence, but also social constraints, such as the respect of privacy or personalization, allowed us to specify this operation. Thus, this system operates within the environment it must analyze, and not via centralized processing in the cloud, and makes automation proposals based on the habits found in the data of the connected objects. Object data is pre-processed, and habits are searched on these data to make automation proposals. As there are an infinite number of different environments, we also made the choice to create an agnostic system, which knows nothing about the data or objects to which it is connected. Finally, in order to adapt to users' desires, this system must consider user feedback.

Then, an architecture has been created based on the previous operation, with building blocks whose functions are clearly defined, to be easily understood by users who would like to know how the system works. It specifies how object data is processed to extract prediction rules, which will be proposed as automation proposals through an interface. Users will then be able to validate these rules, in which case they become active automations, which will

be applied automatically in the environment. Finally, these user feedbacks are considered to improve the different building blocks in order to propose more useful rules for them in the future.

As part of this thesis, several building blocks of the architecture have been designed and implemented.

Pre-processing takes into account the data of various objects, without having *a priori* on them. Using discretization algorithms, the data is unified within a single structure called an atom, and can be taken into account by the rule mining algorithm.

The algorithm for mining prediction rules, named TSRuleGrowth, is the major contribution of this thesis. Helped by a new notion of support, it responds to the multiple problems present in the state-of-the-art algorithms, such as the non-validation of reliable rules in certain cases, and allows several ways of guiding and limiting the search. This cannot only reduce the execution time, but also disregard the data of certain objects that users do not want.

We also presented some perspectives for this ambient intelligence system, regarding the algorithms presented in the thesis, but also on the interactions with the users and the consideration of their feedback.

As we said earlier, this is a step in the world of ambient intelligence. This field is still in its early stages, and many studies are still to come in several disciplines. It would not be unthinkable to imagine work in fields other than computer science, such as psychology, philosophy, or ergonomics. Ambient intelligence is above all a dream, that of having an environment that actively helps its users in their everyday life. However, we must not lose sight of the fact that the means made available to achieve this dream can also be used to track down or control the population. This is also the reason why we, for example, wanted to imagine a system that is not in the cloud. This danger is crucial, and must be taken into account by the various researchers working in this field.

Ambient intelligence is perhaps the next revolution in the world of information technology, and in our society, in the same way as transhumanism. This fusion between computer science and nature can bring many things to all human beings, and shows that computer science, even though it has already known three great eras, is still in its infancy.

Appendix A

Pre-processing

In this appendix, the pseudocodes of the algorithms used in pre-processing are detailed, more precisely the algorithms that discretize the continuous events. In particular, the Sliding Window Algorithm and hierarchical clustering are detailed.

Algorithm 7: Sliding Window Algorithm

Data: $TS = \langle (t_1, e_1), ..., (t_n, e_n) \rangle$: time series, ϵ: threshold
Result: Time series of continuous elements TS^{sim}
// Initialization
$TS^{sim} = \langle \rangle$;
$i \leftarrow 0$;
// Main loop
while *end is False* **do**
    // Create a new segment
    stop $\leftarrow$ False;
    $j \leftarrow i + 2$;
    // It the end of the time series is reached, stop all
    if $t_j > t_n$ **then**
        end $\leftarrow$ True;
        stop $\leftarrow$ True;
    while *stop is False* **do**
        // Compute the equation of the segment ($y = a * x + b$)
        $a \leftarrow \frac{e_j.value - e_i.value}{t_j - t_i}$;
        $b \leftarrow e_i.value - a * t_i$;
        // Compute the distance between the points of the time series and the segment
        foreach $(t_x, e_x) | t_i < t_x < t_j$ **do**
            point $\leftarrow a * t_x + b$;
            distance $\leftarrow$ |point - $e_x.value$|;
            if *distance* $> \epsilon$ **then**
                stop $\leftarrow$ True;
        // If no distance exceeds epsilon, try a bigger segment
        if *stop is False* **then**
            $j \leftarrow j + 1$;
    // If a distance exceeds epsilon, add the end of the previous segment to the time series
    Add (t_{j-1}, e_{j-1}) to TS^{sim};
    $i \leftarrow j - 1$;
Return TS^{sim};

Algorithm 8: Clustering algorithm

Data: Time series of elements TS^{sim}, Time series of elements TS^{ref}
Result: Time series of atoms $TS_a = \langle (t_1, a_1), ..., (t_{n_a}, a_{n_a}) \rangle, a_1, ..., a_{n_a} \in A_o$
`// Initialization`
$TS_a \leftarrow \langle\rangle$;
Convert TS^{sim} to a time series of segments TS^{ssim};
Convert TS^{ref} to a time series of segments TS^{sref};
Calculate the standardization components for the mean, variation and duration from segments in TS^{sref}: $\mu_{mean}, \sigma_{mean}, \mu_{variation}, \sigma_{variation}, \mu_{duration}, \sigma_{duration}$;
`// Delete the outliers`
foreach $(t_i, s_i) \in TS^{ssim}$ **do**
    `// Normalize the characteristics`
    $s_i.mean \leftarrow \frac{s_i.mean - \mu_{mean}}{\sigma_{mean}}$;
    $s_i.variation \leftarrow \frac{s_i.variation - \mu_{variation}}{\sigma_{variation}}$;
    $s_i.duration \leftarrow \frac{s_i.duration - \mu_{duration}}{\sigma_{duration}}$;
    `// If one of those characteristics is too different from the other segments, remove it`
    if $s_i.mean \notin [-3, 3]$ *or* $s_i.variation \notin [-3, 3]$ *or* $s_i.duration \notin [-3, 3]$ **then**
        Remove (t_i, s_i) from TS^{ssim};
`// Build the atoms`
Compute the dendrogram of the segments using Mean Linkage Clustering;
best_silhouette $\leftarrow$ 0;
foreach *Possible cutting of the dendogram* **do**
    Cut the dendogram and get the resulting groups of segments;
    Compute the silhouette value of this group configuration;
    if *silhouette* > *best_silhouette* **then**
        best_silhouette $\leftarrow$ silhouette;
        $\theta_{clu} \leftarrow$ granularity of the cutting;
Build the groups of segments according to θ_{clu} and the dendrogram;
`// Create the atoms corresponding to the groups of segments`
foreach *group of segments* **do**
    $m_{mean}, m_{variation}, m_{duration} \leftarrow$ averages of $mean, variation, duration$ of all the semgents in the group;
    `// Denormalize the characteristics`
    $m_{mean} \leftarrow m_{mean} * \sigma_{mean} + \mu_{mean}$;
    $m_{variation} \leftarrow m_{variation} * \sigma_{variation} + \mu_{variation}$;
    $m_{duration} \leftarrow m_{duration} * \sigma_{duration} + \mu_{duration}$;
    $a \leftarrow$ new Atom($o = o, value = [m_{mean}, m_{variation}, m_{duration}]$);
    Link a to this group;
`// Create the time series of atoms`
foreach $(t_i, s_i) \in TS^{ssim}$ **do**
    $a \leftarrow$ atom of the segment group in which s_i is located;
    Add (t_i, a) to TS_a;
return TS_a;

Appendix B

TSRuleGrowth

Here are presented the detailed pseudocodes of the rule search algorithm, TSRuleGrowth, and the new notion of support presented in this thesis.

Algorithm 9: Count

Data: A_m: multiset, $TS = \langle (t_1, I_1), ..., (t_n, I_n) \rangle, I_1, ..., I_n \subseteq A$: time series, $window$: duration

/* A_m is seen if all of its elements are seen in a window, where those elements have not been seen in a previous occurrence of A_m. The blacklist keeps tracking the occurrences of the elements of A_m that have been seen previously in an occurrence of A_m. */

// Initialization

foreach *unique* $a \in A_m$ **do**
  $b(a) \leftarrow \emptyset$; // A blacklist is assigned to every unique element of A_m
$sup(A_m) \leftarrow 0$; // Support of A_m: number of distinct occurrences
iterator $\leftarrow 1$; // Iterator used for the sliding window
start $\leftarrow t_1$; // Start of the window
end $\leftarrow t_1 + window$; // End of the window

// Sliding window through the time series

while *end* $\leq t_n \in TS$ **do**
  found $\leftarrow$ True;
  Scan TS through [start : end] and record the time stamps of the elements $a \in A_m$ in $T(a)$;
  foreach *element* $a \in A_m$ **do**
    $T(a) \leftarrow T(a) \setminus b(a)$; // Remove the occurrences of a that are in the blacklist
    if $|T(a)| <$ *multiplicity of* a *in* A_m **then** // Not enough occurrences of a in the window
      found $\leftarrow$ False ; // A distinct occurrence of A_m cannot be seen
  if *found is True* **then** // A distinct occurrence of A_m is seen
    $sup(A_m) \leftarrow sup(A_m) + 1$; // The support of A_m is incremented
    foreach *element* $a \in A_m$ **do**
      // The earlier time stamps of $T(a)$ are added to the blacklist of a
      $m \leftarrow$ multiplicity of a in A_m;
      $b(a) \leftarrow b(a) \cup$ earliest m time stamps of $T(a)$;
  // Iteration through the time series
  iterator $\leftarrow$ iterator + 1;
  start $\leftarrow t_{iterator}$ // $t_2, t_3 \ldots$
  end $\leftarrow$ start + $window$;
Return $sup(A_m)$;

Algorithm 10: TSRuleGrowth

Data: $TS = \langle (t_1, I_1), ..., (t_n, I_n) \rangle, I_1, ..., I_n \subseteq A$: times series, min_{sup}: minimum support, min_{int}: minimum interest, $window$: duration

Scan the database TS once. For each atom a found, record the time stamps of the itemsets that contains a in T(a). All the lists of time stamps T(a) are stored in T;
Delete the atoms a where $|T(a)| \leq min_{sup}$ from TS, and delete T(a) from T;
// Creation of basic rules
foreach *pair of atoms i, j* **do**
    $sup(i \Rightarrow j) \leftarrow 0$; // Support of the rule
    $O_c(i \Rightarrow j), O_p(i \Rightarrow j) \leftarrow []$; // Occurrences of the condition and the prediction
    /* Blacklists help calculate distinct occurrences of the rule, for the support */
    $b(i), b(j) \leftarrow \emptyset$; // Blacklists of the occurrences of the condition and the prediction
    foreach t_i *in T(i)* **do**
        foreach t_j *in T(j)* **do**
            if $0 < t_j - t_i \leq window$ **then** // If i and j are seen inside the window and i is before j
                /* A new occurrence of the rule $i \Rightarrow j$ is seen. It is stored for expanding it later */
                $O_c(i \Rightarrow j) \leftarrow O_c(i \Rightarrow j) + \{i : \{t_i\}\}$;
                $O_p(i \Rightarrow j) \leftarrow O_p(i \Rightarrow j) + \{j : \{t_j\}\}$;
                /* If t_i and t_j are not in the blacklists, a new distinct occurrence of the rule $i \Rightarrow j$ is seen. The support is incremented, and the time stamps are added to the blacklist to avoid using those elements for a new distinct occurrence */
                if $t_i \notin b(i)$ *and* $t_j \notin b(j)$ **then**
                    $sup(i \Rightarrow j) \leftarrow sup(i \Rightarrow j) + 1$;
                    $b(i) \leftarrow b(i) \cup \{t_i\}$;
                    $b(j) \leftarrow b(j) \cup \{t_j\}$;
    // Growth of basic rules
    /* If the rule has enough support, TSRuleGrowth tries to expand it, and computes its interest */
    if $sup(i \Rightarrow j) \geq min_{sup}$ **then**
        if *netconf*$(\frac{|T(i)|}{|TS|}, \frac{|T(j)|}{|TS|}, \frac{sup(i \Rightarrow j)}{|TS|})) \geq min_{sup}$ **then**
            Output rule;
        ExpandCondition$(TS, i, j, |T(j)|, O_c(i \Rightarrow j), O_p(i \Rightarrow j), window)$;
        ExpandPrediction$(TS, i, j, |T(i)|, O_c(i \Rightarrow j), O_p(i \Rightarrow j), window)$;

Algorithm 11: ExpandCondition

Data: TS: time series, A_c: multiset, A_p: multiset, $sup(A_p)$: support of A_p, $O_c(A_c \Rightarrow A_p)$: occurrences of the condition of $A_c \Rightarrow A_p$, $O_p(A_c \Rightarrow A_p)$: occurrences of the prediction of $A_c \Rightarrow A_p$, min_{sup}: minimum support, min_{int}: minimum interest, $window$: duration

// Growth of the original rule

/* For each occurrence of the rule $A_c \Rightarrow A_p$, the ocurrences of new items are tracked */

for *i from 0 to* $|O_c(A_c \Rightarrow A_p)|$ **do**

- $o_c \leftarrow O_c(A_c \Rightarrow A_p)[i]$; // Occurrence of the condition
- $o_p \leftarrow O_p(A_c \Rightarrow A_p)[i]$; // Occurrence of the prediction
- /* For each occurrence of $A_c \Rightarrow A_p$, the observation window is between: */
- $start \leftarrow$ last time stamp in $o_p - window$; // The end of the prediction - window
- $end \leftarrow$ easliest time stamp in o_p ; // And the beginning of the prediction
- /* For each new item k seen in the window, where $k \geq max(e), e \in A_p$ to avoid duplicates */
- **foreach** *item k seen in* $[start, end[$, *where* $k \geq max(e), e \in A_c$ **do**
  - $o_k \leftarrow$ occurences of k in $[start, end[$;
  - **if** *k has never been seen before* **then** // Initialization:
    - $A_{ck} \leftarrow A_c \cup \{k\}$; // New multiset, union of A_c and k
    - $sup(A_{ck} \Rightarrow A_p) \leftarrow 0$; // Support of the new rule
    - $O_c(A_{ck} \Rightarrow A_p), O_p(A_{ck} \Rightarrow A_p) \leftarrow []$; // Occurrences of the new rule
    - $B_c(A_{ck} \Rightarrow A_p) \leftarrow \{e : \emptyset | e \in A_{ck}\}$; // Blacklists for the atoms of A_{ck}
    - $B_p(A_{ck} \Rightarrow A_p) \leftarrow \{e : \emptyset | e \in A_p\}$; // Blacklists for the atoms of A_p
  - **foreach** *time stamp* $t_k \in o_k$ *(ascending order)* **do**
    - /* If k is larger than all atoms of A_c or (k is equal to the greatest atom of A_c and its time stamp is greater than all time stamps of k in A_c) */
    - **if** $k > \max(e), e \in A_c$ *or* $t_k >$ *all occurrences of k in* o_c **then**
      - /* A new occurrence of the rule $A_{ck} \Rightarrow A_p$ is seen. It is stored for expanding it later */
      - $O_c(A_{ck} \Rightarrow A_p) \leftarrow O_c(A_{ck} \Rightarrow A_p) + \{o_c + \{k : t_k\}\}$;
      - $O_p(A_{ck} \Rightarrow A_p) \leftarrow O_p(A_{ck} \Rightarrow A_p) + o_p$;
      - **if** *time stamps in* o_c *are not in* $B_c(A_{ck} \Rightarrow A_p)$ *and time stamps in* o_p *are not in* $B_p(A_{ck} \Rightarrow A_p)$ *and* $t_k \notin B_c(A_{ck} \Rightarrow A_p)$ **then**
        - /* A new distinct occurrence of the rule $A_{ck} \Rightarrow A_p$ is seen. The support is incremented, and the time stamps are added to the blacklists to avoid using those atoms for a new distinct occurrence */
        - $sup(A_{ck} \Rightarrow A_p) \leftarrow sup(A_{ck} \Rightarrow A_p) + 1$;
        - Add the time stamps of o_c and t_k to $B_c(A_{ck} \Rightarrow A_p)$;
        - Add the time stamps of o_p to $B_p(A_{ck} \Rightarrow A_p)$;

// Growth of the new rules found

foreach *item c where* $sup(A_{ck} \Rightarrow A_p) \geq min_{sup}$ **do** // If the rule has enough support

- **if** *netconf*$(\frac{sup(A_{ck})}{|TS|}, \frac{sup(A_p)}{|TS|}, \frac{sup(A_{ck} \Rightarrow A_p)}{|TS|}) \geq min_{int}$ **then**
  - Output rule;
- $sup(A_{ck}) \leftarrow Count(A_{ck}, TS, window)$; // Compute the support of A_{ck}
- ExpandCondition$(TS, A_{ck}, A_p, sup(A_p), O_c(A_{ck} \Rightarrow A_p), O_p(A_{ck} \Rightarrow A_p), window)$;

Algorithm 12: ExpandPrediction

Data: TS: time series, A_c: multiset, A_p: multiset, $sup(A_c)$: support of A_c, $O_c(A_c \Rightarrow A_p)$: occurrences of the condition of $A_c \Rightarrow A_p$, $O_p(A_c \Rightarrow A_p)$: occurrences of the prediction of $A_c \Rightarrow A_p$, min_{sup}: minimum support, min_{int}: minimum interest, $window$: duration

// Growth of the original rule

/* For each occurrence of the rule $A_c \Rightarrow A_p$, the ocurrences of new items are tracked */

1 **for** *i from 0 to* $|O_c(A_c \Rightarrow A_p)|$ **do**

2 $o_c \leftarrow O_c(A_c \Rightarrow A_p)[i]$; // Occurrence of the condition

3 $o_p \leftarrow O_p(A_c \Rightarrow A_p)[i]$; // Occurrence of the prediction

/* For each occurrence of $A_c \Rightarrow A_p$, the observation window is between: */

4 $start \leftarrow$ last time stamp in o_c ; // The end of the condition

5 $end \leftarrow$ earliest time stamp in $o_c + window$; // And the beginning of the condition + window

/* For each new item k seen in the window, where $k \geq max(e), e \in A_p$ to avoid duplicates */

6 **foreach** *item k seen in* $]start, end]$*, where* $k \geq max(e), e \in A_p$ **do**

7 $o_k \leftarrow$ occurences of k in $]start, end]$;

8 **if** *k has never been seen before* **then** // Initialization:

9 $A_{pk} \leftarrow A_p \cup \{k\}$; // New multiset, union of A_p and k

10 $sup(A_c \Rightarrow A_{pk}) \leftarrow 0$; // Support of the new rule

11 $O_c(A_c \Rightarrow A_{pk}), O_p(A_c \Rightarrow A_{pk}) \leftarrow []$; // Occurrences of the new rule

12 $B_c(A_c \Rightarrow A_{pk}) \leftarrow \{e : \emptyset | e \in A_c\}$; // Blacklists for the atoms of A_c

13 $B_p(A_c \Rightarrow A_{pk}) \leftarrow \{e : \emptyset | e \in A_{pk}\}$; // Blacklists for the atoms of A_{pk}

14 **foreach** $t_k \in o_k$ *(ascending order)* **do**

/* If k is larger than all atoms of A_p or (k is equal to the greatest atom of A_p and its time stamp is greater than all time stamps of k in A_p) */

15 **if** $k > \max(e), e \in A_p$ *or* $t_k >$ *all occurrences of* k *in* o_p **then**

/* A new occurrence of the rule $A_c \Rightarrow A_{pk}$ is seen. It is stored for expanding it later */

16 $O_c(A_c \Rightarrow A_{pk}) \leftarrow O_c(A_c \Rightarrow A_{pk}) + o_c$;

17 $O_p(A_c \Rightarrow A_{pk}) \leftarrow O_p(A_c \Rightarrow A_{pk}) + \{o_p + \{k : t_k\}\}$;

18 **if** *time stamps in* o_c *are not in* $B_c(A_c \Rightarrow A_{pk})$ *and time stamps in* o_p *are not in* $B_p(A_c \Rightarrow A_{pk})$ *and* $t_k \notin B_p(A_c \Rightarrow A_{pk})$ **then**

/* A new distinct occurrence of the rule $A_c \Rightarrow A_{pk}$ is seen. The support is incremented, and the time stamps are added to the blacklists to avoid using those atoms for a new distinct occurrence */

19 $sup(A_c \Rightarrow A_{pk}) \leftarrow sup(A_c \Rightarrow A_{pk}) + 1$;

20 Add the time stamps of o_c to $B_c(A_c \Rightarrow A_{pk})$;

21 Add the time stamps of o_p and t_k to $B_p(A_c \Rightarrow A_{pk})$;

// Growth of the new rules found

22 **foreach** *item k where* $sup(A_c \Rightarrow A_{pk}) \geq min_{sup}$ **do** // If the rule has enough support

23 $sup(A_{pk}) \leftarrow Count(A_{pk}, TS, window)$; // Compute the support of A_{pk}

24 **if** *netconf*$(\frac{sup(A_c)}{|TS|}, \frac{sup(A_{pk})}{|TS|}, \frac{sup(A_c \Rightarrow A_{pk})}{|TS|}) \geq min_{int}$ **then**

25 Output rule;

26 ExpandCondition($TS, A_c, A_{pk}, sup(A_{pk}), O_c(A_c \Rightarrow A_{pk}), O_p(A_c \Rightarrow A_{pk}), window$);

27 ExpandPrediction($TS, A_c, A_{pk}, sup(A_c), O_c(A_c \Rightarrow A_{pk}), O_p(A_c \Rightarrow A_{pk}), window$);

Appendix C

Databases

Here is the correspondence table between the objects present in the Orange4Home database [Cumin et al., 2017a] and their manually defined functionality, highlighting that these objects are not varied in their functions in section 6.2.2.

Table C.1: Mapping table between the objects present in the Orange4Home database [Cumin et al., 2017a] and their manually defined functionality

Object name	**Category**
bathroom_CO2	CO2
bathroom_door	door
bathroom_heater_command	heater
bathroom_heater_effective_mode	heater
bathroom_heater_effective_setpoint	heater
bathroom_heater_temperature	heater
bathroom_humidity	humidity sensor
bathroom_light1	light bulb
bathroom_light2	light bulb
bathroom_luminosity	luminosity sensor
bathroom_presence	presence sensor

Continued on next page

Table C.1 – continued from previous page

Object name	Category
bathroom_shower_coldwater_instantaneous	water flow sensor
bathroom_shower_coldwater_total	water flow sensor
bathroom_shower_door	door
bathroom_shower_hotwater_instantaneous	water flow sensor
bathroom_shower_hotwater_total	water flow sensor
bathroom_sink_coldwater_instantaneous	water flow sensor
bathroom_sink_coldwater_total	water flow sensor
bathroom_switch_bottom_left	switch
bathroom_switch_bottom_right	switch
bathroom_switch_top_left	switch
bathroom_switch_top_right	switch
bathroom_temperature	temperature
bedroom_CO2	CO2
bedroom_bed_pressure	bed pressure
bedroom_closet_door	door
bedroom_door	door
bedroom_drawer1	drawer
bedroom_drawer2	drawer
bedroom_heater1_command	heater
bedroom_heater1_effective_mode	heater
bedroom_heater1_effective_setpoint	heater
bedroom_heater1_temperature	heater
bedroom_heater2_command	heater
bedroom_heater2_effective_mode	heater
bedroom_heater2_effective_setpoint	heater
bedroom_heater2_temperature	heater
bedroom_humidity	humidity sensor
bedroom_light1	light bulb
bedroom_light2	light bulb
bedroom_light3	light bulb
bedroom_light4	light bulb
bedroom_luminosity	luminosity sensor
bedroom_noise	noise
bedroom_presence	presence sensor
bedroom_shutter1	shutter
bedroom_shutter2	shutter
bedroom_switch_bottom_left	switch
bedroom_switch_bottom_right	switch
bedroom_switch_middle_left	switch
bedroom_switch_middle_right	switch
bedroom_switch_top_left	switch
bedroom_switch_top_right	switch
bedroom_temperature	temperature
entrance_door	door

Continued on next page

Table C.1 – continued from previous page

Object name	Category
entrance_heater_command	heater
entrance_heater_effective_mode	heater
entrance_heater_effective_setpoint	heater
entrance_heater_temperature	heater
entrance_light1	light bulb
entrance_noise	noise
entrance_switch_left	switch
global_active_energy	electrical sensor
global_active_power	electrical sensor
global_clouds_ext	weather
global_coldwater_instantaneous	water flow sensor
global_coldwater_total	water flow sensor
global_commonID_ext	weather
global_condition_ext	weather
global_condition_id_ext	weather
global_current	electrical sensor
global_frequency	electrical sensor
global_gas_total	gas
global_heaters_temperature	heater
global_humidity_ext	humidity sensor
global_lighting_current	light bulb
global_lighting_partial_energy	light bulb
global_lighting_power	light bulb
global_lighting_total_energy	light bulb
global_lighting_voltage	light bulb
global_power_factor	electrical sensor
global_pressure_ext	weather
global_pressure_trend_ext	weather
global_rain_ext	weather
global_shutters_current	shutter
global_shutters_partial_energy	shutter
global_shutters_power	shutter
global_shutters_total_energy	shutter
global_shutters_voltage	shutter
global_snow_ext	weather
global_temperature_ext	temperature
global_temperature_feel_ext	temperature
global_voltage	electrical sensor
global_waterheater_current	electrical sensor
global_waterheater_partial_energy	electrical sensor
global_waterheater_power	electrical sensor
global_waterheater_status	water heater
global_waterheater_total_energy	electrical sensor
global_waterheater_voltage	electrical sensor

Continued on next page

Table C.1 – continued from previous page

Object name	Category
global_wind_direction_ext	weather
global_wind_speed_ext	weather
kitchen_CO2	CO2
kitchen_cooktop_current	electrical sensor
kitchen_cooktop_partial_energy	electrical sensor
kitchen_cooktop_power	electrical sensor
kitchen_cooktop_total_energy	electrical sensor
kitchen_cooktop_voltage	electrical sensor
kitchen_cupboard1	cupboard
kitchen_cupboard2	cupboard
kitchen_cupboard3	cupboard
kitchen_cupboard4	cupboard
kitchen_cupboard5	cupboard
kitchen_dishwasher_current	electrical sensor
kitchen_dishwasher_partial_energy	electrical sensor
kitchen_dishwasher_power	electrical sensor
kitchen_dishwasher_total_energy	electrical sensor
kitchen_dishwasher_voltage	electrical sensor
kitchen_fridge_current	electrical sensor
kitchen_fridge_door	door
kitchen_fridge_partial_energy	electrical sensor
kitchen_fridge_power	electrical sensor
kitchen_fridge_total_energy	electrical sensor
kitchen_fridge_voltage	electrical sensor
kitchen_hood_current	electrical sensor
kitchen_hood_partial_energy	electrical sensor
kitchen_hood_power	electrical sensor
kitchen_hood_total_energy	electrical sensor
kitchen_hood_voltage	electrical sensor
kitchen_humidity	humidity sensor
kitchen_light1	light bulb
kitchen_light2	light bulb
kitchen_luminosity	luminosity sensor
kitchen_noise	noise
kitchen_oven_current	electrical sensor
kitchen_oven_partial_energy	electrical sensor
kitchen_oven_power	electrical sensor
kitchen_oven_total_energy	electrical sensor
kitchen_oven_voltage	electrical sensor
kitchen_presence	presence sensor
kitchen_sink_coldwater_instantaneous	water flow sensor
kitchen_sink_coldwater_total	water flow sensor
kitchen_sink_hotwater_instantaneous	water flow sensor
kitchen_sink_hotwater_total	water flow sensor

Continued on next page

Table C.1 – continued from previous page

Object name	Category
kitchen_switch_bottom_left	switch
kitchen_switch_bottom_right	switch
kitchen_switch_top_left	switch
kitchen_switch_top_right	switch
kitchen_temperature	temperature
kitchen_washingmachine_current	electrical sensor
kitchen_washingmachine_partial_energy	electrical sensor
kitchen_washingmachine_power	electrical sensor
kitchen_washingmachine_total_energy	electrical sensor
kitchen_washingmachine_voltage	electrical sensor
label	context data
livingroom_CO2	CO2
livingroom_couch_noise	noise
livingroom_couch_plug_consumption	electrical sensor
livingroom_heater1_command	heater
livingroom_heater1_effective_mode	heater
livingroom_heater1_effective_setpoint	heater
livingroom_heater1_temperature	heater
livingroom_heater2_command	heater
livingroom_heater2_effective_mode	heater
livingroom_heater2_effective_setpoint	heater
livingroom_heater2_temperature	heater
livingroom_humidity	humidity sensor
livingroom_light1	light bulb
livingroom_light2	light bulb
livingroom_luminosity	luminosity sensor
livingroom_presence_couch	presence sensor
livingroom_presence_table	presence sensor
livingroom_shutter1	shutter
livingroom_shutter2	shutter
livingroom_shutter3	shutter
livingroom_shutter4	shutter
livingroom_shutter5	shutter
livingroom_switch1_bottom_left	switch
livingroom_switch1_top_left	switch
livingroom_switch1_top_right	switch
livingroom_switch2_top_left	switch
livingroom_switch2_top_right	switch
livingroom_table_luminosity	luminosity sensor
livingroom_table_noise	noise
livingroom_table_plug_consumption	electrical sensor
livingroom_temperature	temperature
livingroom_tv_plug_consumption	electrical sensor
livingroom_tv_status	tv status

Continued on next page

Table C.1 – continued from previous page

Object name	Category
office_AC_setpoint	electrical sensor
office_desk_plug_consumption	electrical sensor
office_door	door
office_heater_command	heater
office_heater_effective_mode	heater
office_heater_effective_setpoint	heater
office_heater_temperature	heater
office_light	light bulb
office_luminosity	luminosity sensor
office_noise	noise
office_presence	presence sensor
office_shutter	shutter
office_switch_left	switch
office_switch_middle	switch
office_switch_right	switch
office_tv_plug_consumption	electrical sensor
office_tv_status	tv status
staircase_light	light bulb
staircase_switch_left	switch
staircase_switch_right	switch
toilet_coldwater_instantaneous	water flow sensor
toilet_coldwater_total	water flow sensor
toilet_light	light bulb
toilet_switch_left	switch
toilet_switch_right	switch
walkway_light	light bulb
walkway_noise	noise
walkway_switch1_bottom_left	switch
walkway_switch1_bottom_right	switch
walkway_switch1_top_left	switch
walkway_switch1_top_right	switch
walkway_switch2_bottom_left	switch
walkway_switch2_bottom_right	switch
walkway_switch2_top_left	switch
walkway_switch2_top_right	switch

Appendix D

Survey

Here is a translation of the survey sent to 22 people, the purpose of which was to identify parameters to be taken into account in measuring the usefulness of a rule. This experiment is described in section 6.6.

Automation in an intelligent environment

You are, at this very moment, at home in an intelligent environment. This environment has detected several habits that you have, and, based on them, proposes to automate certain actions. For example, if we detect that you turn on the light after opening a door, we can suggest that the light turns on automatically each time the door is opened.

The list that follows represents habits observed in your home, in the form of prediction rules. A prediction rule is of the form "If ..., then ...". Note that the conditions of the rules are not ordered.

Which of the following proposals would you like to automate? You can choose as many rules as you want, and you will have to explain your global choices afterwards.

Thank you!

1. Which proposals would you accept? (Several choices are possible)

    - If the front door opens, then close the front door in 10 seconds.

- If kitchen cupboard 1 opens and someone is present in the living room sofa and it is 12 o'clock, then close kitchen cupboard 1 in 25 seconds.
- If the bathroom door opens and it is 12 o'clock, then close the bathroom door in 10 seconds.
- If the hallway light comes on and it is Friday, then turn the hallway light off in 20 seconds.
- If the fridge door opens, then close the fridge door in 30 seconds.
- If the bathroom door opens and bathroom light 1 goes out and bathroom light 2 goes out, then close the bathroom door in 10 seconds.
- If the bathroom door opens and it is 8 o'clock, then close the bathroom door in 10 seconds.
- If the room door opens and someone is present in the room, then close the room door in 30 seconds.
- If the office door opens and the office light comes on and someone is present in the office, then close the office door in 25 seconds.
- If the bathroom door opens and someone is present in the bathroom and it is 8 o'clock, then close the bathroom door in 25 seconds.
- If the hallway light comes on and it is 8 o'clock, then turn the hallway light off in 25 seconds.
- If kitchen cupboard 4 opens, then close kitchen cupboard 4 in 20 seconds.
- If the bathroom door opens and the hallway light goes out, then switch on bathroom light 1 in 20 seconds.
- If the office door opens and someone is present in the office, then close the office door in 25 seconds.
- If bathroom light 1 goes out and bathroom light 2 goes out and it is 8 o'clock, then close the bathroom door in 10 seconds.
- If the staircase light comes on and the hallway light comes on, then turn the hallway light off in 20 seconds.
- If the bathroom door opens and someone is present in the bathroom, then close the bathroom door in 10 seconds.
- If chamber drawer 1 opens, then close chamber drawer 1 in 20 seconds.
- If the bathroom door opens and someone is present in the bathroom and the hallway light goes out, then close the bathroom door in 25 seconds.
- If bathroom light 1 goes out and bathroom light 2 goes out, then close the bathroom door in 10 seconds.
- If the bathroom door opens and bathroom light 1 goes out and bathroom light 2 goes out and it is 8 o'clock, then close the bathroom door in 15 seconds.
- If the shower door opens and it is 8 o'clock, then close the shower door in 25 seconds.
- If the bathroom door opens, then close the bathroom door in 10 seconds.
- If the bathroom door opens and it is Tuesday, then close the bathroom door in 10 seconds.

- If kitchen cupboard 1 opens and it is 12 o'clock, then close kitchen cupboard 1 in 5 seconds.
- If the office door opens, then close the office door in 25 seconds.
- If the room door opens, then close the room door in 15 seconds.
- If the room door opens and it is 12 o'clock, then close the room door in 15 seconds.
- If the hallway light comes on and it's Thursday, then turn the hallway light off in 30 seconds.
- If chamber drawer 2 opens, then close chamber drawer 2 in 20 seconds.
- If the front door opens and the entrance light 1 goes out and the staircase light goes out, then close the front door in 10 seconds.
- If the shower door opens, then close the shower door in 25 seconds.
- If the bathroom door opens and no one is present in the kitchen and no one is present towards the living room table and the hallway light goes on and the hallway light goes off, then turn on bathroom light 1 in 30 seconds.
- If the office door opens and the office light goes out, then close the office door in 10 seconds.
- If kitchen cupboard 1 opens, then close kitchen cupboard 1 in 5 seconds.
- If the office light goes out, then close the office door in 10 seconds.

2. Could you explain your choices? (Free text)

3. You are...

    - A man
    - A woman

4. You are...

    - Under 15 years old
    - Between 15 and 19 years old
    - Between 20 and 25 years old
    - Between 25 and 29 years old
    - Between 30 and 35 years old
    - Between 35 and 39 years old
    - Between 40 and 45 years old
    - Between 45 and 49 years old
    - Between 50 and 55 years old
    - Between 55 and 59 years old
    - Between 60 and 65 years old
    - Between 65 and 69 years old
    - Between 70 and 75 years old

- Over 75 years old

5. You are in the category of...

   - Operating farmers
   - Craftsmen, traders and entrepreneurs
   - Professionals and Senior Professionals
   - Intermediate professions
   - Employees
   - Workers
   - Students, or no occupation

Bibliography

[Aarts and Encarnação, 2006] Aarts, E. H. L. and Encarnação, J. L., editors (2006). *True Visions: The Emergence of Ambient Intelligence*. Springer, Berlin ; New York.

[Abowd et al., 1999] Abowd, G. D., Dey, A. K., Brown, P. J., Davies, N., Smith, M., and Steggles, P. (1999). Towards a Better Understanding of Context and Context-Awareness. In Goos, G., Hartmanis, J., van Leeuwen, J., and Gellersen, H.-W., editors, *Handheld and Ubiquitous Computing*, volume 1707, pages 304–307. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Acampora et al., 2013] Acampora, G., Cook, D. J., Rashidi, P., and Vasilakos, A. V. (2013). A Survey on Ambient Intelligence in Healthcare. *Proceedings of the IEEE*, 101(12):2470–2494.

[Aggarwal et al., 2014] Aggarwal, C. C., Bhuiyan, M. A., and Hasan, M. A. (2014). Frequent Pattern Mining Algorithms: A Survey. In Aggarwal, C. C. and Han, J., editors, *Frequent Pattern Mining*, pages 19–64. Springer International Publishing, Cham.

[Ahn and Kim, 2004] Ahn, K.-I. and Kim, J.-Y. (2004). Efficient Mining of Frequent Itemsets and a Measure of Interest for Association Rule Mining. *Journal of Information & Knowledge Management*, 03(03):245–257.

[Allen, 1984] Allen, J. F. (1984). Towards a general theory of action and time. *Artificial Intelligence*, 23(2):123–154.

[Alpaydin, 2014] Alpaydin, E. (2014). *Introduction to Machine Learning*. MIT press.

[Apple, 2017] Apple (2017). An On-device Deep Neural Network for Face Detection. https://machinelearning.apple.com/2017/11/16/face-detection.html.

[Asimov, 1950] Asimov, I. (1950). *I, Robot*. Gnome Press, New York, 1st ed. edition.

[Augusto and McCullagh, 2007] Augusto, J. C. and McCullagh, P. (2007). Ambient intelligence: Concepts and applications. *Computer Science and Information Systems*, 4(1):1–27.

[Augusto and Nugent, 2004] Augusto, J. C. and Nugent, C. D. (2004). The use of temporal reasoning and management of complex events in smart homes. In *Proceedings of the 16th European Conference on Artificial Intelligence*, ECAI'04, pages 778–782, Valencia, Spain. IOS Press.

[Azevedo and Jorge, 2007] Azevedo, P. J. and Jorge, A. M. (2007). Comparing Rule Measures for Predictive Association Rules. In Kok, J. N., Koronacki, J., de Mantaras, R. L., Matwin, S., Mladenič, D., and Skowron, A., editors, *Machine Learning: ECML 2007*, Lecture Notes in Computer Science, pages 510–517, Berlin, Heidelberg. Springer.

[Aztiria et al., 2010] Aztiria, A., Izaguirre, A., and Augusto, J. C. (2010). Learning patterns in ambient intelligence environments: A survey. *Artificial Intelligence Review*, 34(1):35–51.

[Baccouche et al., 2012] Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2012). Spatio-Temporal Convolutional Sparse Auto-Encoder for Sequence Classification. In *Procedings of the British Machine Vision Conference 2012*, pages 124.1–124.12, Surrey. British Machine Vision Association.

[Balit et al., 2018] Balit, E., Vaufreydaz, D., and Reignier, P. (2018). PEAR: Prototyping Expressive Animated Robots - A framework for social robot prototyping. In *HUCAPP 2018 - 2nd International Conference on Human Computer Interaction Theory and Applications*, page 1.

[Barden and Leonard, 2011] Barden, L. H. and Leonard (2011). From the archive, 12 May 1997: Deep Blue win a giant step for computerkind. *The Guardian*.

[Bascol et al., 2016] Bascol, K., Emonet, R., Fromont, E., and Odobez, J.-M. (2016). Unsupervised Interpretable Pattern Discovery in Time Series Using Autoencoders. In Robles-Kelly, A., Loog, M., Biggio, B., Escolano, F., and Wilson, R., editors, *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science, pages 427–438, Cham. Springer International Publishing.

[Beaumont, 2008] Beaumont, B. C. (2008). Bill Gates's dream: A computer in every home. http://www.telegraph.co.uk/technology/3357701/Bill-Gatess-dream-A-computer-in-every-home.html.

[Ben Allouch et al., 2009] Ben Allouch, S., van Dijk, J. A. G. M., and Peters, O. (2009). The Acceptance of Domestic Ambient Intelligence Appliances by Prospective Users. In Tokuda, H., Beigl, M., Friday, A., Brush, A. J. B., and Tobe, Y., editors, *Pervasive Computing*, Lecture Notes in Computer Science, pages 77–94, Berlin, Heidelberg. Springer.

[Ben-David et al., 1997] Ben-David, S., Kushilevitz, E., and Mansour, Y. (1997). Online learning versus offline learning. *Machine Learning*, 29(1):45–63.

[Berndt and Clifford, 1994] Berndt, D. J. and Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining*, AAAIWS'94, pages 359–370, Seattle, WA. AAAI Press.

[Berners-Lee, 1989] Berners-Lee, T. (1989). The original proposal of the WWW, HTMLized. https://www.w3.org/History/1989/proposal.html.

[Betters, 2018] Betters, E. (2018). What is IFTTT and how does it work? https://www.pocket-lint.com/smart-home/news/130082-what-is-ifttt-and-how-does-it-work.

[Black and Pieterse, 2006] Black, P. E. and Pieterse, V. (2006). Manhattan distance.

[Blasco et al., 2014] Blasco, R., Marco, Á., Casas, R., Cirujano, D., and Picking, R. (2014). A Smart Kitchen for Ambient Assisted Living. *Sensors*, 14(1):1629–1653.

[Bosche et al., 2018] Bosche, A., Crawford, D., Jackson, D., Schallehn, M., and Schorling, C. (2018). Unlocking Opportunities in the Internet of Things. Technical report, Bain & Company.

[Bouakkaz et al., 2017] Bouakkaz, M., Ouinten, Y., Loudcher, S., and Fournier-Viger, P. (2017). Efficiently mining frequent itemsets applied for textual aggregation. *Applied Intelligence*, pages 1–7.

[Bühler, 2009] Bühler, C. (2009). Ambient Intelligence in Working Environments. In Stephanidis, C., editor, *Universal Access in Human-Computer Interaction. Intelligent and Ubiquitous Interaction Environments*, Lecture Notes in Computer Science, pages 143–149, Berlin, Heidelberg. Springer.

[Cadwalladr and Graham-Harrison, 2018] Cadwalladr, C. and Graham-Harrison, E. (2018). Revealed: 50 million Facebook profiles harvested for Cambridge Analytica in major data breach. *The Guardian*.

[Chee et al., 2019] Chee, C.-H., Jaafar, J., Aziz, I. A., Hasan, M. H., and Yeoh, W. (2019). Algorithms for frequent itemset mining: A literature review. *Artificial Intelligence Review*, 52(4):2603–2621.

[Chen et al., 2002] Chen, C.-Y., Hwang, S.-C., and Oyang, Y.-J. (2002). An Incremental Hierarchical Data Clustering Algorithm Based on Gravity Theory. In Chen, M.-S., Yu, P. S., and Liu, B., editors, *Advances in Knowledge Discovery and Data Mining*, pages 237–250. Springer Berlin Heidelberg.

[Chen et al., 2012a] Chen, L., Hoey, J., Nugent, C. D., Cook, D. J., and Yu, Z. (2012). Sensor-Based Activity Recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):790–808.

[Chen et al., 2012b] Chen, L., Nugent, C. D., and Wang, H. (2012). A Knowledge-Driven Approach to Activity Recognition in Smart Homes. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):961–974.

[Chen et al., 1996] Chen, M.-S., Han, J., and Yu, P. S. (1996). Data mining: An overview from a database perspective. *IEEE Transactions on Knowledge and data Engineering*, 8(6):866–883.

[Cheng et al., 2004] Cheng, H., Yan, X., and Han, J. (2004). IncSpan: Incremental mining of sequential patterns in large database. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 527–532, Seattle, WA, USA. Association for Computing Machinery.

[Christensen et al., 2004] Christensen, W. D., Hooker, C. A., et al. (2004). Representation and the meaning of life. *Representation in mind: New approaches to mental representation*, pages 41–69.

[Computer Chronicles, 1990] Computer Chronicles (1990). High Tech France - Part One. http://archive.org/details/frenchtech1.

[Computer Chronicles, 1995] Computer Chronicles (1995). Gary Kildall Special. http://archive.org/details/GaryKild.

[Cook et al., 2009] Cook, D. J., Augusto, J. C., and Jakkula, V. R. (2009). Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277–298.

[Cook et al., 2013] Cook, D. J., Crandall, A. S., Thomas, B. L., and Krishnan, N. C. (2013). CASAS: A Smart Home in a Box. *Computer*, 46(7):62–69.

[Cook et al., 2003] Cook, D. J., Youngblood, M., Heierman, E. O., Gopalratnam, K., Rao, S., Litvin, A., and Khawaja, F. (2003). MavHome: An agent-based smart home. In *Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, 2003. (PerCom 2003).*, pages 521–524.

[Cumin, 2018] Cumin, J. (2018). *Reconnaissance et Prédiction d'activités Dans La Maison Connectée*. PhD thesis, Université Grenoble Alpes (ComUE).

[Cumin et al., 2017a] Cumin, J., Lefebvre, G., Ramparany, F., and Crowley, J. L. (2017). A Dataset of Routine Daily Activities in an Instrumented Home. In *11th International Conference on Ubiquitous Computing and Ambient Intelligence (UCAmI)*.

[Cumin et al., 2017b] Cumin, J., Lefebvre, G., Ramparany, F., and Crowley, J. L. (2017). Human Activity Recognition Using Place-Based Decision Fusion in Smart Homes. In Brézillon, P., Turner, R., and Penco, C., editors, *Modeling and Using Context*, volume 10257, pages 137–150. Springer International Publishing, Cham.

[Cuofano, 2018] Cuofano, G. (2018). What Is a Hidden Revenue Business Model? Google's Business Model Explained. https://fourweekmba.com/hidden-revenue-model-google/.

[Das et al., 1998] Das, G., Lin, K.-I., Mannila, H., Renganathan, G., and Smyth, P. (1998). Rule discovery from time series. In *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining*, KDD'98, pages 16–22, New York, NY. AAAI Press.

[Deogun and Jiang, 2005] Deogun, J. and Jiang, L. (2005). Prediction Mining – An Approach to Mining Association Rules for Prediction. In *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, Lecture Notes in Computer Science, pages 98–108. Springer Berlin Heidelberg.

[Douglas and Peucker, 1973] Douglas, D. H. and Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122.

[Duman et al., 2019] Duman, T. B., Bayram, B., and İnce, G. (2019). Acoustic Anomaly Detection Using Convolutional Autoencoders in Industrial Processes. In *14th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2019)*, pages 432–442. Springer, Cham.

[Elegant, 2019] Elegant, N. X. (2019). The Internet Cloud's Dirty Secret: It Consumes Tons of Energy, Has Large Carbon Footprint. https://fortune.com/2019/09/18/internet-cloud-server-data-center-energy-consumption-renewable-coal/.

[European Parliament, 2016] European Parliament (2016). Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). http://data.europa.eu/eli/reg/2016/679/2016-05-04.

[Favreau, 2008] Favreau, J. (2008). Iron Man.

[Feng-Hsiung Hsu, 1999] Feng-Hsiung Hsu (1999). IBM's Deep Blue Chess grandmaster chips. *IEEE Micro*, 19(2):70–81.

[Ferrucci et al., 2010] Ferrucci, D., Brown, E., Chu-Carroll, J., Fan, J., Gondek, D., Kalyanpur, A. A., Lally, A., Murdock, J. W., Nyberg, E., Prager, J., Schlaefer, N., and Welty, C. (2010). Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3):59.

[Fournier-Viger et al., 2014] Fournier-Viger, P., Gueniche, T., Zida, S., and Tseng, V. S. (2014). ERMiner: Sequential Rule Mining Using Equivalence Classes. In *Advances in Intelligent Data Analysis XIII*, pages 108–119. Springer International Publishing.

[Fournier-Viger and Lin, 2017] Fournier-Viger, P. and Lin, J. C.-W. (2017). A Survey of Sequential Pattern Mining. *Data Science and Pattern Recognition*, pages 54–77.

[Fournier-Viger et al., 2015] Fournier-Viger, P., Wu, C.-W., Tseng, V. S., Cao, L., and Nkambou, R. (2015). Mining Partially-Ordered Sequential Rules Common to Multiple Sequences. *IEEE Transactions on Knowledge and Data Engineering*, 27(8):2203–2216.

[Frey, 2013] Frey, J. (2013). AdAPT – A Dynamic Approach for Activity Prediction and Tracking for Ambient Intelligence. In *2013 9th International Conference on Intelligent Environments*, pages 254–257.

[Fritzke, 1995] Fritzke, B. (1995). A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems*, pages 625–632.

[Frost, 2019] Frost, J. (2019). 5 Ways to Find Outliers in Your Data.

[Galbrun et al., 2018] Galbrun, E., Cellier, P., Tatti, N., Termier, A., and Cremilleux, B. (2018). Mining Periodic Patterns with a MDL Criterion. *CoRR*, abs/1807.01706:16.

[Galdeano et al., 2018] Galdeano, A., Gonnot, A., Cottet, C., Hassas, S., Lefort, M., and Cordier, A. (2018). Developmental Learning for Social Robots in Real-World Interactions. In *First Workshop on Social Robots in the Wild at the 13th Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI 2018)*, page 5, Chicago, IL, United States.

[Gardner, 1983] Gardner, H. (1983). *Frames of Mind: The Theory of Multiple Intelligences*. Basic Books, New York.

[Gardner, 1993] Gardner, H. (1993). *Frames of Mind: The Theory of Multiple Intelligences*. BasicBooks, New York, NY, 10th anniversary ed. edition.

[Georgia Institute of Technology, 1998] Georgia Institute of Technology (1998). Welcome | Aware Home Research Initiative. http://www.awarehome.gatech.edu/.

[Ghahramani, 2004] Ghahramani, Z. (2004). Unsupervised learning. In *Advanced Lectures on Machine Learning*, pages 72–112. Springer.

[Gil-Quijano and Sabouret, 2010] Gil-Quijano, J. and Sabouret, N. (2010). Prediction of Humans' Activity for Learning the Behaviors of Electrical Appliances in an Intelligent Ambient Environment. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, pages 283–286, Toronto, AB, Canada. IEEE.

[Gomes et al., 2019] Gomes, L., Ramos, C., Jozi, A., Serra, B., Paiva, L., and Vale, Z. (2019). IoH: A Platform for the Intelligence of Home with a Context Awareness and Ambient Intelligence Approach. *Future Internet*, 11(3):58.

[González García et al., 2017] González García, C., Meana-Llorián, D., Pelayo García-Bustelo, B., and Cueva Lovelle, J. (2017). A review about Smart Objects, Sensors, and Actuators. *International Journal of Interactive Multimedia and Artificial Intelligence*, 4:7–10.

[Gorrie, 2016] Gorrie, C. (2016). Three ways to detect outliers. http://colingorrie.github.io/outlier-detection.html.

[Gottfried et al., 2006] Gottfried, B., Guesgen, H. W., and Hübner, S. (2006). Spatiotemporal Reasoning for Smart Homes. *Designing Smart Homes*, pages 16–34.

[Griffith, 2019] Griffith, B. E. (2019). The US Is the Undisputed Leader in Smart Homes. https://www.pcmag.com/news/367137/the-us-is-the-undisputed-leader-in-smart-homes.

[Gross, 2011] Gross, D. (2011). Apple introduces Siri, Web freaks out. https://www.cnn.com/2011/10/04/tech/mobile/siri-iphone-4s-skynet/index.html.

[Gu et al., 2004] Gu, T., Wang, X. H., Pung, H. K., and Zhang, D. Q. (2004). An Ontology-based Context Model in Intelligent Environments. In *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, pages 270–275.

[Harnad, 1990] Harnad, S. (1990). The symbol grounding problem. *Physica D: Nonlinear Phenomena*, 42(1-3):335–346.

[Harnad, 2007] Harnad, S. (2007). Symbol grounding problem. *Scholarpedia*, 2(7):2373.

[Hauben, 2007] Hauben, M. (2007). History of ARPANET. http://pages.infinit.net/jbcoco/Arpa-Arpanet-Internet.pdf.

[Hawkins, 2012] Hawkins, D. (2012). Part II. Constructivism: Some history. *The Content Of Science: A Constructive Approach To Its Teaching And Learning*, page 9.

[Inria, 2013] Inria (2013). Smart Home – Amiqual4Home. https://amiqual4home.inria.fr/fr/tools/smart-home/.

[Jakkula and Cook, 2007] Jakkula, V. and Cook, D. (2007). Using Temporal Relations in Smart Environment Data for Activity Prediction. *Proceedings of the 24th International Conference on Machine Learning*.

[Jakkula and Cook, 2008] Jakkula, V. R. and Cook, D. J. (2008). Anomaly detection using temporal data mining in a smart home environment. *Methods of information in medicine*, 47(1):70–75.

[Jin and Agrawal, 2007] Jin, R. and Agrawal, G. (2007). Frequent Pattern Mining in Data Streams. In Aggarwal, C. C., editor, *Data Streams: Models and Algorithms*, Advances in Database Systems, pages 61–84. Springer US, Boston, MA.

[Johnson, 1967] Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

[Jones, 2001] Jones, J. (2001). Halo: Combat Evolved.

[Katehakis and Veinott, 1987] Katehakis, M. N. and Veinott, A. F. (1987). The Multi-Armed Bandit Problem: Decomposition and Computation. *Mathematics of Operations Research*, 12(2):262–268.

[Kay, 1993] Kay, S. M. (1993). *Fundamentals of Statistical Signal Processing*. Prentice Hall PTR.

[Keogh et al., 2001] Keogh, E., Chu, S., Hart, D., and Pazzani, M. (2001). An online algorithm for segmenting time series. In *Proceedings 2001 IEEE International Conference on Data Mining*, pages 289–296, San Jose, CA, USA. IEEE Comput. Soc.

[Keogh and Smyth, 1997] Keogh, E. and Smyth, P. (1997). A probabilistic approach to fast pattern matching in time series databases. In *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining*, KDD'97, pages 24–30, Newport Beach, CA. AAAI Press.

[Keshavarz et al., 2006] Keshavarz, A., Tabar, A. M., and Aghajan, H. (2006). Distributed vision-based reasoning for smart home care. In *Proc. of ACM SenSys Workshop on DSC*.

[Kieras and Meyer, 1997] Kieras, D. E. and Meyer, D. E. (1997). An overview of the EPIC architecture for cognition and performance with application to human-computer interaction. *Human-Computer Interaction*, 12(4):391–438.

[Kohonen, 1990] Kohonen, T. (1990). The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480.

[Kubrick, 1968] Kubrick, S. (1968). 2001: A Space Odyssey.

[Kuipers et al., 2006] Kuipers, B. J., Beeson, P., Modayil, J., and Provost, J. (2006). Bootstrap learning of foundational representations. *Connection Science*, 18(2):145–158.

[Kurzweil, 2006] Kurzweil, R. (2006). *The Singularity Is Near: When Humans Transcend Biology*. Penguin Books, New York.

[La Tona et al., 2018] La Tona, G., Petitti, A., Lorusso, A., Colella, R., Milella, A., and Attolico, G. (2018). Modular multimodal user interface for distributed ambient intelligence architectures. *Internet Technology Letters*, 1(2):e23.

[Lago et al., 2017] Lago, P., Lang, F., Roncancio, C., Jiménez-Guarín, C., Mateescu, R., and Bonnefond, N. (2017). The ContextAct@A4H Real-Life Dataset of Daily-Living Activities. In Brézillon, P., Turner, R., and Penco, C., editors, *Modeling and Using Context*, Lecture Notes in Computer Science, pages 175–188, Cham. Springer International Publishing.

[Laird et al., 1987] Laird, J. E., Newell, A., and Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64.

[Langley et al., 1991] Langley, P., McKusick, K. B., Allen, J. A., Iba, W. F., and Thompson, K. (1991). A design for the ICARUS architecture. *ACM SIGART Bulletin*, 2(4):104–109.

[LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

[Lee and Yun, 2017] Lee, G. and Yun, U. (2017). A new efficient approach for mining uncertain frequent patterns using minimum data structure without false positives. *Future Generation Computer Systems*, 68:89–110.

[Lovrić et al., 2014] Lovrić, M., Milanović, M., and Stamenković, M. (2014). Algoritmic methods for segmentation of time series: An overview. *Journal of Contemporary Economic and Business Issues*, 1(1):31–53.

[MacQueen, 1967] MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. The Regents of the University of California.

[Madhulatha, 2012] Madhulatha, T. S. (2012). An Overview on Clustering Methods. *IOSR Journal of Engineering*, 02(04):719–725.

[Mannila et al., 1997] Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of Frequent Episodes in Event Sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, USA.

[Masciotra, 2007] Masciotra, D. (2007). Le constructivisme en termes simples. *Vie pédagogique*, 143:48–52.

[Mazac, 2015] Mazac, S. (2015). *Approche Décentralisée de l'apprentissage Constructiviste et Modélisation Multi-Agent Du Problème d'amorçage de l'apprentissage Sensorimoteur En Environnement Continu : Application à l'intelligence Ambiante*. These de doctorat, Université Claude Bernard Lyon 1.

[Mazac et al., 2014] Mazac, S., Armetta, F., and Hassas, S. (2014). On bootstrapping sensori-motor patterns for a constructivist learning system in continuous environments. In *Artificial Life Conference Proceedings 14*, pages 160–167. The MIT Press.

[Mazzetti et al., 2019] Mazzetti, M., Perlroth, N., and Bergman, R. (2019). It Seemed Like a Popular Chat App. It's Secretly a Spy Tool. *The New York Times*.

[McCarthy, 2007] McCarthy, J. (2007). What is Artificial Intelligence?

[McCarthy and Hayes, 1969] McCarthy, J. and Hayes, P. J. (1969). Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press.

[Melnick, 2018] Melnick, K. (2018). Navigate Your Smart Home Devices In AR. https://vrscout.com/news/smart-home-devices-in-ar/.

[Meurer et al., 2018] Meurer, R. S., Frohlich, A. A., and Hubner, J. F. (2018). Ambient Intelligence for the Internet of Things Through Context-Awareness. In *2018 International Symposium on Rapid System Prototyping (RSP)*, pages 83–89, Torino, Italy. IEEE.

[Mnih et al., 2015] Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

[Moravec, 1990] Moravec, H. (1990). *Mind Children - The Future of Robot & Human Intelligence*. Harvard University Press, Cambridge, reprint edition.

[Murphy, 2012] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. MIT press.

[Nakashima et al., 2010] Nakashima, H., Aghajan, H. K., and Augusto, J. C., editors (2010). *Handbook of Ambient Intelligence and Smart Environments*. Springer, New York.

[Nazari Shirehjini and Semsar, 2017] Nazari Shirehjini, A. A. and Semsar, A. (2017). Human interaction with IoT-based smart environments. *Multimedia Tools and Applications*, 76(11):13343–13365.

[Newell and Simon, 1976] Newell, A. and Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, 19(3):113–126.

[Ng, 2019] Ng, A. (2019). Google calls Nest's hidden microphone an 'error'. https://www.cnet.com/news/google-calls-nests-hidden-microphone-an-error/.

[Ngiam et al., 2011] Ngiam, J., Khosla, A., Kim, M., Nam, J., Lee, H., and Ng, A. Y. (2011). Multimodal Deep Learning. *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696.

[Nguyen et al., 2005] Nguyen, S. N., Sun, X., and Orlowska, M. E. (2005). Improvements of IncSpan: Incremental Mining of Sequential Patterns in Large Database. In *Advances in Knowledge Discovery and Data Mining*, pages 442–451. Springer, Berlin, Heidelberg.

[Olaru et al., 2013] Olaru, A., Florea, A. M., and El Fallah Seghrouchni, A. (2013). A Context-Aware Multi-Agent System as a Middleware for Ambient Intelligence. *Mobile Networks and Applications*, 18(3):429–443.

[Orange, 2016] Orange (2016). Orange IoT Mashups. http://orange-iot-mashups.nprpaas.ddns.integ.dns-orange.fr/cloudlife/v2/homepage/?setLng=en.

[Pancardo et al., 2018] Pancardo, P., Wister, M., Acosta, F., and Hernández, J. A. (2018). Ambient Assisted Working Applications. In *Intelligent Data Sensing and Processing for Health and Well-Being Applications*, pages 81–99. Elsevier.

[Paulin et al., 2018] Paulin, R., Fraichard, T., and Reignier, P. (2018). Human-Robot Motion: Taking Human Attention into Account. In *IROS 2018- IEEE/RSJ International Conference on Intelligent Robots and Systems; Workshop on Assistance and Service Robotics in a Human Environment*, pages 1–5.

[Pazhaniraja et al., 2017] Pazhaniraja, N., Paul, P. V., Roja, G., Shanmugapriya, K., and Sonali, B. (2017). A study on recent bio-inspired optimization algorithms. In *2017 Fourth International Conference on Signal Processing, Communication and Networking (ICSCN)*, pages 1–6.

[Pei et al., 2001] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., and Mei-Chun Hsu (2001). PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th International Conference on Data Engineering*, pages 215–224.

[Philips, 2019] Philips (2019). Official Philips Hue apps. https://www2.meethue.com/en-us/app.

[Piaget, 1936] Piaget, J. (1936). *La Naissance de l'intelligence Chez l'enfant*. Delachaux et Niestlé.

[Piaget and Cook, 1952] Piaget, J. and Cook, M. (1952). *The Origins of Intelligence in Children*. International Universities Press New York.

[Porle et al., 2015] Porle, R. R., Ruslan, N. S., Ghani, N. M., Arif, N. A., Ismail, S. R., Parimon, N., and Mamat, M. (2015). A survey of filter design for audio noise reduction. *J. Adv. Rev. Sci. Res*, 12:26–44.

[Ramadan et al., 2010] Ramadan, R. A., Hagras, H., Nawito, M., Faham, A. E., and Eldesouky, B. (2010). The Intelligent Classroom: Towards an Educational Ambient Intelligence Testbed. In *2010 Sixth International Conference on Intelligent Environments*, pages 344–349, Kuala Lumpur, Malaysia. IEEE.

[Randall 5th, 2006] Randall 5th, A. (2006). Q&A: A lost interview with ENIAC co-inventor J. Presper Eckert. http://www.computerworld.com/article/2561813/computer-hardware/q-a–a-lost-interview-with-eniac-co-inventor-j–presper-eckert.html.

[Randell et al., 1992] Randell, D. A., Cui, Z., and Cohn, A. G. (1992). A Spatial Logic Based on Regions and Connection. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, KR'92, pages 165–176, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

[Rousseeuw, 1987] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

[Rushe, 2020] Rushe, D. (2020). $15bn a year: YouTube reveals its ad revenues for the first time. *The Guardian*.

[Sanchez-Picot et al., 2016] Sanchez-Picot, A., Martin, D., de Rivera, D. S., Bordel, B., and Robles, T. (2016). Modeling and Simulation of Interactions Among People and Devices in Ambient Intelligence Environments. In *2016 30th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 784–789. IEEE.

[Schilit, 2008] Schilit, B. (2008). Projects. https://sites.google.com/site/schilit/projects.

[Schlüter and Conrad, 2011] Schlüter, T. and Conrad, S. (2011). About the analysis of time series with temporal association rule mining. In *2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, pages 325–332.

[Searle, 1980] Searle, J. R. (1980). Minds, brains, and programs. *Behavioral and brain sciences*, 3(03):417–424.

[Shi et al., 2003] Shi, Y., Xie, W., Xu, G., Shi, R., Chen, E., Mao, Y., and Liu, F. (2003). The smart classroom: Merging technologies for seamless tele-education. *IEEE Pervasive Computing*, 2(2):47–55.

[Silver et al., 2018] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., and Hassabis, D. (2018). A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144.

[Silver et al., 2017] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359.

[Singh et al., 2017] Singh, D., Merdivan, E., Psychoula, I., Kropf, J., Hanke, S., Geist, M., and Holzinger, A. (2017). Human Activity Recognition Using Recurrent Neural Networks. In *Machine Learning and Knowledge Extraction*, pages 267–274. Springer, Cham.

[Somfy, 2019] Somfy (2019). myLink for Smartphones and Tablets. https://www.somfysystems.com/en-us/product-sheet-page.

[Spiegel et al., 2011] Spiegel, S., Gaebler, J., Lommatzsch, A., De Luca, E., and Albayrak, S. (2011). Pattern recognition and classification for multivariate time series. In *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data - SensorKDD '11*, pages 34–42, San Diego, California. ACM Press.

[Stikic and Schiele, 2009] Stikic, M. and Schiele, B. (2009). Activity Recognition from Sparsely Labeled Data Using Multi-Instance Learning. In Choudhury, T., Quigley, A., Strang, T., and Suginuma, K., editors, *Location and Context Awareness*, Lecture Notes in Computer Science, pages 156–173. Springer Berlin Heidelberg.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. MIT press Cambridge.

[Tan et al., 2018] Tan, P.-N., Steinbach, M., Karpatne, A., and Kumar, V. (2018). *Introduction to Data Mining*. Pearson, second edition.

[Tanbeer et al., 2009] Tanbeer, S. K., Ahmed, C. F., Jeong, B.-S., and Lee, Y.-K. (2009). Sliding window-based frequent pattern mining over data streams. *Information Sciences*, 179(22):3843–3865.

[Tang et al., 2016] Tang, Z., Guo, J., Miao, S., Acharya, S., and Feng, J. H. (2016). Ambient Intelligence Based Context-Aware Assistive System to Improve Independence for People with Autism Spectrum Disorder. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3339–3348, Koloa, HI, USA. IEEE.

[Tran et al., 2010] Tran, A. C., Marsland, S., Dietrich, J., Guesgen, H. W., and Lyons, P. (2010). Use Cases for Abnormal Behaviour Detection in Smart Homes. In *Aging Friendly Technology for Health and Independence*, pages 144–151. Springer, Berlin, Heidelberg.

[Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, LIX(236):433–460.

[Tuyls et al., 2018] Tuyls, K., Perolat, J., Lanctot, M., Leibo, J. Z., and Graepel, T. (2018). A Generalised Method for Empirical Game Theoretic Analysis. In *Proc. of the 17th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2018)*.

[van Kasteren et al., 2011] van Kasteren, T. L. M., Englebienne, G., and Kröse, B. J. A. (2011). Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software. In Chen, L., Nugent, C. D., Biswas, J., and Hoey, J., editors, *Activity Recognition in Pervasive Intelligent Environments*, Atlantis Ambient and Pervasive Intelligence, pages 165–186. Atlantis Press, Paris.

[Vaughan et al., 2018] Vaughan, J., Sudjianto, A., Brahimi, E., Chen, J., and Nair, V. N. (2018). Explainable Neural Networks based on Additive Index Models. *arXiv:1806.01933 [cs, stat]*.

[Vernon et al., 2007] Vernon, D., Metta, G., and Sandini, G. (2007). A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents. *IEEE transactions on evolutionary computation*, 11(2):151–180.

[Von Glasersfeld, 1984] Von Glasersfeld, E. (1984). An introduction to radical constructivism. *The invented reality*, pages 17–40.

[Want et al., 1995] Want, R., Schilit, B., Adams, N., Gold, R., Petersen, K., Goldberg, D., Ellis, J., and Weiser, M. (Dec./1995). An overview of the PARCTAB ubiquitous computing experiment. *IEEE Personal Communications*, 2(6):28–43.

[Warneke et al., 2001] Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. (2001). Smart Dust: Communicating with a cubic-millimeter computer. *Computer*, 34(1):44–51.

[Weiser, 1991] Weiser, M. (1991). The computer for the 21st century. *Scientific american*, 265(3):94–104.

[Weiser, 1996] Weiser, M. (1996). Nomadic Issues in Ubiquitous Computing. http://www.ubiq.com/hypertext/weiser/NomadicInteractive/.

[Widyantoro et al., 2002] Widyantoro, D., Ioerger, T., and Yen, J. (2002). An incremental approach to building a cluster hierarchy. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 705–708.

[Wouk, 2019] Wouk, K. (2019). 6 Smart Home Devices that Are Totally Useless. https://www.iottechtrends.com/useless-smart-home-devices/.

[Yan, 2016] Yan, S. (2016). A Google computer victorious over the world's 'Go' champion. https://money.cnn.com/2016/03/12/technology/google-deepmind-alphago-wins/index.html.

[Yang et al., 2017] Yang, C., Lu, X., Lin, Z., Shechtman, E., Wang, O., and Li, H. (2017). High-resolution image inpainting using multi-scale neural patch synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6721–6729.

[Yang and Karamanoglu, 2013] Yang, X.-S. and Karamanoglu, M. (2013). Swarm Intelligence and Bio-Inspired Computation. In *Swarm Intelligence and Bio-Inspired Computation*, pages 3–23. Elsevier.

[Yang et al., 2019] Yang, Z., Zhang, A., and Sudjianto, A. (2019). Enhancing Explainability of Neural Networks through Architecture Constraints. *arXiv:1901.03838 [cs, stat]*.

[Ye et al., 2015] Ye, J., Stevenson, G., and Dobson, S. (2015). KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, 19:47–70.

[Yu et al., 2018] Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5505–5514.

[Zaki, 2001] Zaki, M. J. (2001). SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Machine Learning*, 42(1):31–60.

[Zarrouk and Gouider, 2012] Zarrouk, M. and Gouider, M. S. (2012). Frequent Patterns mining in time-sensitive Data Stream. *arXiv:1206.1032 [cs]*.

[Zou et al., 2008] Zou, J., Han, Y., and So, S.-S. (2008). Overview of Artificial Neural Networks. *Artificial Neural Networks*, pages 14–22.