UNIVERSIDAD DE DEUSTO

LEARNING FOR DYNAMIC AND PERSONALISED KNOWLEDGE-BASED ACTIVITY MODELS

Tesis doctoral presentada por Gorka Azkune

Dirigida por Dr. Diego López-de-Ipiña y Dr. Liming Chen

Bilbao, Enero de 2015

UNIVERSIDAD DE DEUSTO

LEARNING FOR DYNAMIC AND PERSONALISED KNOWLEDGE-BASED ACTIVITY MODELS

Tesis doctoral presentada por Gorka Azkune
dentro del Programa de Doctorado en Ingeniería para la Sociedad de la Información y Desarrollo Sostenible

Dirigida por Dr. Diego López-de-Ipiña y Dr. Liming Chen

El doctorando Los directores

Bilbao, Enero de 2015

Author: Gorka Azkune
Advisor: Dr. Diego López-de-Ipiña
Co-advisor: Dr. Liming Chen

Text printed in Bilbao
First edition, January 2015

Ama eta Nagoreri

Abstract

Human activity recognition is one of the key competences for human adaptive technologies. The idea of such technologies is to adapt their services to human users, so being able to recognise what human users are doing is an important step to adapt services suitably.

One of the most promising approaches for human activity recognition is the knowledge-driven approach, which has already shown very interesting features and advantages. Knowledge-driven approaches allow using expert domain knowledge to describe activities and environments, providing efficient recognition systems. However, there are also some drawbacks, such as the usage of generic and static activity models, i.e. activities are defined by their generic features - they do not include personal specificities - and once activities have been defined, they do not evolve according to what users do.

This dissertation presents an approach to using data-driven techniques to evolve knowledge-based activity models with a user's behavioural data. The approach includes a novel clustering process where initial incomplete models developed through knowledge engineering are used to detect action clusters which describe activities and aggregate new actions. Based on those action clusters, a learning process is then designed to learn and model varying ways of performing activities in order to acquire complete and specialised activity models. The approach has been tested with real users' inputs, noisy sensors and demanding activity sequences. Results have shown that the 100% of complete and specialised activity models are properly learnt at the expense of learning some false positive models.

Resumen

El reconocimiento de actividades realizadas por humanos es una de las competencias clave para las tecnologías cuyo objetivo es adaptarse a los humanos. La principal idea de dichas tecnologías es adaptar los servicios que ofrecen a sus usuarios, por lo que ser capaz de identificar lo que el usuario hace en cada momento es muy importante para adaptar los sevicios de forma adecuada.

Uno de los sistemas más prometedores para reconocer las actividades humanas es el llamado sistema basado en el conocimiento, que ya ha demostrado varias propiedades interesantes y ventajas importantes. Los sistemas basados en el conocimiento permiten usar el conocimiento de expertos para describir las actividades y los entornos en los que se llevan a cabo, ofreciendo sistemas de reconocimiento eficientes. Sin embargo, dichos sistemas también presentan algunos inconvenientes, como son el uso de modelos de actividad genéricos y estáticos. Es decir, por un lado las actividades se definen por sus características generales, y por ello no son capaces de incorporar información personal, y por otro, una vez que las actividades se han definido, no hay mecanismos para hacer que evolucionen a medida que los usuarios cambian.

Esta tesis doctoral presenta un sistema de modelado que usa técnicas basadas en datos para hacer evolucionar modelos de actividad basados en el conocimiento usando los datos generados por un usuario. El sistema de modelado se basa en un algoritmo nuevo de *clustering* donde se usan modelos incompletos iniciales creados por técnicas basadas en el conocimiento para detectar grupos de acciones que describen actividades y poder añadir así nuevas acciones. Sobre estos grupos de acciones se despliega un proceso de aprendizaje para aprender y modelar distintas formas de realizar actividades. De este modo, se obtienen modelos completos y especializados de las actividades definidas inicialmente. El sistema de modelado se ha probado con información de usuarios reales, sensores defectuosos y secuencias exigentes de actividades. Los resultados muestran que se pueden aprender el 100 % de los modelos de actividad completos y especializados, con el inconveniente de aprender también algunos modelos falsos.

Laburpena

Giza-jarduerak igartzeko gaitasuna mugarrietako bat da gizakiengana egokitzen diren teknologiak garatzerako orduan. Teknologia horien helburua zerbitzuak gizakien behar eta nahietara egokitzea da, eta horretarako, giza-erabiltzailea momentuoro egiten ari dena igartzeko gai izatea oso garrantzitsua da.

Giza-jarduerak igartzeko egin diren sistemen artean, etorkizun oparoenetako bat duen sistema ezagutzan oinarriturikoa da. Ezagutzan oinarrituriko sistemek jada hainbat abantaila eta ezaugarri interesgarri erakutsi dituzte. Beren muinean dago adituen ezagutza erabili ahal izatea giza-jarduerak eta inguruneak deskribatzeko, eta modu horretara, jarduerak igartzeko sistema eraginkorrak eskaintzen dituzte. Hala ere, sistema horiek badituzte beren desabantailak ere. Hala nola, jarduera ereduak orokorrak izan ohi dira, hots, jarduerak beren ezaugarri orokorren medioz deskribatzen direnez, ez dituzte ezaugarri pertsonalak erabiltzen, eta gainera jarduerak behin definituz gero, ez dira denboran zehar eguneratzen.

Tesi honetan giza-jarduerak modelatzeko sistema berri bat aurkezten da. Erabiltzaileek sortutako datuak erabiliz eta datuetan oinarritutako teknikak aplikatuz, ezagutzan oinarritutako jarduera ereduak eguneratzako sistemak aurkezten dira. Sistemaren oinarrietako bat *clustering* algoritmo berri bat da. Bere bitartez, ezagutzan oinarrituriko hasierako eredu osatugabeak erabiltzen dira giza-jarduerak deskribatzen dituzten ekintza multzoak detektatu eta ekintza berriak gehitzeko. Ekintza multzo horien gainean, ikasketa prozesu bat jartzen da martxan, jarduera bat egiteko modu ezberdinak ikasi eta modelatzeko. Modu horretara, jarduera eredu oso eta espezializatuak ikas daitezke. Modelatze sistema benetako erabiltzaileen informazioa, sentsore zaratatsuak eta jarduera sekuentzia konplexuak erabiliz probatu da. Emaitzek erakusten duten arabera, jarduera eredu oso eta espezializatuak ikas daitezke %100eko arrakasta tasekin, nahiz eta eredu faltsu batzuk ere ikasten diren prozesuan.

Acknowledgements

Even though science aims at overcoming human biases and irrational behaviours, science is done by scientists. And scientists, in the end of the day, are human. Humans tend to feel gratitude for those who help them pursuing an objective. I am not an exception. I think it is very important to express that gratitude, because without the help and support of several people, this PhD dissertation would not have been possible. Thus, my objective, my dream, would only exist in my mind.

Lo primero de todo es agradecer a Diego: has creído en mí y has puesto toda la carne en el asador para que se pueda realizar esta tesis doctoral tal y como habíamos previsto. Después de muchos años, has conseguido que me sienta un investigador de nuevo.

It is also very important for me to thank Luke. Your research work inspired many of the ideas of this dissertation and I was lucky enough to get you on board. It is not only about your technical support and guidance, but also about your kindness. It has been a pleasure to work with you.

La persona que más ha tenido que soportarme ha sido sin duda Aitor. Desde el primer día me has ayudado mucho. Hemos tenido muchas reuniones y discusiones donde esta tesis ha ido cogiendo forma. Has revisado todo mi trabajo, nuestras publicaciones y me has echado una mano con la burocracia interna que siempre tanto me cuesta. Por todo ello, muchísimas gracias.

No podría olvidarme tampoco de todos los compañeros del laboratorio. Además de acogerme tan bien, os agradezco el esfuerzo que dedicasteis a la evaluación de esta tesis. Toda la información que me disteis ha sido clave para poder validar todo mi trabajo.

Pertsonarik garrantzitsuenak egunero zurekin daudenak dira. Eskerrik asko bihotz-bihotzez Nagoreri. Badakit zenbaterainoko sakrifizioak egin dituzun bide honetan. Zu izan zara nire sustengu egunero. Asko zor dizut eta espero dut denborarekin dena itzultzeko gai izango naizela. Eskerrik asko baita ere nire anaiei eta hiloba paregabeei, bizitzako momentu zoragarriak oparitu dizkidazuelako. Eta eskerrik asko noski Taber eta Edurneri. Zuen etxean besoak zabalik hartu nauzue dena askoz errazago bilakatuz.

Baina batez ere pertsona bat dut gogoan hitz hauek idaztean: ama. Tesi hau, beste gauza guztien gainetik, nire amarentzat omenaldi xume bat da. Badakit, ama, zuretzat berandu iritsi dela, ezin duzula gustatuko litzaizukeen bezala disfrutatu, baina oraindik badut esperantza zure burutxo horretan, txokoren batean eta momenturen batean, zure semeaz harro sentituko zarela. Ez zenuen hainbeste gustatzen zitzaizun pianoa jotzen ikasteko aukerarik izan, txikia zinenean lan egitea tokatu baitzitzaizun. Eta lan horrekin jarraituz, beti langile, beti maitakor, zure bizitza osoa eman duzu zure semeek dena izan dezaten. Hezkuntzaren garrantzia beti izan duzu oso present eta hala erakutsi diguzu guri ere. Inoiz ez dugu nahikoa egingo zuri eskertzeko. Ama, izan dadila lan hau nire esker on guztia erakusteko beste modu bat.

Eskerrik asko,

Gorka Azkune

January 2015

Contents

List of Figures

List of Tables

List of definitions

Acronyms

AA	Action Aggregator
ADL	Activities of Daily Living
AML	Activity Model Learner
API	Application Programming Interface
CHMM	Coupled Hidden Markov Model
COM	Continuous varied Order Multi Threshold
CRF	Conditional Random Field
CSV	Comma Separated Value
DBN	Dynamic Bayesian Network
DL	Description Logic
EAM	Extended Activity Model
EC	Event Calculus
GCP	Google conditional Probabilities
GNU	GNU’s Not Unix
GPS	Global Positioning System
HMM	Hidden Markov Model
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol

IAM	Initial Activity Model
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
ILSA	Independent Lifestyle Assistant
IO	Input/Output
JSON	JavaScript Object Notation
KL	Kullback-Leibler
LDS	Linear Dynamical System
NN	Nearest Neighbour
OWL	Web Ontology Language
PEAT	Planning and Execution Assistant and Trainer
RDF	Resource Description Framework
RFID	Radio Frequency Identification
SA^3	Semantic Activity Annotation Algorithm
SMC	Sequential Monte Carlo
SVM	Support Vector Machine

The first step, my son, which one makes in the world, is the one on which depends the rest of our days.

François Marie Arouet, Voltaire

CHAPTER 1 Introduction

HUMAN activity recognition aims at recognising what a human is doing in different environments and domains. To perform activity recognition, different kinds of sensors have to be deployed in human-populated environments to monitor inhabitants' behaviours and capture environmental changes generated by human actions. The information provided by those sensors has to be processed through data analysis techniques and/or knowledge representation formalisms to create appropriate activity models and subsequently use them for activity recognition. Thus using sensing and computing capabilities, activities performed by humans can be detected and recognised.

Being able to recognise what human beings are doing in their daily life can open the door to a lot of possibilities in diverse areas. Technology is becoming more and more human-centred in order to provide personalised services. In other words, technological services have to adapt to human users and not the other way around.

Following that trend, human activity recognition becomes a natural enabler to adaptive technologies. As such, it has become an important research topic in areas such as pervasive and mobile computing (Choudhury and Consolvo, 2008), ambient assisted living (Philipose and Fishkin, 2004), social robotics (Fong et al., 2003), surveillance-based security (Fernández-Caballero, 2012) and context-aware computing (Laerhoven and Aidoo, 2001).

The scientific community has developed two main approaches to solve activity recognition, namely the data-driven and knowledge-driven approaches. Data-driven approaches make use of large-scale datasets of sensors to learn activity models using data mining and machine learning techniques. On the other hand, knowledge-driven approaches exploit rich prior knowledge in the domain of interest to build activity models using knowledge engineering and management technologies.

For knowledge-driven activity recognition systems, a widely recognised drawback is that activity models are usually static, i.e. once they have been defined, they cannot be automatically adapted to users' specificities (Chen et al., 2014). This is a very restrictive limitation, because it is not generally possible to define complete activity models for every user. Domain experts have the necessary knowledge about activities, but this knowledge may not be enough to generate complete models in all the cases. To make knowledge-driven activity recognition systems work in real world applications, activity models have to evolve automatically to adapt to users' varying behaviours. It turns out that model adaptability and evolution are aspects that can be properly addressed by data-driven approaches. Hence, the objective of this dissertation is to use data-driven techniques to make knowledge-based activity models evolve automatically based on sensor data generated by specific users.

The remainder of this chapter is structured as follows: Section 1.1 explains the context and motivation of the research. Afterwards, Section 1.2 formulates the hypothesis, derived goals and the scope of the work. Section 1.3 describes the followed research methodology to achieve the goals, and Section 1.4 summarises the scientific and technical contributions of this dissertation. The chapter finalises with an outline of the dissertation in Section 1.5.

1.1 Context and Motivation

Independently from the approach used to implement activity recognition systems, it is usually the case that activity modelling and recognition are carried out separately. Activity modelling is very important for activity recognition, since it provides the models which have to be recognised in the sensor datasets generated by humans while they perform activities. While data-driven approaches obtain those models directly from user generated data, knowledge-driven approaches obtain them by means of knowledge engineering techniques. However, both approaches share usually the same assumption, i.e. activity modelling is performed once and afterwards, the generated models are used for activity recognition.

Assuming the static nature of activity models does not fit to the reality. People tend to change the way they perform the same activities in time. Hence, the same activity models cannot be used forever. From the point of view of data-driven activity modelling approaches, implementing dynamic activity models is not a problem, as incremental learning techniques can be used (Rashidi and Cook, 2011). Moreover, as activity models are learnt directly from user generated data, personalised activity models are naturally obtained. Two of the positive features of data-driven modelling are thus dynamic and personalised models. But those virtues pose also some drawbacks: activity models are not generic so they cannot be applied to other users and activity models are usually incomprehensible for humans which make their usage for other applications much harder.

On the other hand, knowledge-driven modelling produces generic and human understandable models, since activities are modelled by their intrinsic features and relations rather than from data, and activity models are expressed using logical and descriptive formalisms (Chen et al., 2012b). However, those models are usually static, i.e. once they have been defined they do not evolve. In addition to it, obtaining personalised models is very complicated because experts cannot know in advance all the personal features of a concrete user.

It is quite clear then that both activity modelling approaches have complementary advantages and disadvantages. An approach excels where the other fails. It seems natural thus to try to combine both approaches in order to develop a better activity modelling process where generic and personalised models can be obtained and those models evolve in time with user generated data.

In this dissertation, a continuous activity modelling process is presented, where domain experts provide initially generic activity models using knowledge engineering tools, to afterwards run data-driven learning algorithms on user generated data to learn personalised models. Such personalised models will be presented to the expert, who can add or update them in the knowledge base. This loop is continuously repeated in time to achieve dynamic and personalised models based on initially provided generic models and user generated data. Figure 1.1 shows a diagram with the described process.

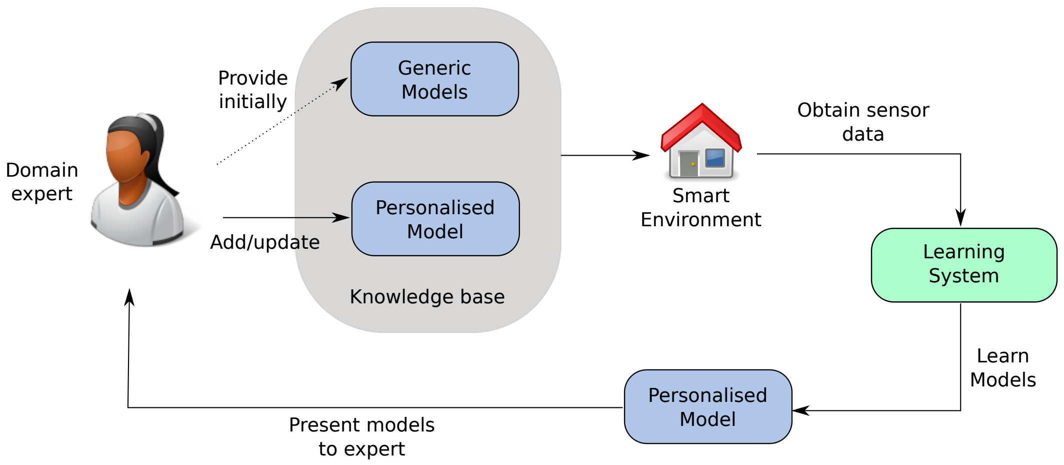


Figure 1.1: The continuous activity model process to obtain dynamic and personalised models from generic expert provided models.

The described activity modelling process uses generic knowledge-based activity models provided by a domain expert, achieving thus generic and understandable activity models. To solve the problem of personalised models, user generated data is used in a learning process to produce knowledge-based personalised models. This means that data-driven learning techniques are used to capture personal features of generic activities and obtain understandable personalised activity models. Such models, after expert validation, are added to the knowledge base to use them subsequently in activity recognition systems. Repeating this process continuously, dynamic activity models are obtained, which evolve in time as users change - or not - their habits.

When modelling activities, two major aspects are considered: the actions carried out to perform the activity and the descriptive features of the activity (Chen et al., 2014). Actions refer to object interactions. For example, to prepare a coffee a user might have coffee and a cup to drink the coffee. This sequence can be modelled as the action sequence of *hasCoffee* and *hasContainer*, which are obtained from the interactions with the respective objects monitored by sensors. Descriptive properties of an activity refer to the time when the activity has been performed, its duration, the concrete set of objects used and their order. Both aspects of an activity model, i.e. actions and descriptive properties, may vary from person to person.

As far as learning personalised descriptive properties for knowledge-based models regards, Chen et al. already combined data-driven and knowledge-driven approaches in (Chen et al., 2014). However, to the best of our knowledge, a similar work to learn specific actions for activity models has not been published yet. In that sense, the work presented in this dissertation tackles a novel problem. In fact, Chen et al. assume in their work that the actions in the *seed* activity models provided by domain experts can be applied to any user. This means that every user has to execute the same actions to perform a concrete activity.

In this dissertation, the problem of learning actions for given activities is analysed. It cannot be generally assumed that every user executes the same actions to perform a concrete activity. Imagine, for instance, that a user prepares a coffee and adds sugar to it. To make it simple, the activity model for this user could contain the actions *hasCoffee*, *hasContainer*, *hasWater* and *hasFlavour*. But there is another user who does not like any flavourant, so she prepares coffee by means of actions *hasCoffee*, *hasContainer* and *hasWater*. As it can be seen, both activity models are different as far as actions regard, but both of them are valid models for the activity of making a coffee.

To develop a proper activity modelling process, learning actions is a key feature. It has been observed that generic activity models provided by a domain expert have the necessary actions to perform an activity, but not sufficient actions for personalised models. Those activity models can be used to further learn new actions for different users and thus generate new personalised activity models. Let us illustrate our hybrid activity modelling approach with an example. Figure 1.2 shows an initial generic activity model for the MakeCoffee activity, which is composed by the actions *hasCoffee* and *hasContainer*. These are the necessary actions for every person to perform a MakeCoffee activity, which highlight the indispensable actions of making a coffee, i.e. the use of coffee and a container to place the coffee in. Nevertheless, some users might add some milk and sugar, others cream and so on. The idea of our approach is to create high-level activity models with only these indispensable actions - generic models -, and then use the data generated by a specific user performing the activity to learn those new actions which also configure the personal way of making coffee. In the case of Figure 1.2, where the initial activity model will only include coffee and container, the system would learn that MakeCoffee is performed in two ways by the user: in the first one, the user adds milk (*hasMilk*) and sugar (*hasFlavour*), while in the second one only sugar is added (*hasFlavour*). Hence two specialised and complete activity models of MakeCoffee can be learnt. This way, experts, initially, only have to provide generic but incomplete activity models with necessary actions. Afterwards the learning system can analyse a user's behavioural data and learn the specialised and complete models to enrich the knowledge base, thus improving initial activity models.

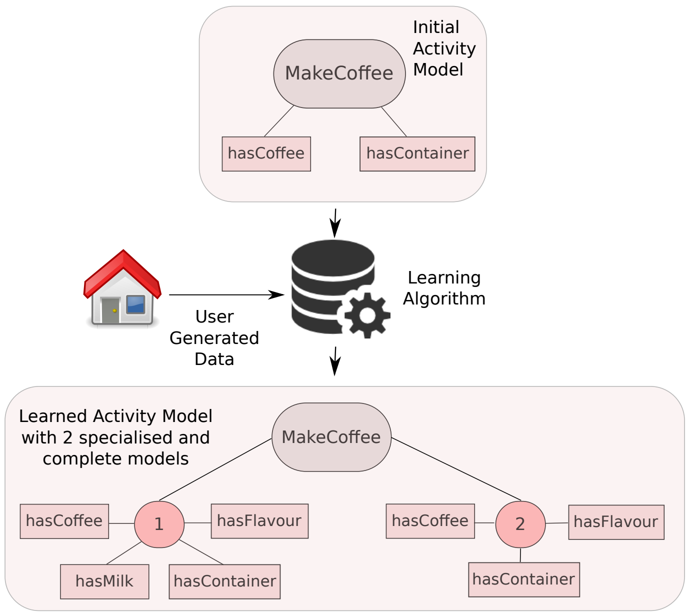


Figure 1.2: Illustrative example of the objective of the dissertation: using the initial incomplete model for MakeCoffee and user generated data, the learning algorithm learns two specialised and complete models.

Running the proposed learning process periodically with new data generated by a concrete user, a dynamic activity modelling system is achieved. As a user evolves regarding the way she performs certain activities, the learning approach learns new versions of the initial activity models. Hence, activity models can be adapted to users' varying behaviours.

Notice that the proposed modelling approach does not only learn new actions for initial activity models, but it also learns sub-activities of already defined activities. In the example depicted in Figure 1.2, the two specialised models are two sub-activities of the MakeCoffee activity, namely MakeBlackCoffee and MakeCoffeeWithMilk. This feature makes possible for the expert to define activities in a higher level of abstraction and let the system learn different sub-activities to generate specialised knowledge.

As a summary, this dissertation is focused on making another step towards dynamic and personalised activity models for knowledge-driven activity recognition systems. A learning system to obtain complete and specialised activity models from generic but incomplete expert provided models has been developed. This step allows implementing activity modelling approaches that use expert knowledge to have generic activity models applicable to every user, and learn new actions for concrete users achieving personalised and dynamic activity models.

1.2 Hypothesis, Objectives and Scope

Based on the current state of activity modelling and recognition, the hypothesis of this dissertation is:

Hypothesis. *Using domain experts' previous knowledge as generic but incomplete activity models and data-driven learning techniques on user generated data, it is possible to learn accurately new actions to obtain personalised activity models for every user.*

To be able to validate this hypothesis the general goal of this dissertation is:

Goal. *To design and implement a learning system that uses generic but incomplete activity models to analyse unlabelled activity sensor datasets and acquire personalised models which contain new actions.*

This general goal can be achieved by addressing the following more specific and measurable objectives:

1. To study the current state of the art on knowledge- and data-driven activity modelling and recognition.

2. To choose a knowledge representation formalism and design proper structures to represent domain experts' knowledge.

3. To design and implement a multiple step learning algorithm which uses previous knowledge and user generated data to obtain personalised activity models.

4. To identify the evaluation methodology for the learning system which better addresses the requirements of the system.

5. To validate the obtained results quantitatively, with the objective of capturing the 100% of real activity models performed by a user.

The resulting learning system should also fulfil the following requirements:

1. User independence: the learning system should be able to acquire personalised models for any user.

2. Use the same generic activity models for every user: to show that incomplete activity models provided by experts are really generic, the same models have to be used for every user.

3. Environment independence: the knowledge representation formalism and the learning algorithm should be defined to cope with different environments. As such, representing new locations and objects of a different environment should be straightforward.

The work presented in this dissertation does not deal with the following conditions:

1. Unknown activities: this dissertation does not propose any method to identify and learn unknown activities. Personalised models of already defined activities are learnt, thus leaving the integration of other techniques to learn unknown activities for future work.

2. Multiple users being monitored: it is assumed that only one user will be monitored in a particular experiment and dataset. Considering multi-user monitoring approaches is beyond the scope of this research work.

3. Concurrent and interwoven activities: users are not allowed to perform activities concurrently, i.e. once they start an activity, they have to finish it before starting another activity. Concurrent and interwoven activities pose challenges that will not be addressed in this dissertation.

1.3 Methodology

A research strategy has been defined in order to achieve the statement and derived goals presented in Section 1.2. The strategy is defined as follows:

1. Update knowledge by reviewing the literature in the area of activity modelling and recognition, knowledge engineering, machine learning and data mining. This analysis has been reinforced by attending specialised scientific conferences.
2. Critically evaluate existing activity modelling and recognition solutions, analysing their scope and limitations and identifying weak areas where contributions to the state of the art can be done.
3. Design and develop the different modules of the activity model learner, incrementally adopting more complex and efficient solutions.
4. Test developed modules through experiments and analyse results to enhance the performance of the system.
5. Attend conferences and workshops to present partial results and validate them with the scientific community.
6. Experimentation and evaluation of the prototype at each particular stage.
7. Network with experts at conferences, project meetings and consortia (the author is an active member of the SONOPA project[1] - SOcial Networks for Older adults to Promote an Active Life -). Contact for particular details by e-mail or by visiting other research groups (the author was a visiting researcher in De Montfort University for 6 weeks in 2014).
8. Redesign the activity model learner system with feedback from all this network, as well as the literature.
9. Select an appropriate evaluation methodology and evaluate consequently the activity model learner system.
10. Dissemination of the results obtained during the research process.

This methodology is illustrated in Figure 6.1, as a cyclic process which starts with the review of the state of the art and finishes with the publications and the final prototype.

[1]http://www.sonopa.eu/

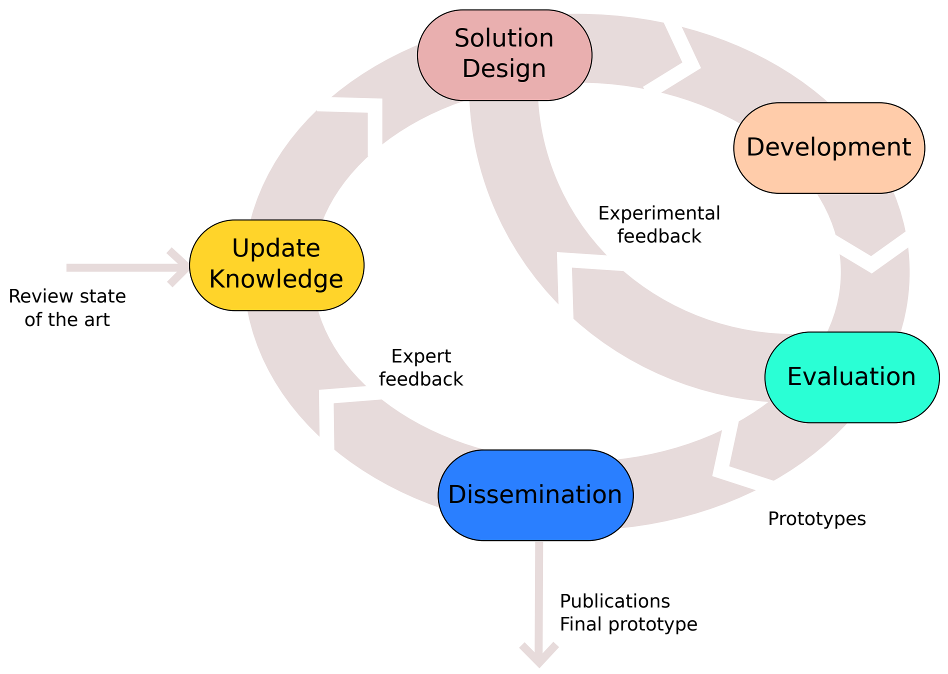


Figure 1.3: Followed research methodology.

1.4 Contributions

The following scientific contributions can be found in this dissertation:

- A two-step activity clustering algorithm which uses generic but incomplete activity models and context knowledge to recognise action clusters that form an activity and aggregate new actions. The activity clustering algorithm idea is introduced in Chapter 3 and is explained through Chapter 4.
- A learning algorithm that uses action clusters which describe activities to learn personalised activity models for every defined activity. This learning algorithm is first presented in Chapter 3 and further developed in Chapter 5.
- A hybrid evaluation methodology for activity modelling and recognition, based on surveys to users and simulation tools. This evaluation methodology is fully described and applied in Chapter 6. The details of the methodology can be found in Section 6.1.2.

In order to deliver those scientific contributions, a technical contribution has also be posed in this dissertation:

- A synthetic dataset generator tool: a special simulation tool for sensor dataset generation designed and developed to apply the hybrid evaluation methodology to the activity modelling approach. The software tool is described in Chapter 6, concretely in Section 6.1.2.2.

1.5 Thesis Outline

The thesis is structured in seven chapters.

Chapter 1 is the current chapter. It presents the context and motivation of the research, as well as the hypothesis, goals and scope. To achieve those goals and validate the hypothesis, a research methodology is proposed. Finally, the contributions of the dissertation and their location in the document are also shown.

Chapter 2 describes the state of the art relevant to the dissertation. The most important activity monitoring, modelling and recognition approaches are presented providing a well structured taxonomy. The chapter also shows the position in such taxonomy of the research performed in this dissertation.

Chapter 3 establishes the basis of the dissertation, giving a high-level view of the whole system. First of all, the chapter describes in detail ontology-based activity modelling approaches, which serve as reference for the work carried out in this dissertation. Afterwards, a set of definitions and constraints is posed to establish the scope of the research clearly. The knowledge representation formalism and structures are presented and the input data of the system is identified. Finally, the intuition behind the learning system and a high-level design are described.

Chapter 4 describes the proposed activity clustering algorithm, which is divided into two main steps: initialisation and action aggregation. Both steps are described in detail, providing pseudo-code for their implementation.

Chapter 5 explains the last stage of the learning process. Design decisions are explained and a rigorous analysis of all available information to learn activity models is done. The learning algorithm's steps are described in detail and pseudo-code for its implementation is provided.

Chapter 6 evaluates the complete learning system and its constituent parts. For this purpose, the most relevant evaluation methodologies for activity recognition systems are shown and their suitability to evaluate the learning system is analysed. The chapter proposes a new hybrid evaluation methodology, based on surveys to users and simulation tools. Afterwards, evaluation scenarios and metrics are presented and obtained results are shown, leading to a discussion of the learning system, its advantages and drawbacks.

Finally, Chapter 7 summarises the dissertation, draws the conclusions of this research work and shows the related future lines of work.

If I have seen further it is by standing on the shoulders of giants.

Isaac Newton

CHAPTER

2 Related Work

HUMAN activity recognition is a broad research area which can be analysed from many different points of view. The objective of this chapter is to provide a high-level picture of the current status of the research topic, to then describe more in detail the concrete approaches that are related to the work presented in this dissertation.

First of all, Section 2.1 introduces the concept of *Human Activity Recognition*, shows its main applications and provides a high-level taxonomy of activity recognition approaches. Section 2.2 describes in detail the category in which this dissertation fits regarding the monitoring approach: *Sensor-Based Activity Recognition*. Based on the specific sensors used for activity monitoring, Section 2.3 presents a more detailed taxonomy for the category of sensor-based activity recognition. On the other hand, the following sections analyse the main currents of activity modelling, which is the focus of this dissertation: Section 2.4 presents *Data-Driven Approaches*, while Section 2.5 describes in detail *Knowledge-Driven Approaches*. A summary and comparison of both activity modelling currents is provided in Table 2.1. Some recent work has settled the basis to combine both currents, giving as result the *Hybrid approaches*, as shown in Section 2.6. This dissertation is another step in order to achieve hybrid approaches that can combine effectively the best features of knowledge- and data-driven approaches. The chapter finalises with Section 2.7, where a concise summary is provided alongside with some conclusions.

2.1 Human Activity Recognition

Human activity recognition has become a key research topic in diverse areas, including pervasive and mobile computing (Weiser, 1991) (Choudhury and Consolvo, 2008), surveillance-based security (Poppe, 2010), (Akdemir et al., 2008), (Weinland et al., 2011), context-aware computing (Laerhoven and Aidoo, 2001), (Wren and Tapia, 2006), ambient assisted living (Philipose and Fishkin, 2004), (Cook and Schmitter-Edgecombe, 2009), (Kasteren and Noulas, 2008), (Chen et al., 2012b) and social robotics (Fong et al., 2003). The recent increase in interest in activity recognition can be attributed to the intensive thrusts from the latest technology development and application demands. The progress in sensor technologies has been substantial over the past decade, especially low-power, low-cost, high-capacity and miniaturised sensors, wired and wireless communication networks (Pantelopoulos and Bourbakis, 2010), (Alemdar and Ersoy, 2010), (Ding et al., 2011). In parallel, data processing techniques have also shown important advances, based on higher computational capabilities of devices and the development of novel algorithms. The progress and maturity of these supporting technologies have pushed the research focuses of the aforementioned areas to shift from low-level data collection and transmission towards high-level information integration, context processing and activity recognition and inference.

At the same time, activity recognition has been demanded by a growing number of solutions for real-world problems and applications. For example, surveillance and security try to make use of activity recognition technologies to address the threats of terrorists (Akdemir et al., 2008). Ambient assisted living aims to exploit activity monitoring, recognition and assistance to support independent living and ageing in place. Other emerging applications, such as intelligent meeting rooms (Mikic et al., 2000) and smart hospitals (Sánchez et al., 2008), are also dependent on activity recognition in order to provide multimodal interactions, proactive service provision, and context aware personalised activity assistance.

As a result of the technology push and application pull, activity recognition has built its own space in the academic world. As such, research related to activity recognition has become regular topics in mainstream international conferences in related areas such as the AAAI Conference on Artificial Intelligence[1], Computer Vision and Pattern Recognition[2], International Joint Conference on Artificial Intelligence[3], International Joint Conference on Pervasive and Ubiquitous Computing[4], International Conference on Pervasive Computing and Communications[5] and In-

[1]http://www.aaai.org/
[2]http://www.cvpr201x.org/
[3]http://ijcai.org/
[4]http://ubicomp.org/
[5]http://www.percom.org/

ternational Joint Conferences on Ambient Intelligence[1]. In order to try to channel the research towards future applications, a substantial number of projects and initiatives have been undertaken. Good examples are *The Ambient Assisted Living Joint Programme*[2], *The House of the Future*[3], *The Gator-Tech Smart House*[4] or *The iDorm project*[5].

Apart from potential applications, activity recognition has gained research interest because it is a multidisciplinary and complex process. Activity recognition can be roughly characterised by four basic tasks. These tasks include:

1. Selection and deployment of appropriate sensors to objects and environments in order to monitor and capture a user's behaviour along with the state change of the environment.

2. To collect, store and process perceived information through data analysis techniques and/or knowledge representation formalisms at appropriate levels of abstraction.

3. To create computational activity models in a way that allows software systems/agents to conduct reasoning and manipulation.

4. To select or develop reasoning algorithms to infer activities from sensor data.

Each individual task can be tackled by a great variety of methods, technologies and tools. It is often the case that the selection of a method used for one task is dependent on the method of another task. As such, activity recognition has been classified in the following ways according to (Chen et al., 2012a). The first classification criterion focuses on the sensors used for activity monitoring, while the second one pays attention to activity modelling techniques:

1. *Vision-based vs. sensor-based activity recognition:* In terms of the type of sensor that is used for activity monitoring, activity recognition can be generally classified into two categories. The first is referred to as vision-based activity recognition, which is based on the use of visual sensing facilities such as video cameras to monitor an actor's behaviour and environmental changes. The generated sensor data are video sequences or digitized visual data. The approaches in this category exploit computer vision techniques, including feature extraction, structural modelling, person and object recognition, movement segmentation, action extraction and movement tracking to

[1]http://www.ami-conferences.org/
[2]www.aal-europe.eu
[3]http://architecture.mit.edu/house_n
[4]http://www.icta.ufl.edu/gt.htm
[5]http://cswww.essex.ac.uk/iieg/idorm.htm

analyse visual observations for pattern recognition. The second category is referred to as sensor-based activity recognition, which is based on the use of emerging sensor network technologies for activity monitoring (Section 2.2). The generated sensor data from sensor-based monitoring are mainly time series of state changes and/or various parameter values that are usually processed through data fusion, probabilistic or statistical analysis methods and formal knowledge technologies for activity recognition. Sensor-based activity monitoring can be further classified into two categories: (i) wearable sensor-based activity monitoring, where sensors are attached to an actor under observation, and (ii) dense sensing-based activity monitoring, where sensors are attached to objects that constitute the activity environment. Wearable sensors, including modern smartphones, often use inertial measurement units and RFID tags to gather an actor's behavioural information. This approach is effective for recognising physical movements such as physical exercises. In contrast, dense sensing infers activities by monitoring human-object interactions through the usage of multiple multi-modal miniaturised sensors.

2. *Data-driven vs. knowledge-driven activity recognition:* The information obtained through activity monitoring has to be structured and processed to recognise activities. For that purpose, activity models play a critical role. In particular, the mechanisms activities are recognised are closely related to the nature and representation of activity models. Generally speaking, activity models can be built using one of two methods. The first is to learn activity models from pre-existent large-scale datasets of users' behaviours using data mining and machine learning techniques. This method involves the creation of probabilistic or statistical activity models, followed by training and learning processes. As this method is driven by data, and the ensued activity inference is based on probabilistic or statistical classification, it is often referred to as data-driven or bottom-up approaches (in the rest of this dissertation, data-driven will be used to refer to this category). The advantages of the data-driven approaches are the capabilities of handling uncertainty and temporal information (Brand et al., 1997). However, this method requires large datasets for training and learning, and suffers from the data scarcity or the "cold start" problem. It is also difficult to apply learnt activity models from one person to another. As such this method suffers from the problems of scalability and reusability. Data-driven approaches are described in detail in Section 2.4. The other method for building activity models is to exploit rich prior knowledge in the domain of interest to construct activity models directly using knowledge engineering and management technologies. This usually involves knowledge acquisition, formal modelling and representation. Activity models generated in this method are normally used for activity

recognition or prediction through formal logical reasoning, e.g., deduction, induction or abduction. As such, this method is referred to as knowledge-driven or top-down approach (knowledge-driven will be used from now on). Knowledge-driven approaches have the advantages of being semantically clear, logically elegant and easy to get started. But they are weak in handling uncertainty and temporal information and the models could be viewed as static and incomplete. For a complete review of knowledge-driven approaches, see Section 2.5. With the purpose of combining the advantages of those two methods, hybrid approaches have recently emerged (Section 2.6).

The situation generated by both classification criteria is represented in Figure 2.1, providing a high-level view of activity recognition systems' taxonomy.

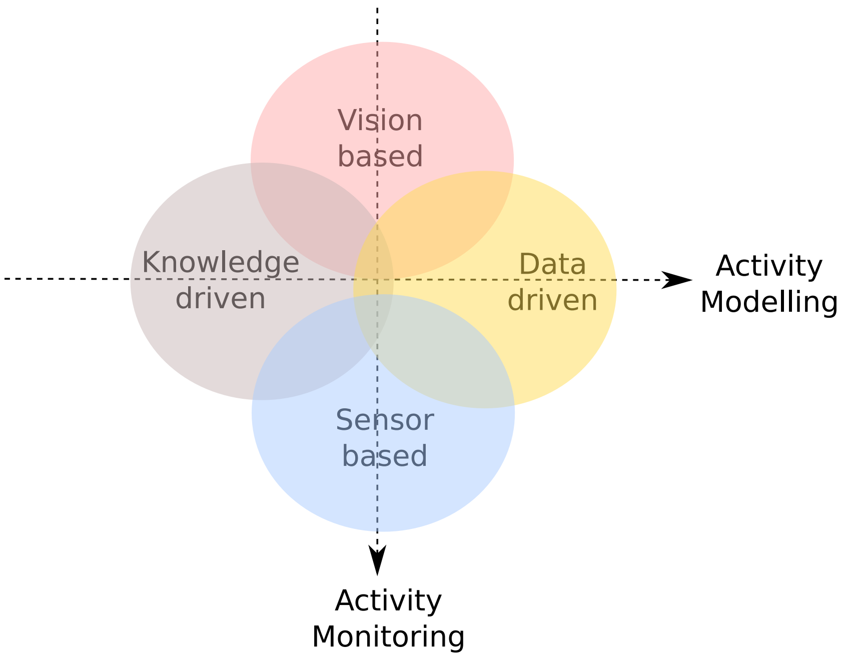


Figure 2.1: A high-level taxonomy of activity recognition systems, spanned by two axis: monitoring and modelling approach.

Vision-based activity recognition has been a research focus for a long period of time due to its important role in areas such as surveillance, robot learning and security. However, as this dissertation is based on sensor-based approaches, an exhaustive review of vision-based systems is not provided. Detailed up to date reviews can be found in (Poppe, 2010), (Moeslund et al., 2006), (Yilmaz et al., 2006), (Weinland et al., 2011) and (Turaga, 2008). Even though all those surveys cover different topics, together they have provided an extensive overview on the vision-based approach. It is concluded from those contributions that while visual monitoring is intuitive and information-rich, vision-based activity recognition suffers from issues relating to privacy and ethics (Yilmaz et al., 2006) as cameras are generally perceived as recording devices.

Compared to the number of surveys in vision-based activity recognition, and considering the wealth of literature in sensor-based activity recognition, there is a lack of extensive review on the state of the art of sensor-based activity recognition. This may be because the approach only recently became feasible when the sensing technologies matured to be realistically deployable in terms of the underpinning communication infrastructure, costs and sizes. An exception to this rule is the survey published by Chen et al. (Chen et al., 2012a), which can be considered as the reference work for any sensor-based activity recognition approach.

2.2 Sensor-Based Activity Recognition

For the upcoming discussions about activity monitoring, modelling and recognition, it is useful to distinguish human behaviours at different levels of granularity. For physical behaviours, the terms "action" and "activity" are commonly used in activity recognition communities. In some cases they are used interchangeably and in other cases they are used to denote behaviours of different complexity and duration. In the latter cases the term "action" is usually referred to as simple ambulatory behaviour executed by a single person and typically lasting for short durations of time. Examples of actions include bending, retrieving a cup from a cupboard, opening a door, putting a teabag into a cup and so forth. On the other hand, the term "activities" here refers to complex behaviours consisting of a sequence of actions and/or interleaving or overlapping actions. They could be performed by a single human or several humans who are required to interact with each other in a constrained manner. They are typically characterized by much longer temporal durations, such as making tea or two persons making meals. For the rest of this dissertation, the second case will be adopted. As such, "actions" will be considered short-time simple executions, while "activities" will be described by a sequence of actions.

The first approaches that implemented the idea of using sensors for activity monitoring and recognition appeared in the late 90s. The *Neural Network House* (Mozer, 1998), in the context of home automation, can be considered a pioneer in this area, alongside with a number of location-based applications aiming to adapt systems to users' whereabouts (Leonhardt and Magee, 1998), (Golding and Lesh, 1999), (Ward et al., 1997). The approach was soon found to be more useful and suitable in the area of ubiquitous and mobile computing – an emerging area in the late 90s, due to its easy deployment. As such, extensive research has been undertaken to investigate the use of sensors in various application scenarios of ubiquitous and mobile computing, leading to considerable work on context-awareness (Schmidt et al., 1999), (Randell and Muller, 2000), (Gellersen et al., 2002), smart appliances (Schmidt and Laerhoven, 2001), (Laerhoven and Aidoo, 2001) and activity recognition (Laerhoven, 2001), (Foerster and Fahrenberg, 2000), (Lee and Mase, 2002). Those initial research works usually made use of wearable sensors, either dedicated sensors attached to human bodies or portable devices like mobile phones, with application to ubiquitous computing scenarios such as providing context-aware mobile devices. Activities being monitored in these researches are mainly physical activities like motion, walking and running. These early works lay a solid foundation for wearable computing and still inspire and influence today's research.

In the early 2000s, a new sensor-based approach that uses sensors attached to objects to monitor human activities appeared. This approach, which was later dubbed as the "dense sensing" approach, performs activity recognition through the inference of user-object interactions (Bao and Intille, 2004), (Patterson et al., 2003). The approach is particularly suitable for dealing with activities that involve a number of objects within an environment, or instrumental Activities of Daily Living (ADL) (Chan et al., 2008), (Nugent and Mulvenna, 2009). Research on this approach has been heavily driven by the intensive research interests and huge research effort on smart home based assisted living, such as the EU's Ambient Assisted Living program. In particular, sensor-based activity recognition can better address sensitive issues in assisted living such as privacy, ethics and obtrusiveness than conventional vision-based approaches. This combination of application needs and technological advantages has stimulated considerable research activities in a global scale, which gave rise to a large number of research projects, where a plethora of impressive works on sensor-based activity recognition have been developed (Kern et al., 2003), (Mantyjarvi, 2001), (Philipose and Fishkin, 2004), (Patterson and Fox, 2005), (Buettner and Prasad, 2009), (Wren and Tapia, 2006), (Gu et al., 2009), (Patterson et al., 2003), (Liao et al., 2007a).

While substantial research has been undertaken, and significant progress has been made, the two main approaches, wearable sensors based and dense sensing based activity recognition are currently still focuses of study. The former is mainly driven by the ever-popular pervasive and mobile computing while the latter is predominantly driven by smart environment applications such as ambient assisted living. Interests in various novel applications are still increasing and application domains are rapidly expanding.

2.3 Sensor and Activity Monitoring

Thanks to the rapid development in electronics, a wide range of sensors, including contact sensors, RFID, accelerometers, audio and motion detectors, to name but a few, are available for activity monitoring. These sensors are different in types, purposes, output signals, underpinning theoretical principles and technical infrastructure. However, they can be classified into two main categories in terms of the way they are deployed in activity monitoring applications. These are wearable sensors and dense sensors, and are described in detail in the following sections.

2.3.1 Wearable sensor-based activity monitoring

Wearable sensors generally refer to sensors that are positioned directly or indirectly on human body. These kinds of sensors can be worn by humans, so they are named wearable sensors. They generate signals when the user performs actions and activities. As a result, they can monitor features that are descriptive of the person's physiological state or movement. Wearable sensors can be embedded into clothes, eyeglasses, belts, shoes, wristwatches, mobile devices or positioned directly on the body. They can be used to collect information such as body position and movement, pulse, and skin temperature. Researchers have found that different types of sensor information are effective for classifying different types of activities. As wearable sensors are not directly related to this dissertation, some references are shown for further reading, classified by the sensors used:

1. Inertial measurement units: those sensors are composed by accelerometers and gyroscopes and are probably the most frequently used wearable sensor for activity monitoring. Inertial measurement units provide information about acceleration and speed of the units while moving. In consequence, they are appropriate to monitor human movements and actions and activities related to body motion, such as physical exercise. Some examples of the usage of inertial measurement units are provided in (Bao and Intille, 2004), (Lukowicz et al., 2004), (Lee and Mase, 2002) and (Mantyjarvi, 2001).

2. GPS sensors: specially used for monitoring outdoor location-based activities, such as in (Patterson et al., 2003), (Ashbrook and Starner, 2003) and (Liao et al., 2007b).

3. Biosensors: to monitor activities through vital signs, e.g. (Sung, M., DeVaul, R., Jimenez, S., Gips, J., Pentland, 2004), (Harms et al., 2008) and (Finni et al., 2007).

Wearable sensor-based activity monitoring suffers from limitations. Most wearable sensors need to run continuously and be operated hands-free. This may have difficulties in real world application scenarios. Practical issues include the acceptability or willingness to use wearable sensors and the viability and ability to wear them. Technical issues include the size, ease of use, battery life and effectiveness of the approach in real-world scenarios. A way to overcome such problems is to make use of existing gadgets that have already been carried in a daily basis like smartphones as intelligent sensors for activity monitoring, recognition and assistance. This practice has been in place for a while (Gellersen et al., 2002), (Schmidt and Laerhoven, 2001) and is expected to gain large-scale uptake given the latest development and affordability of such palm-held electronic devices. Recently, a new and promising class of wearable sensors have appeared in the market, namely the Google Glasses[1] (see Figure 2.2). Those special glasses include a camera, a small eye-screen, headphones and a computing unit. It is expected that the irruption of such glasses will bring to market more similar options. The potential of these kinds of devices for activity monitoring and recognition is still to be explored.

Obviously wearable sensors are not suitable for monitoring activities that involve complex physical motions and/or multiple interactions with the environment. In some cases, sensor observations from wearable sensors alone are not sufficient to differentiate activities involving simple physical movements (e.g., making tea and making coffee). As a result, dense sensing based activity monitoring has emerged, which is described below.

Figure 2.2: The Google Glass wearable, with its processing unit, camera, headphones and screen.

2.3.2 Dense sensing-based activity monitoring

Dense sensing-based activity monitoring is based on attaching sensors to objects that a user manipulates while performing activities. Hence activity monitoring is

[1]https://www.google.com/glass/start/

carried out by detecting user-object interactions. The approach is based on real-world observations that activities are characterised by the objects that are manipulated during their performance. A simple indication of an object being used can often provide powerful clues about the activity being undertaken. As such it is assumed that activities can be recognised from sensor data that monitors human interactions with objects in the environment. By dense sensing, the way and scale with which sensors are used is characterised. Using dense sensing a large number of sensors, normally low-cost low-power and miniaturised, are deployed in a range of objects or locations within an environment for the purpose of monitoring movement and behaviour. Following the dense sensing paradigm, sensors are moved from human bodies to human-populated environments.

The dense sensing paradigm has been widely used for creating ambient intelligent applications such as smart homes, because sensors are embedded within environments. The pre-eminence of dense sensing in ambient assisted living (AAL), via the smart home paradigm, can be seen in (Chan et al., 2008), (Nugent and Mulvenna, 2009) or (Helal et al., 2005). Sensors in a smart home can monitor an inhabitant's movements and environmental events so that activities can be inferred based on the sensor observations, thus providing just-in-time context-aware assistance. For instance, a switch sensor in the bed can strongly suggest sleeping or relaxing, and pressure mat sensors can be used for tracking the movement and position of people within the environment.

Since the introduction of the idea in early 2000s (Bao and Intille, 2004), (Patterson et al., 2003), extensive research has been undertaken to investigate the applicability of the approach in terms of sensor types, modalities and applications. For example, Tapia et al. (Tapia et al., 2004) use environmental state-change sensors to collect information about interaction with objects and recognise activities that are of interest to medical professionals such as toileting, bathing, and grooming. Wilson et al. (Wilson and Atkeson, 2005) use four kinds of anonymous and binary sensors, motion detectors, break-beam sensors, pressure mats, and contact switches for simultaneous tracking and activity recognition. Wren et al. (Wren and Tapia, 2006) employ networks of passive infrared motion sensors to detect presence and movement of heat sources. With this captured data they can recognise low-level activities such as walking, loitering, and turning, as well as mid-level activities such as visiting and meeting. Srivastava et al. (Srivastava et al., 2001) exploit wireless sensor network to develop smart learning environment for young children. Hollosi et al. (Hollosi and Schroder, 2010) use voice detection techniques to perform acoustic event classification for monitoring in Smart Homes. Simple object sensors are adopted in (Aipperspach et al., 2006).

Given the abundance of different types and modalities of sensors, sensors have been used in different ways and combinations for dense sensing activity monitoring in many application scenarios. It is impossible to claim that one sensor deployment for a specific application scenario is superior to the other. The suitability and performance is usually down to the nature of the type of activities being assessed and the characteristics of the concrete applications.

Generally speaking, wearable sensor-based activity monitoring receives more attention in mobile computing while dense sensing is more suitable for intelligent environment enabled applications. It is worth pointing out that wearable sensors and dense sensing are not mutually exclusive. In some applications they have to work together. For example, RFID (Radio Frequency Identification) based activity monitoring requires that objects are instrumented with tags and users wear an RFID reader affixed to a glove or a bracelet. Philipose and Fishkin (Philipose and Fishkin, 2004), (Fishkin et al., 2005) developed two devices, iGlove and iBracelet, working as wearable RFID readers that detect when users interact with unobtrusively tagged objects. Patterson et al. (Patterson and Fox, 2005) performed fine-grained activity recognition (i.e., not just recognising that a person is cooking but determining what they are cooking) by aggregating abstract object usage. Hodges et al. (Hodges and Pollack, 2007) proposed to identify individuals from their behaviour based on their interaction with the objects they use in performing daily activities. Buettner et al. (Buettner and Prasad, 2009) recognise indoor daily activities by using an RFID sensor network. In most cases wearable sensors and dense sensing are complementary and can be used in combination in order to yield optimal recognition results. For example, Gu et al. (Gu et al., 2009) combine wearable sensors and object sensors for collecting multimodal sensor information. Through a pattern-based method, they recognise sequential, interleaved and concurrent activities.

Dense sensing activity monitoring has also some drawbacks. One of the most obvious ones is the need to install a lot of sensors in human populated environments, which may involve considerable infrastructure changes (install pressure mats or movement sensors), or the problem of adding sensors to any new object introduced in the environment (attach contact sensors to glasses bought in the supermarket). The practical feasibility of the approach is probably linked to introducing sensors in the production steps of objects and building phases of the environments. Another drawback is the simple information provided by the sensors, which cannot be enough if very detailed activities have to be recognised. A contact sensor may indicate an interaction between a human and an object, but it cannot provide any information about how the object is being used by the human.

2.4 Data-Driven Approaches

In terms of the techniques and methods used for activity modelling and recognition, abstracting from specific activity monitoring systems, data-driven activity modelling is one of the most successful approaches. Data-driven activity modelling can be generally classified into two main categories: generative and discriminative. The generative approach attempts to build a complete description of the input or data space, usually with a probabilistic model such as a Bayesian Network. In contrast, the discriminative approach only models the mapping from inputs (data) to outputs (activity labels). Discriminative approaches include many heuristic (rule-based) approaches, neural networks, conditional random fields and linear or non-linear discriminative learning (e.g. support vector machines). Results using each of those methods are covered in the following.

2.4.1 Generative modelling

The simplest possible generative approach is the Naïve Bayes classifier, which has been used with promising results for activity recognition (Bao and Intille, 2004), (Brdiczka et al., 2007), (Cook and Schmitter-Edgecombe, 2009), (Tapia et al., 2004), (van Kasteren and Krose, 2007), (Maurer and Rowe, 2006). Naïve Bayes classifiers model all observations (e.g. sensor readings) as arising from a common causal source: the activity, as given by a discrete label. The dependence of observations on activity labels is modelled as a probabilistic function that can be used to identify the most likely activity given a set of observations. Despite the fact that these classifiers assume conditional independence of the features, the classifiers yield good accuracy when large amounts of sample data are provided. Nevertheless, Naïve Bayes classifiers do not explicitly model any temporal information, usually considered important in activity recognition.

The Hidden Markov Model (HMM) is probably the most popular generative approach that includes temporal information. A HMM is a probabilistic model with a particular structure that makes it easy to learn from data, to interpret the data once a model is learnt, and is both easy and efficient to implement. It consists of a set of hidden (latent) states coupled in a stochastic Markov chain, such that the distribution over states at some time depends only on the values of states at a finite number of preceding times. The hidden states then probabilistically generate observations through a stochastic process. HMMs made their impact initially through use in the speech recognition literature, where latent states correspond to phoneme labels, and observations are features extracted from audio data. HMMs have more recently been adopted as a model of choice in computer vision for modelling sequential (video) data (see (Gavrila, 1999), (Moeslund et al., 2006) for surveys and (Galata et al., 1999), (Starner, 1995) for early examples). HMMs use a Markov chain over a discrete set of states. A closely relative of the HMM uses continuous states, a model usually referred to as a linear dynamical system (LDS). State estimation in LDSs is better known as a Kalman filter. LDSs have been used with inputs from a variety of sensors for physiological condition monitoring (Quinn, 2009) in which a method is also introduced to deal with unmodelled variations in data, one of the major shortcomings of the generative approach.

HMMs form the basis of statistical temporal models. They are, in fact, a special case of the more general Dynamic Bayesian Networks (DBNs), which are Bayesian networks in which a discrete time index is explicitly represented. Inference and learning in DBNs is simply an application of network propagation in Bayesian networks. DBNs usually make a Markovian assumption, but explicitly represent conditional independence in the variables, allowing for more efficient and accurate inference and learning. A well known early use of DBNs for activity monitoring was in the *Lumiére* project, where a Microsoft Windows user's need for assistance was modelled based on their activities on the screen (Horvitz et al., 1998).

A simple DBN extension of HMMs is the coupled HMM for recognition of simultaneous human actions (e.g. pedestrian motions (Brand et al., 1997)). Coupled Hidden Markov Models (CHMMs) have two Markovian chains, each modelling a different stream of data, with a coupling between them to model their interdependence. Oliver et al. (Oliver et al., 2004) learn a multi-layer model of office activity to choose actions for a computational agent. The model uses multimodal inputs, making only very slight use of computer vision. The Assisted Cognition project (Kautz et al., 2002) has made use of DBNs, in particular for Opportunity Knocks (Liao et al., 2007b), a system designed to provide directional guidance to a user navigating through a city. This system uses a three level hierarchical Markov model represented as a DBN to infer a user's activities from GPS sensor readings. Movement patterns, based on the GPS localization signals, are translated into a probabilistic model using unsupervised learning. From the model and the user's current location, future destinations and the associated mode of transportation can be predicted. Based on the prediction, the system has the ability to prompt the user if an error in route is detected.

Wilson and Atkeson use DBNs to simultaneously track persons and model their activities from a variety of simple sensors (motion detectors, pressure sensors, switches and so on) (Wilson and Atkeson, 2005). DBNs were also used in the iSTRETCH system, a haptic robotic device to assist a person with stroke rehabilitation (Kan et al., 2011). The DBN models the person's current behaviours, their current abilities, and some aspects of their emotional state (e.g. their responsiveness, learning rate and fatigue level). The person's behaviours correspond to how long they take for each exercise, what type of control they exhibit and whether they compensate. These behaviours are inferred from sensors on the device and in the person's chair.

Even though they are simple and popular, HMMs and DBNs have some limitations. A HMM is incapable of capturing long-range or transitive dependencies of the observations due to its very strict independence assumptions (on the observations). Furthermore, without significant training, a HMM may not be able to recognise all of the possible observation sequences that can be consistent with a particular activity.

2.4.2 Discriminative modelling

A drawback of the generative approach is that enough data must be available to learn the complete probabilistic representations that are required. In this section, an alternative approach for modelling is discussed, in which the focus is on solving the classification problem, rather than on the representation problem. The complete data description of a generative model induces a classification boundary, which can be seen by considering every possible observation and applying the classification rule using inference. The boundary is thus implicit in a generative model, but a lot of work is necessary to describe all the data to obtain it. A discriminative approach, on the other hand, considers this boundary to be the primary objective.

Perhaps the simplest discriminative approach is Nearest Neighbour (NN), in which a novel sequence of observations is compared to a set of template sequences in a training set, and the most closely matching sequences in the training set vote for their activity labels. This simple approach can often provide very good results. Bao and Intille investigated this method along with numerous other base-level classifiers for the recognition of activities from accelerometer data (Bao and Intille, 2004). They found that the simple NN approach is outperformed by decision trees, a related method, where the training data is partitioned into subsets according to activity labels and a set of rules based on features of the training data. The rules can then be used to identify the partition (and hence the activity label) corresponding to a new data sample. Maurer et al. (Maurer and Rowe, 2006), employed decision trees to learn logical descriptions of activities from complex sensor readings from a wearable device (the eWatch). The decision tree approach offers the advantage of generating rules that are understandable by the user, but it is often brittle when high precision numeric data is collected. Stikic and Schiele use a clustering method in which activities are considered as a “bag of features” to learn template models of activities from data with only sparse labels (Stikic and Schiele, 2009).

For the classification problem, the points closest to the boundary are the ones that give more information about the boundary itself. Points lying far apart the boundary should not be taken into account then. There are several discriminative approaches that effectively take advantage of this fact. The challenge is therefore to find these “hard” data points (the ones closest to the boundary). These data points will be known as the “support vectors”, and actually define the boundary. A support vector machine (SVM) is a machine learning technique to find these support vectors automatically. A recent example of an SVM in use for activity modelling is presented by Brdiczka et al. (Brdiczka, 2009) where a model of situations is learnt automatically from data by first learning roles of various entities using SVMs and labelled training data, then using unsupervised clustering to build ’situations’ or relations between entities, which are then labelled and further refined by end users. The key idea in this work is to use a cognitive model (situation model) based on cognitive theory motivated by models of human perception of behaviour in an environment. The CareMedia project (Chen et al., 2005) also uses an SVM to locate and recognise social interactions in a care facility from multiple sensors, including video and audio. The fusion of video and audio allowed 90% recall and 20% precision in identifying interactions including shaking hands, touching, pushing and kicking. The CareMedia project's goals are to monitor and report behaviour assessments in a care home to caregivers and medical professionals.

Ravi et al. also found that SVMs performed consistently well, but also investigated meta-level classifiers that combined the results of multiple base-level classifiers (Ravi et al., 2005). Features extracted from worn accelerometers are extracted and classified using five different base-level classifiers (decision tables, decision trees, k-NN, SVM and Naïve Bayes). The meta-level classifiers are generated through a variety of techniques such as boosting, bagging, voting, cascading and stacking. For recognising a set of eight activities including standing, walking, running, going up/down stairs, vacuuming and teeth brushing, they found that a simple voting scheme performed the best for three easier experimental settings, whereas boosted SVM performed best for the most difficult setting (test/training separation across users and days)

In practice, many activities may have non-deterministic natures, where some steps of the activities may be performed in any order, and so are concurrent or interwoven. A conditional random field (CRF) is a more flexible alternative to the HMM that addresses such practical requirements. It is a discriminative and generative probabilistic model that represents the dependence of a hidden variable y on an observed variable x (Sutton et al., 2007). Both HMMs and CRFs are used to find a sequence of hidden states based on observation sequences. Nevertheless, instead of finding a joint probability distribution $p(x, y)$ as the HMM does, a CRF attempts to find only the conditional probability $p(y|x)$. A CRF allows for arbitrary, non-independent relationships among the observation sequences, hence the added flexibility. Another major difference is the relaxation of the independence assumptions, in which the hidden state probabilities may depend on the past and even future observations. A CRF is modelled as an undirected acyclic graph, flexibly capturing any relation between an observation variable and a hidden state. CRFs are applied to the problem of activity recognition in (Vail et al., 2007) where they are compared to HMMs, but only in a simple simulated domain. Liao et al. use hierarchical CRFs for modeling activities based on GPS data (Liao et al., 2007a). Hu and Yang use skip-chain CRFs, an extension in which multiple chains interact in a manner reminiscent of the CHMM, to model concurrent and interleaving goals (Hu and Yang, 2008), a challenging problem for activity recognition. Mahdaviani and Choudhury show how semi-supervised CRFs can be used to learn activity models from wearable sensor data (Mahdaviani and Choudhury, 2008).

2.4.3 Other approaches

There are many approaches in the literature which cannot be clearly classified into discriminative or generative categories, but rather use a combination of both. The Independent Lifestyle Assistant (ILSA) is an example, as it uses a combination of heuristic rules and statistical models of sequential patterns of sensor firings and time intervals to help a person with planning and scheduling (Guralnik and Haigh, 2002). PEAT (the Planning and Execution Assistant and Trainer) is a cognitive assistant that runs on a mobile device, and helps compensate for executive functional impairment. PEAT uses reactive planning to adjust a user's schedule based on their current activities. Activity recognition in PEAT is based on what the user is doing, and on data from sensors on the mobile device. These are fed into a HMM, the outputs of which are combined with the reactive planning engine (Modayil et al., 2008).

Other work has investigated how activities can be modelled with a combination of discriminative and generative approaches (Lester et al., 2005), how common sense models of everyday activities can be built automatically using data mining techniques (Pentney et al., 2008), (Pentney et al., 2007), and how human activities can be analysed through the recognition of object use, rather than the recognition of human behaviour (Wu and Osuntogun, 2007). This latter work uses DBNs to model various activities around the home, and a variety of radio frequency identification (RFID) tags to bootstrap the learning process. Some authors have attempted to compare discriminative and generative models (Bao and Intille, 2004), (Ravi et al., 2005), generally finding the discriminative models yield lower error rates on unseen data, but are less interpretable. Gu et al. use the notion of emerging patterns to look for frequent sensor sequences that can be associated with each activity as an aid for recognition (Gu et al., 2009). Omar et al. present a comparative study of a variety of classification methods for analysing multi-modal sensor data from a smart walker (Omar et al., 2010).

One of the biggest problems of data-driven approaches, regardless their category, is the need of large-scale labelled data bases. This problem arises because activity recognition is posed as a supervised learning problem. However, due to the required effort and time for annotating activity data bases, supervised methods do not scale up well in practice. Annotating activity data imposes a burden on annotators and users and often introduces a source of error in the process. Rashidi and Cook show how to overcome the problem of depending on labelled activity data bases in (Rashidi and Cook, 2011), improving the approach in (Rashidi and Cook, 2013). They use a non-labelled data base, where they extract activity clusters using unsupervised learning techniques. More concretely, they develop a new activity modelling method named COM, which stands for Continuous, varied Order, Multi Threshold. COM not only discovers discontinuous patterns and their variations, but is able to better handle real life data by dealing with different frequencies/sensor problem. All patterns discovered by COM are then used as inputs for a hierarchical agglomerative clustering algorithm. The clustering algorithm aims at grouping those sensor patterns which are similar in terms of their structure. An activity structure is defined as the start time, duration and region or location. Those clusters are finally used to train a boosted HMM, which is shown to be able to recognise discovered activities.

Rashidi and Cook bring a new paradigm to data-driven activity recognition in their work, showing the real possibility of using an unsupervised model and avoid data base annotation problems. However, their approach still suffers some other traditional problems of data-driven approaches. Firstly, even though they do not use labelled data bases, they do not overcome the "cold start" problem, since they have to collect data in order to discover activities and train HMMs. Besides, as other data-driven approaches, they have many difficulties when generalising what they have discovered and learnt from previous users, since every user has his/her own ways of performing activities. The price to pay for avoiding labelled data bases is that the approach cannot set activity granularity and that discovered activities do not have any semantic meaning. The first problem refers to the impossibility of deciding at design phase what kind of activities will be recognised. For example, if a user usually washes dishes after preparing meal, COM will discover this sequence as a typical activity. However, the discovered activity is actually a sequence of two lower-grain activities. The second problem refers to the fact that discovered activities are only sequences of sensors with their location and time information, but without any semantic meaning. A human expert is needed afterwards to interpret discovered activities and label them with semantic tags. Notice that this *a posteriori* labelling step may be difficult since activity granularity is not set.

The other big problem of data-driven approaches is related to activity model generalisation. As activity models are learnt from specific users' behavioural data, the acquired models are personal. Those personal models cannot be generalised and applied to other users. To overcome this problem, some researchers have already worked on transfer learning techniques. A complete survey on transfer learning techniques for activity recognition can be found in (Cook et al., 2013). The idea behind transfer learning is to take advantage of the learnt models whenever a new training process is launched in a new environment and/or for a new user. This way, the new training phase can be boosted, transferring the knowledge acquired in a previous scenario. Information about sensors, specific activities or locations can be very similar and can provide interesting hints to train new models with few data.

Transfer learning scenarios usually distinguish between the source domain and the target domain. The first domain is composed by the environment, sensors, activities and person where initial activity models have been learnt. The second domain refers to the environment, sensors, activities and person where the new models have to be learnt. Notice that there can be big differences between both domains. For instance, sensors may be completely different, some new activities may appear in the target domain or environments may vary slightly. Depending on those differences, learning techniques vary. However, the most influential parameters to define the learning techniques are related to the availability of labelled data in both domains. Following the classification established in (Cook et al., 2013), depending on the availability of labelled data, the following scenarios are distinguished:

- Informed supervised transfer learning: when labelled data is available for both, target and source domains. Those scenarios are solved using inductive learning techniques.
- Uninformed supervised transfer learning: labelled data is only available in the source domain, so transductive learning approaches are applied.
- Informed unsupervised transfer learning: in this case, labelled data is available only in the target domain. Inductive learning techniques are used for those kinds of scenarios.
- Uninformed unsupervised transfer learning: there is no labelled data available, neither in the target nor in the source domain. Unsupervised learning techniques have to be used to work in those scenarios.

Transfer learning can mitigate the problem of model generalisation, and thus, reusability, but it cannot solve those problems completely. Notice that whenever a new domain is approached, even using transfer learning techniques, a new training phase has to be carried out. The merit of transfer learning is to use previously learnt models to enhance and accelerate new training processes.

2.4.4 Summary of data-driven approaches

Data-driven approaches have been extensively used for activity recognition. Their advantages and disadvantages are very well known by the scientific community (a summary and comparison to knowledge-driven approaches can be found in Table 2.1):

Advantages

- Uncertainty modelling: as probabilistic and statistical activity modelling is used, data-driven approaches are very good modelling sensor uncertainty.

As sensors do not provide certain information and are exposed to failures, modelling uncertainty is a very important feature for real-world deployments.

- Temporal information modelling: modelling approaches such as DBNs, HMMs and similar implicitly model temporal information of activities, such as time lapses between two sensor readings, activity duration or activity start time. Time information can be learnt from data in a natural way.

- Introduce heuristics: discriminative approaches make possible introducing heuristics about activity models. Heuristics can be used to insert basic knowledge about activities and avoid making activity models totally dependent on data.

- Dynamic activity models: as learning is a continuous process, activity models evolve as the user changes its habits. As such, activity models are dynamic.

- Personal activity models: learning activity models directly over user data, makes activity models personalised. This means that activity models capture the particular way of performing an activity by a user.

Disadvantages

- "Cold start" problem: data is needed to model activities. The dependency with data arises the "cold start" problem, since data-driven techniques cannot work immediately after deployment. A data collecting and activity model training process is required for every user.

- Lack of reusability: as activity models are directly learnt from concrete user data, they cannot generally be used for other users. The main problem is that data-driven approaches cannot build generic activity models, only personal activity models. These problems are mitigated, but not solved, by transfer learning techniques, as shown in Section 2.4.3.

- Dataset annotation problems: if annotated datasets are needed (supervised learning approaches), scalability becomes a real problem, since the effort required for annotation is huge and annotation methods are prone to errors. If annotated datasets are not used (unsupervised approach), activity granularity and activity semantics is lost.

2.5 Knowledge-Driven Approaches

It has been observed that for most activities the list of objects required for a particular activity is limited and functionally similar. Knowledge-driven activity modelling is motivated by those observations. The idea is that even if the activity can be performed in different ways the number and type of involved objects do not vary significantly. For example, it is common sense that the activity “make coffee” consists of a sequence of actions involving a coffee pot, hot water, a cup, coffee, sugar and milk; the activity “brush teeth” contains actions involving a toothbrush, toothpaste, water tap, cup and towel. On the other hand, as humans have different life styles, habits or abilities, they may perform various activities in different ways. For instance, one may like strong white coffee and another may prefer a special brand of coffee. Even for the same type of activity (e.g., making white coffee), different individuals may use different items (e.g., skimmed milk or whole milk) and in different orders (e.g., adding milk first and then sugar, or vice versa). Such domain-dependent activity-specific prior knowledge provides valuable insights into how activities can be constructed in general and how they can be performed by individuals in specific situations.

Similarly, knowledge-driven activity recognition is founded upon the observations that most activities, in particular, routine activities of daily living and working, take place in a relatively specific circumstance of time, location and space. The space is usually populated with events and entities pertaining to the activities, forming a specific environment for specific purposes. For example, brushing teeth is normally undertaken twice a day in a bathroom in the morning and before going to bed and involves the use of toothpaste and a toothbrush; meals are made in a kitchen with a cooker roughly three times a day. The implicit relationships between activities, related temporal and spatial context and the entities involved (objects and people) provide a diversity of hints and heuristics for inferring activities.

Knowledge-driven activity modelling and recognition intends to make use of rich domain knowledge and heuristics for activity modelling and pattern recognition. The rationale is to use various methods, in particular, knowledge engineering methodologies and techniques, to acquire domain knowledge. The captured knowledge can then be encoded in various reusable knowledge structures, including activity models for holding heuristics and prior knowledge in performing activities, and context models for holding relationships between activities, objects and temporal and spatial contexts. Comparing to data-driven activity modelling that learns models from large-scale datasets and recognises activities through data intensive processing methods, knowledge-driven activity modelling avoids a number of problems, including the requirement for large amounts of observation data, the inflexibility that arises when each activity model needs to be computationally learnt, and the lack of reusability that results when one person’s activity model is different from another's.

Knowledge structures can be modelled and represented in different forms, such as schemas, rules or networks. This will decide the way and the extent to which knowledge is used for further processing such as activity recognition, prediction and assistance. In terms of the manner in which domain knowledge is captured, represented and used, knowledge-driven approaches to activity modelling and recognition can be roughly classified into three main categories as presented in the following sections.

2.5.1 Mining-based approach

The objective of a mining-based approach is to create activity models by mining existing activity knowledge from publicly available sources. More specifically, given a set of activities, the approach seeks to discover from the text corpuses a set of objects used for each activity and extract object usage information to derive their associated usage probabilities. The approach essentially views an activity model as a probabilistic translation between activity names (e.g., "make coffee") and the names of involved objects (e.g., "mug", "milk"). As the correlations between activities and their objects are common-sense prior knowledge (e.g., most of us know how to carry out daily activities), such domain knowledge can be obtained from various sources such as how-tos (e.g., those at *ehow.com* or *wikihow.com*), recipes (e.g., from *epicurious.com*), training manuals, experimental protocols, and facility/device user manuals.

A mining-based approach consists of a sequence of distinct tasks. Firstly it needs to identify activities of concern and relevant sources that describe these activities. Secondly, it uses various methods, predominantly information retrieval and analysis techniques, to retrieve activity definitions from specific sources and extract phrases that describe the objects used during the performance of the activity. Then algorithms, predominantly probabilistic and statistic analysis methods such as co-occurrences and association are used to estimate the object-usage probabilities. Finally, the mined object and usage information is used to create activity models such as a HMM that can be used further for activity recognition.

Mining-based activity modelling was initially investigated by researchers from Intel Research (Wu and Osuntogun, 2007), (Wyatt et al., 2005). Perkowitz et al. (Perkowitz et al., 2004) proposed the idea of mining the Web for large-scale activity modelling. They used the QTag tagger to tag each word in a sentence with its part of speech (POS) and a customized regular expression extractor to extract objects used in an activity. They then used the Google Conditional Probabilities (GCP) APIs to determine automatically the probability values of object usage. The mined object and their usage information are then used to construct DBN models through Sequential Monte Carlo (SMC) approximation. They mined the website *ehow.com* for roughly 2300 directions on performing domestic tasks (from “boiling water in the microwave” to “change your air filter”), and the website *ffts.com* and *epicurious.com* for a further 400 and 18,600 recipes respectively, generating a total 21,300 activity models. Using the DBN activity models they have performed activity recognition for a combination of real user data and synthetic data. While initial evaluation results were positive, the drawback was that there are no mechanisms to guarantee the mined models capturing completely the sequence probabilities and the idiosyncrasy of certain activities. The inability to capture such intrinsic characteristics may limit the model’s accuracy in real deployments.

Wyatt et al. (Wyatt et al., 2005) followed Perkowitz’s approach by mining the Web to create DBN activity models. However, this group extended the work in three aspects, aiming to address the idiosyncrasies and to improve model accuracy. To cover the wide variety of activity definition sources, they mined the Web in a more discriminative way in a wider scope. They did this by building a specialised genre classifier trained and tested with a large number of labelled Web pages. To enhance model applicability, they used the mined models as base activity models and then exploited the Viterbi Algorithm and Maximum Likelihood to learn customized activity parameters from unsegmented, unlabelled sensor data. In a bid to improve activity recognition accuracy they also presented a bootstrap method that produced labelled segmentations automatically. Then they used the Kullback-Leibler (KL) divergence to compute activity similarity.

A difficulty in connecting mined activities with tagged objects (Perkowitz et al., 2004), (Wyatt et al., 2005) is that the activity models may refer to objects synonymously. For example, both a “mug” and “cup” can be used for making tea; both a “skillet” and “frying pan” may be used for making pasta. This leads to a situation that one activity may have different models with each having the same activity name but different object terms. To address this, Tapia et al. (Tapia et al., 2006) proposed to extract collections of synonymous words for the functionally-similar objects automatically from WordNet, an online lexical reference system for the English language. The set of terms for similar objects is structured and represented in a hierarchical form known as the object ontology. With the similarity measure provided by the ontology, an activity model will not only cover a fixed number of object terms but also any other object terms that are in the same class in the ontology.

Another shortcoming of early work in the area (Perkowitz et al., 2004), (Wyatt et al., 2005) is that the segmentation is carried out in sequential order based on the duration of an activity. As the duration of performing a specific activity may vary substantially from one to another, this may give rise to applicability issues. In addition, in sequential segmentation one error in one segment may affect the segmentations of the subsequent traces. To tackle this, Palmes et al. (Palmes et al., 2010) proposed an alternate method for activity segmentation and recognition. Instead of relying on the order of object use, they exploited the discriminative trait of the usage frequency of objects in different activities. They constructed activity models by mining the Web and extracting relevant objects based on their weights. The weights are then utilised to recognise and segment an activity trace containing a sequence of objects used in a number of consecutive and non-interleaving activities. To do this, they proposed an activity recognition algorithm, KeyExtract, which uses the list of discriminatory key objects from all activities to identify the activities present in a trace. They further proposed two heuristic segmentation algorithms, MaxGap and MaxGain, to detect the boundary between each pair of activities identified by KeyExtract. Boundary detection is based on the calculation, aggregation, and comparison of the relative weights of all objects sandwiched in any two key objects representing adjacent activities in a trace. Though the mining based approach has a number of challenges relating to information retrieval, relation identification and the disambiguation of term meaning, nevertheless, it provides a feasible alternative to model large amount of activities. Initial research has demonstrated the approach is promising.

Mining-based approaches are similar to data-driven approaches in that they all adopt probabilistic or statistical activity modelling and recognition. But they are different from each other in the way that the parameters of the activity models are decided. The mining-based approaches make use of publicly available data sources avoiding the "cold start" problem. Nevertheless they are weak in dealing with idiosyncrasies of activities. On the other hand, data-driven approaches have the strength of generating personalised activity models, but they suffer from issues such as "cold start" and model reusability for different users.

2.5.2 **Logic-based approaches**

A logic-based approach views an activity as a knowledge model that can be formally specified using various logical formalisms. From this perspective, activity modelling is equivalent to knowledge modelling and representation. As such, systematic knowledge engineering methodologies and techniques are used for domain knowledge acquisition and formal construction of activity structures. Knowledge representation formalisms or languages are used to represent these knowledge models and concrete knowledge instances, thus enabling inference and reasoning. In this way, activity recognition and other advanced application features such as prediction and explanation can be mapped to knowledge-based inference such as deduction, induction and abduction.

Logic-based approaches are composed of a number of distinct tasks. Even though each task can be undertaken in different ways the role of each task is specific and unique. Normally the first step is to carry out knowledge acquisition, which involves eliciting knowledge from various knowledge sources such as domain experts and activity manuals. The second step is to use various knowledge modelling techniques and tools to build reusable activity structures. This will be followed by a domain formalization process in which all entities, events and temporal and spatial states pertaining to activities, along with axioms and rules, are formally specified and represented using representation formalism. This process usually generates the domain theory. The following step will be the development of a reasoning engine in terms of knowledge representation formalisms to support inference. In addition, a number of supportive system components will be developed, which are responsible for aggregating and transforming sensor data into logical terms and formula. With all functional components in place, activity recognition proceeds by passing the logical representation of sensor data onto the reasoning engine. The engine performs logical reasoning, e.g., deduction, abduction or induction, against the domain theory. The reasoning will extract a minimal set of covering models of interpretation from the activity models based on a set of observed actions, which could semantically explain the observations.

There exist a number of logical modelling methods and reasoning algorithms in terms of logical theories and representation formalisms. One thread of work is to map activity recognition to the plan recognition problem in the well studied artificial intelligence field (Carberry, 2001). The problem of plan recognition can be stated in simple terms as: given a sequence of actions performed by an actor, how to infer the goal pursued by the actor and also to organise the action sequence in terms of a plan structure. Kautz et al. (Kautz, 1991) adopted first-order axioms to build a library of hierarchical plans. They proposed a set of hypotheses such as exhaustiveness, disjointedness and minimum cardinality to extract a minimal covering model of interpretation from the hierarchy, based on a set of observed actions. Wobke (Wobcke, 2002) extends Kautz's work using situation theory to address the different probabilities of inferred plans by defining a partial order relation between plans in terms of levels of plausibility. Bouchard et al. (Bouchard et al., 2006) borrow the idea of plan recognition and apply it to activity recognition. They use action Description Logic (DL) to formalize actions and entities and variable states in a smart home to create a domain theory. They model a plan as a sequence of actions and represent it as a lattice structure, which, together with the domain theory, provides an interpretation model for activity recognition. As such, given a sequence of action observations, activity recognition amounts to reasoning against the interpretation model to classify the actions through a lattice structure. It was claimed that the proposed DL models can organize the result of the recognition process into a structured interpretation model in the form of lattice, rather than a simple disjunction of possible plans without any classification. This minimises the uncertainty related to the observed actor's activity by bounding the plausible plans set.

Another thread of work is to adopt the Event Calculus (EC) (Shanahan, 1997) formalism, a highly developed logical theory of actions, for activity recognition and assistance. The EC formalizes a domain using fluents, events and predicates. Fluents are any properties of the domain that can change over time. Events are the fundamental instrument of change. All changes to a domain are the result of named events. Predicates define relations between events and fluents that specify what happens when and which fluents hold at what times. Predicates also describe the initial situation and the effects of events. Chen et al. (Chen and Nugent, 2008) proposed an EC-based framework in which sensor activations are modelled as events, and object states as properties. In addition, they developed a set of high-level logical constructors to model compound activities, i.e. the activities consisting of a number of sequential and/or parallel events. In the framework, an activity trace is simply a sequence of events that happen at different time points. Activity recognition is mapped to deductive reasoning tasks, e.g., temporal projection or explanation, and activity assistance or hazard prevention is mapped to abductive reasoning tasks. The major strength of this work is its capability to address temporal reasoning and the use of compound events to handle uncertainty and flexibility of activity modelling.

Logic-based approaches are totally different from data-driven approaches in the way activities are modelled and the mechanisms activities are recognised. They do not require pre-existing large-scale datasets, and activity modelling and recognition is semantically clear and elegant in computational reasoning. It is easy to incorporate domain knowledge and heuristics for activity models and data fusion. The weakness of logical approaches is their inability or inherent infeasibility to represent fuzziness and uncertainty even though there are recent works trying to integrate fuzzy logics into the logical approaches. Another drawback is that logical activity models are viewed as one-size-fits-all, inflexible for adaptation to different users' activity habits.

2.5.3 **Ontology-based approaches**

Using ontologies for activity recognition is a recent endeavour and has gained growing interest. In the vision-based activity recognition community, researchers have realised that symbolic activity definitions based on manual specification of a set of rules suffer from limitations in their applicability, because the definitions are only deployable to the scenarios for which they have been designed. There is a need for a commonly agreed explicit representation of activity definitions or an ontology. Such ontological activity models are independent of algorithmic choices, thus facilitating portability, interoperability and reuse and sharing of both underlying technologies and systems. Chen et al. (Chen et al., 2004) propose activity ontologies for analysing social interaction in nursing homes, Hakeem et al. (Hakeem and Shah, 2004) for the classification of meeting videos, and Georis et al. (Georis et al.,

2004) for activities in a bank monitoring setting. To consolidate these efforts and to build a common knowledge base of domain ontologies, a collaborative effort has been made to define ontologies for six major domains of video surveillance. This has led to a video event ontology (Nevatia et al., 2004) and the corresponding representation language (Francois et al., 2005). For instance, Akdemir (Akdemir et al., 2008) used the video event ontologies for activity recognition in both bank and car park monitoring scenarios. In principle these studies use ontologies to provide common terms as building primitives for activity definitions. Activity recognition is performed using individually preferred algorithms, such as rule-based systems (Hakeem and Shah, 2004) and finite-state machines (Akdemir et al., 2008).

In the dense sensing-based activity recognition community, ontologies have been utilised to construct reliable activity models. Such models are able to match different object names with a term in an ontology which is related to a particular activity. For example, a Mug sensor event could be substituted by a Cup event in the activity model MakeTea as Mug and Cup can both be used for the MakeTea activity. This is particularly useful to address model incompleteness and multiple representations of terms. Tapia et al. (Tapia et al., 2006) generated a large object ontology based on functional similarity between objects from WordNet, which can complete mined activity models from the Web with similar objects. Yamada et al. (Yamada et al., 2007) use ontologies to represent objects in an activity space. By exploiting semantic relationships between things, the reported approach can automatically detect possible activities even given a variety of object characteristics including multiple representation and variability. Similar to vision-based activity recognition, these studies mainly use ontologies to provide activity descriptors for activity definitions. Activity recognition can then be performed based on probabilistic and/or statistical reasoning (Tapia et al., 2006), (Yamada et al., 2007).

Ontology-based modelling and representation has been applied to general ambient assisted living. Latfi et al. (Latfi et al., 2007) propose an ontological architecture of a telehealth-based smart home aiming at high-level intelligent applications for elderly persons suffering from loss of cognitive autonomy. Klein et al. (Klein et al., 2007) developed an ontology-centred design approach to create a reliable and scalable ambient middleware. Chen et al. (Chen et al., 2009) pioneered the notion of semantic smart homes in an attempt to leverage the full potential of semantic technologies in the entire lifecycle of assistive living i.e. from data modelling, content generation, activity representation, processing techniques and technologies to assist with the provision and deployment. While these endeavours, together with existing work in both vision- and dense sensing-based activity recognition, provide solid technical underpinnings for ontological data, object, sensor modelling and representation, there is a gap between semantic descriptions of events/objects related to activities and semantic reasoning for activity recognition. Most works use ontologies either as mapping mechanisms for multiple terms of an object (Tapia et al., 2006) or the categorization of terms (Yamada et al., 2007) or a common conceptual template for data integration, interoperability and reuse (Latfi et al., 2007), (Klein et al., 2007), (Chen et al., 2009). Activity ontologies which provide an explicit conceptualization of activities and their interrelationships have only recently emerged and have been used for activity recognition. Chen et al. (Chen and Nugent, 2009), (Chen et al., 2012b) proposed and developed an ontology-based approach to activity recognition. They constructed context and activity ontologies for explicit domain modelling. Sensor activations over a period of time are mapped to individual contextual information and then fused to build a context at any specific time point. They made use of subsumption reasoning to classify the constructed context based on the activity ontologies, thus inferring the ongoing activity. Ye et al. (Ye et al., 2011) developed an upper activity ontology that facilitates the capture of domain knowledge to link the meaning implicit in elementary information to higher-level information that is of interest to applications. Riboni et al. (Riboni and Bettini, 2011b) investigated the use of activity ontologies, in particular, the new feature of rule representation and rule-based reasoning from OWL2, to model, represent and reason complex activities.

Compared with data-driven and mining-based approaches, ontology-based approaches offer several compelling features: firstly, ontological ADL models can capture and encode rich domain knowledge and heuristics in a machine understandable and processable way. This enables knowledge based intelligent processing at a higher degree of automation. Secondly, DL- based descriptive reasoning along a time line can support incremental progressive activity recognition and assistance as an ADL unfolds. The two levels of abstraction in activity modelling, concepts and instances, also allow coarse-grained and fine-grained activity assistance. Thirdly, as the ADL profile of an inhabitant is essentially a set of instances of ADL concepts, it provides an easy and flexible way to capture a user's activity preferences and styles, thus facilitating personalised ADL assistance. Finally, the unified modelling, representation and reasoning for ADL modelling, recognition and assistance makes it natural and straightforward to support the integration and interoperability between contextual information and ADL recognition. This will support systematic coordinated system development by making use of seamless integration and synergy of a wide range of data and technologies.

Compared with logic-based approaches, ontology-based approaches have the same mechanisms for activity modelling and recognition. However, ontology-based approaches are supported by a solid technological infrastructure that has been developed in the semantic Web and ontology-based knowledge engineering communities. Technologies, tools and APIs are available to help carry out each task in the ontology-based approach, e.g., ontology editors for context and activity modelling, web ontology languages for activity representation, semantic repository technologies for large-scale semantic data management and various reasoners for activity inference. This gives ontology-based approaches huge advantage in large-scale adoption, application development and system prototyping.

2.5.4 Summary of knowledge-driven approaches

The advantages and drawbacks of knowledge-driven approaches are summarised in the following listing. A summary and comparison with data-driven approaches can be found in Table 2.1.

Advantages

- No "cold start" problem: knowledge-driven approaches model activities using generic knowledge rather than data, so activity models are built before deployment and the system does not need any training/learning process before beginning to work.
- Interoperability and reusability: specially true for ontology-based approaches, but also for all the other approaches, as activity models are modelled using knowledge engineering techniques and built models are generic and not specific to a concrete user.
- Clear semantics: activity models are semantically clear and can be understood by human beings. This allows interpreting how the system works and developing easier auxiliary systems that work on top of the activity recognition system, such as notification systems, recommender systems and so on.

Disadvantages

- Weak in handling uncertainty: inference and reasoning are usually based on certain facts, rather than uncertain sensor information. There are some approaches that work using fuzzy logics and/or probabilistic reasoning, but they are not fully integrated with modelling techniques yet (Helaoui et al., 2013), (Almeida and López-de-Ipiña, 2012).
- Weak in handling temporal information: inference and reasoning mechanisms used for activity recognition do not usually consider temporal aspects of activities. In order to tackle this limitation, there are already some approaches that, for example, integrate ontological and temporal knowledge modelling formalisms for activity modelling (Okeyo and Chen, 2012).
- Static activity models: knowledge-based activity models are static, since once they are defined, they cannot automatically evolve. This means that

if a user changes its way of performing activities, initially defined activity models will still be used for activity recognition.

2.6 Hybrid Approaches

Analysing carefully the advantages and disadvantages of knowledge- and data-driven approaches, it can be seen that they are complementary at some extent. For instance, while knowledge-driven approaches are weak in handling uncertainty and temporal information, data-driven ones are good at it. On the other hand, while data-driven approaches cannot build reusable generic activity models, knowledge-driven approaches can. This complementarity has recently raised the need to research on *hybrid approaches to activity modelling*. The main idea of hybrid approaches is to fuse data- and knowledge-driven approaches in order to solve the problems of both approaches in a single system.

Tran et al. (Tran and Davis, 2008) and Healaoui et al. (Helaoui et al., 2011) use Markov Logic Networks (MLN) to model temporal relations between actions and activities. The first one uses MLNs to naturally integrate common sense reasoning with uncertain analyses produced by computer vision algorithms for object detection, tracking and movement recognition. The second work is focused on providing activity models for interleaved and concurrent executions using qualitative and quantitative temporal relationships. With a similar objective, Steinhauer et al. (Steinhauer and Chua, 2010) combine HMMs with Allen logic, providing another hybrid approach example. All those approaches encode and use temporal knowledge and rely on automatically extracting these temporal patterns from datasets.

Another hybrid approach example is provided by Riboni et al. (Riboni and Bettini, 2011a). Their approach combines statistical inference techniques with ontological activity modelling and recognition. Statistical inferencing is performed based on raw data retrieved from body-worn sensors (e.g., accelerometers) to predict the most probable activities. Then, symbolic reasoning is applied to refine the results of statistical inferencing by selecting the set of possible activities performed by a user based on his/her current context. By decoupling the use of context information, statistical inferencing becomes more manageable in terms of necessary training data, while symbolic reasoning can more effectively select candidate activities taking into account context-dependent ontological relationships.

Finally, Chen et al. (Chen et al., 2014) present an ontology-based hybrid approach to activity modelling. They combine knowledge-based activities with specially developed learning algorithms to overcome the "cold start" problem, model reusability and incompleteness. The approach uses semantic technologies as a conceptual backbone and technology enablers for ADL modelling, classification and learning. The distinguishable feature of the approach from existing approaches is

	Knowledge-Driven Approaches				**Data-Driven Approaches**	
	Mining-based	Logic-based	Ontology-based		Generative	Discriminative
Model Type	HMM, DBN, SVM, CRF, NN	Plans, lattices, event trees	HMM, DBN, SVM, CRF, NN	Sensor and Activity ontologies	Naïve Bayes, HMM, LDS, DBN	NN, SVM, CRF, Decision tree
Modelling Mechanism	Information retrieval and analysis	Formal knowledge modelling	(un)supervised learning from datasets	Ontological engineering	(un)supervised learning from datasets	(un)supervised learning from datasets
Activity Recognition Method	Generative or discriminative methods	Logical inference (deduction, induction)	Generative or discriminative methods	Semantic reasoning (subsumption, consistency)	Probabilistic classification	Similarity or rule based reasoning
Advantage	No "cold start" problem, Using multiple data sources	No "cold start" problem, clear semantics on modelling & inference	Shared terms, interoperability and reusability	No "cold start" problem, multiple models, clear semantics on modelling & inference, interoperability & reusability	Modelling uncertainty, temporal information	Modelling uncertainty, temporal information, heuristics
Disadvantage	The same problem as DDA	Weak in handling uncertainty and scalability	The same problem as DDA	Weak in handling uncertainty and time	"Cold start" problem, lack of reusability & scalability	"Cold start" problem, lack of reusability & scalability

Table 2.1: The summary and comparison of activity recognition approaches.

that ADL modelling is not a one-off effort, instead, a multi-phase iterative process that interleaves knowledge-based model specifications and data-driven model learning. The process consists of three key phases. In the first phase the initial seed ADL models are created through ontological engineering by leveraging domain knowledge and heuristics, thus solving the "cold start" problem. Ontological activity modelling creates activity models at two levels of abstractions, namely as ontological activity concepts and their instances respectively. Ontological activity concepts represent generic coarse-grained activity models applicable and reusable for all users, thus solving the reusability problem. The seed ADL models are then used in applications for activity recognition at the second phase. In the third phase, the activity classification results from the second phase are analysed to discover new activities and user profiles. These learnt activity patterns are in turn used to update the ADL models, thus solving the incompleteness problem. Once the first phase completes, the remaining two-phase process can be iterated many times to incrementally evolve the ADL models, leading to complete, accurate and up-to-date ADL models. In consequence, this approach is a first step to solve the static nature of activity models in knowledge-driven systems. However, the approach has a limitation: it considers that seed activity models contain all the actions performed by any user to perform an activity and hence, it is limited to learn only descriptive properties. This means that if different users perform the same activity by means of different sets of actions, the system will not be able to learn those distinct actions.

2.7 Summary and Conclusions

A deep review of human activity recognition has been provided in this chapter. In terms of activity monitoring technologies, the work presented in this dissertation can be classified into the dense sensing-based category. In principle, there are no limitations to extend the methods and techniques exposed in this dissertation to other activity monitoring approaches such as wearable sensor- or vision-based categories. But dense sensing paradigm has been chosen because it is the best approach for Intelligent Environments - since there are no privacy issues with users and specific objects can be monitored - and the usage of simple sensors makes sensor processing steps simpler. As research on sensor processing is beyond the scope of this work, dense sensing-based activity monitoring provides a perfect scenario. Notice that the selection of the dense sensing-based activity monitoring scenario is more an implementation decision rather than a theoretical limitation.

In terms of activity modelling, the work presented in this dissertation clearly falls into the hybrid approach. It combines knowledge- and data-driven techniques in order to provide dynamic activity models that combine generic and personalised models. As such, it tries to solve the problem of the hybrid activity modelling approach presented by Chen et al. (Chen et al., 2014), i.e. learning new actions for any user. Solving other problems of knowledge-driven approaches, such as sensor uncertainty or temporal information handling, is out of scope of this dissertation. The concrete position of this dissertation in the two axis taxonomy adopted for activity recognition can be seen in Table 2.2.

		Activity Monitoring Approach		
		Vision-based	Sensor-based	
			Wearable-based	Dense sensing-based
Activity Modelling Approach	Data-Driven			
	Knowledge-Driven			
	Hybrid			X

Table 2.2: The classification of this dissertation in terms of activity modelling and activity monitoring approaches marked with an X.

Finally, this chapter has been mainly focused on single user - single activity scenarios, although some examples of single user - concurrent activities scenarios were also described. This is because the human activity recognition research community has mainly focused its efforts in the single user - single activity scenario, despite recently, single user - concurrent activities scenario has gained more popularity. It is worth to highlight that the work presented in this dissertation has been designed to tackle the single user - single activity scenario.

Simplicity is the shortest path to a solution.

Ward Cunningham

CHAPTER

3

The Approach to Learn Specialised and Complete Activity Models

THIS chapter describes the theoretical foundations of the dissertation and describes the high-level solution designed to learn activity models from user's behavioural data and previous knowledge. The contributions of this dissertation are based on the ontology-based activity modelling approach, where an important limitation has been found: providing complete and generic activity models is generally impossible. To solve this problem, a learning solution is suggested, whose underpinning modules and their relationships are described in detail. The integration of the developments presented in Chapters 4 and 5 is structured around the designed solution in this chapter.

The chapter is divided in four sections: Section 3.1 describes the ontology-based activity modelling approach on which the contributions of this dissertation are built. Section 3.2 provides formal definitions of the concepts and terms that will be used during the following chapters and shows the constraints of the approach to learn activity models. Section 3.3 discusses the adopted solution design, describing the defined architecture, constituent modules and their relationships. And finally Section 3.4 concludes the chapter with a summary and extracted conclusions.

3.1 Ontology-Based Activity Modelling

Ontology-based activity modelling has already been introduced in Chapter 2, concretely in Section 2.5.3. Some of the most relevant research work was presented there. However, the objective of this section is to provide a deep description of one of the presented approaches, which is the basis of the contributions presented in this dissertation.

The activity modelling approach introduced in (Chen et al., 2012b) is central to the real-time activity recognition system for smart homes described in the paper. This approach has been designed for the dense sensing-based activity monitoring scenario and deployed in a smart home research laboratory, obtaining an average activity recognition rate of 94.44%. The reported results suggest that the system performs robustly for the selected scenarios.

Even though the activity recognition process is very interesting, the focus of this section will be on the ontology-based activity modelling approach which is introduced in (Chen et al., 2012b). Roughly speaking, two high-level concepts are modelled in the paper to support activity recognition: (i) Activities of Daily Living (ADL) and (ii) the context in which those ADLs are performed (in the case of the paper, a smart home). To specify conceptual structures and relationships, DL based markup language, i.e. Web Ontology Language (OWL) and Resource Description Framework (RDF)[1] are used. Both languages support inference and reasoning, a feature which plays a key role for activity recognition.

3.1.1 Ontological ADL modelling

Activities of Daily Living, or ADLs, refer to a set of activities that are performed by any person in a daily basis. Some examples for ADLs are making tea, making pasta or washing dishes. When ADLs are carefully analysed, two features can be identified: (i) ADLs can be defined at different levels of granularity, and (ii) ADLs are generic, but each person performs them in different ways. Both features can be properly addressed by the ontological ADL modelling approach.

The first feature, activity granularity, refers to the fact that ADLs form super-class and sub-class relations. For example, making a hot chocolate can be considered a sub-activity of making a hot drink, which can also be classified as a sub-activity of making a drink. As such, ADLs form a tree where leafs model *primitive* ADLs and other nodes refer to *composite* ADLs (see Figure 3.1).

To address different levels of activity granularity, ontological modelling, i.e. the process to explicitly specify key concepts and their properties for a problem domain, is applied. More concretely, ADLs are modelled as ontological concepts

[1]http://www.w3.org/

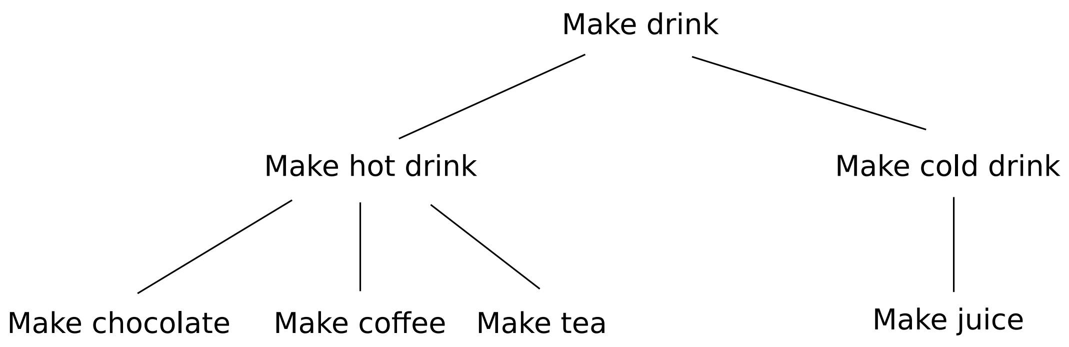


Figure 3.1: An example of ADL granularity represented as a tree.

which are organised in a hierarchical structure in terms of their shared properties to form super-class and sub-class relations. Those relations between primitive and composite ADLs are characterised by “is-a” and “part-of” relationships, properly supported by OWL. For example, as shown in Figure 3.1 making tea is a subclass of making hot drink. The ADL for making tea is represented by the ontological concept of MakeTea, while making hot drink is represented by MakeHotDrink. Properties establish the interrelations between concepts. For instance, *hasDrinkType* is a property of the MakeHotDrink activity that links the DrinkType concept (e.g., tea, coffee, chocolate) to the MakeHotDrink concept. Both, concepts and properties, are modelled using the commonly shared terms in the problem community. The resulting ontologies are essentially knowledge models able to encode and represent domain knowledge and heuristics.

The second feature of ADLs highlights the difference between generic and personalised ADLs. Even though making coffee implies the use of coffee and a container, some users prefer African coffee to Colombian coffee and some users use cups and others mugs. Furthermore, the order of the sequence of objects used to make coffee, vary from user to user, e.g. some users might add sugar before coffee and others add sugar once coffee has been poured to the container. In consequence, generic and personalised ADLs can be distinguished.

1. Generic activity models: defined at the conceptual level as an activity class described by a number of properties. These properties describe the types of objects and their usage to perform an activity. Generic activity models are applicable to all users.

2. Personalised activity models: they capture the specific way of a user to perform an activity and are modelled as instances of a corresponding conceptual activity model, which consists of specific information related to the concrete objects used and the sequential order they are used in.

Activities are defined as ontological concepts and all actions that are required to perform the activity as the properties of the concept. For example, making tea involves taking a cup from the cupboard, putting a teabag into the cup, adding hot water to the cup, then milk and/or sugar. The ontological model of making tea, i.e. MakeTea concept, can be defined by action properties *hasContainer*, *hasTeabag*, *hasHotwater*, *hasMilk* and *hasFlavour* in conjunction with descriptive properties such as activity start time *actStartTime* and duration *actDuration*. Notice that two kinds of properties are defined: *action properties*, which refer to action-based properties and *descriptive properties*, which are used to characterise the manner in which an activity is performed. This distinction is very important, because action properties play a crucial role for activity recognition. Indeed, activities are inferred from action properties which are generated by the sensor activations occurring due to human-object interactions. On the other hand, descriptive properties are not determinant in activity recognition. For instance, making coffee can happen at any time, it can be performed in different sequences and it may take variable amounts of time. As such, descriptive properties are used to model user's activity profiles, which are not used for the activity recognition process.

Hence, action properties define generic ADL models, which are used for activity recognition and are applicable to any user. Descriptive properties capture the different ways of performing generic ADLs, in terms of the objects used, the order in which those objects have been used, activity start time and duration. Personalised ADL models are instances of the corresponding generic ADL model.

In summary, ontological ADL modelling properly addresses different levels of activity granularity and the difference between generic and personalised ADLs, using the tools provided by markup languages such as OWL and RDF.

3.1.2 Ontological context modelling

The ADL modelling scheme described above establishes a clear dependency between the context and an ADL. Indeed, ADLs are described as a set of context events, where context refers to temporal, spatial, event and environmental aspects. Exemplary temporal aspects of an ADL may be the start time and the duration of the activity, which are captured by descriptive properties. Spatial aspects relate to location information and surrounding entities such as rooms, household furniture and appliances. Event contexts capture dynamic state changes of the objects of the environment, for example, state changes of doors, fridge door or a microwave oven. Those events are modelled by action properties. Finally, environmental aspects of the context are composed of environmental information such as temperature, humidity and general weather conditions.

In the dense sensing paradigm, contextual information is captured through various sensors. Each sensor monitors and reflects one facet of a situation. Based on this observation the context modelling is centred on ontological sensor modelling. Sensors are inherently linked to a number of physical and conceptual entities such as objects, locations and states. For example, a contact sensor is attached to a teapot in the second cupboard to the left of the sink in the kitchen. By explicitly capturing and encoding such domain knowledge in a sensor model it is possible to infer the corresponding objects and location from the activation of the sensor. This implies that a user performs an activity in the inferred location with the inferred object.

As most ADLs require the interlinking and fusion of data from multiple, disparate sensor sources in order to infer the high-level activities, it is necessary to aggregate a sequence of sensor activations to generate a situation at a specific time point. The situation formation process can be described as follows. Each sensor has a default state value for its state property, denoting the state of the object to which it is attached. When a sensor is activated, the state property will change. Subsequently the system will translate the state change as an occurrence of a user-object interaction at the specific time. As it is difficult, if not impossible, to monitor what happens at detailed levels after an object is interacted with, it is common practice to interpret a sensor activation as a user-object interaction, ignoring how the object is used and when it is de-activated. As such a user-object interaction is equivalent to an instantaneous sensor activation and can be interpreted as an object being used for performing an activity. In this way, by aggregating individual user-object interactions along a timeline the situation at specific time points can be generated.

Under this context modelling approach, sensor activations are assumed to be produced by user-object interactions, which may denote an action property or a descriptive property, depending on the nature of the object and sensor. For example, a thermometer state change will be captured by a descriptive property such as *hasTemperature(value)*, whereas a contact sensor installed in a cup will generate the action property *hasContainer(cup)*.

3.1.3 Final remarks about ontology-based activity modelling

A common framework to model ADLs and contextual information is provided by the ontology-based activity modelling approach described in (Chen et al., 2012b). Ontology-based activity modelling provides semantically clear, structured and reusable models. Furthermore, it offers a unified framework to combine generic models that can be applied to any user with personal models. By means of ontological modelling some traditional problems of data-driven approaches can be avoided, such as the manual class labelling, pre-processing and training processes. In addition, ontologies allow software agents to interpret data and reason against ontological contexts, thus enhancing the capabilities of automated data interpretation and inference.

However, the main problem of this modelling approach is related to model incompleteness. Obtaining complete models from domain knowledge for every person is not generally possible. An ADL model is composed by action and descriptive properties, so model incompleteness refers to the impossibility of completely defining all the action and descriptive properties of an ADL. In the case of descriptive properties, model incompleteness is obvious, since those properties refer to personalised ways of performing activities. Chen et al. propose in (Chen et al., 2014) a hybrid activity modelling approach to learn descriptive properties from user generated data. But they assume that ADL models in terms of action properties are complete. This assumption does not generally hold, because even though there are certain actions that every user performs for a given activity, there might be some other actions that cannot be known in the modelling step.

Hence, this dissertation tackles the problem of learning unmodelled action properties for activity models. By means of learning new actions through incremental data-driven learning techniques, the static and generic nature of knowledge-based activity models is also tackled. The contributions of this dissertation allow dynamic activity models which evolve in time and support both, generic and personalised models.

3.2 Definitions and Constraints

For the rest of the dissertation the following terms and concepts will be used according to the definitions provided in this section:

Definition 1 (Sensor activation (SA))**.** A sensor activation occurs when a sensor changes its state from the no-interaction state to interaction state. Reverse transitions, i.e. de-activations of sensors, are not considered. For example, when a user takes a glass, the activation is tagged as *glassSens*. A sensor activation is then composed by a timestamp and a sensor identifier:

$$SA = \{timestamp, sensorID\} \tag{3.1}$$

Definition 2 (Sensor activation dataset)**.** A time-ordered sequence of sensor activations.

Definition 3 (Actions)**.** Actions are the primitives of activities and are directly linked to sensor activations. For example, the activation of sensors *cupSens* and *glassSens* are linked to the action *hasContainer*. A sensor activation can only be mapped to one single action, but an action can be generated by different sensor activations. Actions do not have any duration and are hence described by a timestamp.

Definition 4 (Type)**.** When referring to activities and objects, type captures a purpose based classification. In this dissertation, the types considered are *Cooking*, *Hygiene*, *Entertainment* and *Housework*. An activity can only have one type, while objects can have more than one. For example, an object like a tap can be used for cooking, housework or hygiene. But a MakeCoffee activity can only be a cooking activity. On the other hand, when referring to sensors, type classifies sensors in terms of their technological base. In this dissertation, the modelled sensor types are *contact*, *electric*, *pressure* and *tilt* sensors.

Definition 5 (Location)**.** Location refers to semantic places of the monitored environment, where sensor activations occur and hence, activities are performed. For example, in a house, possible locations might be *bedroom*, *bathroom* or *kitchen*.

Definition 6 (Initial Activity Model (IAM))**.** Activity models are sequences of actions defined by a domain expert. Initial activity models refer to the minimum number of necessary actions to perform an activity. The objective of such models is to represent incomplete but generic activity models applicable to any user. Initial activity models also have an estimation of the maximum duration of the activity.

$$IAM(Activity_n) = \{action_a, action_b, \ldots, max\_duration\} \tag{3.2}$$

Definition 7 (Extended Activity Model (EAM))**.** A complete and specialised version of an IAM. By *complete* we mean that an activity model contains all the actions performed by a user for the corresponding activity. By *specialised* we mean there are two or more different complete action sequences for the corresponding activity, i.e. specialised sub-classes of that activity exist. The EAM for an activity is represented as a list of action sequences with their occurrence frequency:

$$\begin{gathered} EAM(Activity_n) = \{as_1, as_2, \ldots, as_n\} \quad where \\ as_i = \{frequency, action_a, action_b, \ldots, action_m\} \end{gathered} \tag{3.3}$$

Definition 8 (User Erratic Behaviour)**.** It happens when a user interacts with an object but the interacted object is not being used to perform the ongoing activity. Consider the case where a user wants to prepare pasta. In order to take the pasta from the store, the sugar package has to be removed first. The user will touch the sugar package and thus, a sensor activation will be generated. But this interaction does not mean that sugar is being used to prepare pasta.

Definition 9 (Positive Sensor Noise)**.** A sensor that should not get activated, i.e. there is no interaction with the object monitored by the sensor, gets activated because of sensor or monitoring infrastructure errors.

Definition 10 (Missing Sensor Noise)**.** A sensor that should get activated, i.e. there is an interaction with the object monitored by the sensor, does not get activated because of sensor or monitoring infrastructure errors.

The learning approach for activity models presented in this dissertation has some constraints:

Constraint 1 (Dense sensing-based activity monitoring)**.** *Even though there is no theoretical constraint that prevents the approach to be implemented for other activity monitoring approaches, the implementation presented in this dissertation is designed for the dense sensing-based activity monitoring approach.*

Constraint 2 (Single user - single activity scenario)**.** *This is a hard constraint, in the sense that the theoretical foundations of the learning approach assume explicitly that only one user is being monitored and this user is not allowed to perform concurrent or interleaving activities.*

Constraint 3 (Uniquely located activities)**.** *It is assumed that an activity can only be performed in a single location, during a concrete execution, i.e. although ReadBook can be performed in the bedroom and in the lounge, a concrete execution will take place in the bedroom or in the lounge (exclusive or).*

Constraint 4 ("Static" objects)**.** *Objects used by a user are assumed to be "static" respect to the location, i.e. if the location of a toothbrush is the bathroom, the object will always stay in the bathroom.*

3.3 **Solution Design**

Based on the definitions and constraints shown in Section 3.2, an offline system for extended activity model learning is presented. The objective of the system is to learn extended activity models (EAM, Definition 7) for every defined activity and a particular user, since every user has his/her own way to perform activities in terms of executed actions. The initial activity models (IAM, Definition 6) for every activity provided by a domain expert will be the same for every user, because they are generic models. But the extended activity models are personal models. They capture different complete activity models performed by a user. Notice that EAM learning has several implications in the ontology-based activity model approach:

1. Complete models are learnt for every activity and a particular user. Complete refers to the sequence of executed actions for a concrete activity.

2. If two or more different complete models are learnt for a concrete activity, specialised models of the same activity are found. This means that the defined activity is not a leaf in the ADL ontological tree and hence, subactivities of that activity are discovered.

3. Activity models can be dynamic. If the EAM learning system is continuously updated with new data, activity models can be updated. In consequence, activity models evolve and are adapted to users' varying behaviours.

3.3.1 Inputs of the learning system

The learning system for EAMs works on user generated data. It combines domain knowledge and user generated data in a learning algorithm. Notice that user data comes in an unlabelled sensor activation dataset, so there is no need to manually label any dataset. On the other hand, domain knowledge concerning the environment and activities is provided to the learning system. More concretely, the inputs of the EAM learning system are:

1. Context knowledge: the context knowledge represents the prior knowledge provided by a domain expert which is relevant to learn activity models. The concepts modelled in the context knowledge include:

    (a) Activities: every activity is defined by its name, type, location and IAM. Following Definition 4, an activity has only one type. However, an activity can be performed in several locations (notice that this does not contradict Constraint 3).

    (b) Objects: objects refer to every object in the environment which is monitored by sensors and is used to perform activities. An object is defined by its name, type, location and attached sensors. As Definition 4 states, an object can have several types. But an object is subjected to Constraint 4, so it has a unique location. In principle, objects can be monitored by several sensors. For instance, a fridge may have a tilt sensor attached to its door to monitor open/close door actions and an electric sensor to monitor the power usage.

    (c) Sensors: following Constraint 1, sensors are attached to objects and they are defined by a sensor identifier, type, described action and object to which is attached. Sensors produce sensor activations (Definition 1) when they transition from their no-interaction state to interaction state. Sensors have a unique type and they are mapped to one single action, as stated in Definition 3. The object to which the sensor is attached is also stored in the context knowledge.

2. Sensor activation dataset: as defined in Definition 2, an unlabelled timestamped sensor activation dataset with the activity traces of a concrete user.

In the current implementation, context knowledge is formatted in a JavaScript Object Notation (JSON)[1] file. JSON has been selected because it provides a lightweight knowledge formatting syntax which is widely supported and used to share

[1]http://json.org/

information. Although OWL or RDF could be used to implement context knowledge, the overhead introduced by such markup languages comes without any advantage for the EAM learning system. The decision of using JSON responds to the principle of providing a simple solution, focusing on modelling only relevant knowledge. As an example, Figure 3.2 shows how an activity, an object and a sensor are modelled in the context knowledge file. This knowledge is provided by a domain expert and modelled by knowledge engineers.

```
"MakeCoffee": {
        "type": ["Cooking"],
        "location": ["Kitchen"],
        "IAM": ["hasContainer", "hasCoffee"],
        "duration": 300
}
```

```
"kitchen-tap": {
        "type": ["Cooking", "HouseWork"]
        "location": "Kitchen"
        "sensors": ["ktapSens"]
}
```

```
"ktapSens": {
        "type": "tilt",
        "action": "turnOnTap",
        "attached-to": "kitchen-tap"
}
```

Figure 3.2: Example of activities, objects and sensors modelled in the context knowledge file. Activity duration is given in seconds.

The second input to the EAM learning system is the sensor activation dataset. For the implementation presented in this dissertation, sensor activation datasets are formatted in Comma Separated Value files (CSV)[1]. Each row of the file contains a timestamp (year, month, day and time) and a sensor activation tagged with the sensor identifier which has produced the activation (see Figure 3.3).

Based on those inputs, a context knowledge file and a sensor activation dataset, the output of the EAM learning system is a list of action sequences for each activity, which describe the EAMs for each activity (see Equation 3.3).

[1]http://en.wikipedia.org/wiki/Comma-separated_values

```
2014-05-23 09:47:33.984341,storeSens
2014-05-23 09:47:39.333528,potSens
2014-05-23 09:47:52.750216,cookerSens
2014-05-23 09:48:07.764138,fridgeSens
2014-05-23 09:48:12.591836,wmilkSens
2014-05-23 09:48:47.199512,chocoSens
2014-05-23 09:54:11.553695,mugSens
2014-05-23 09:54:40.794979,rcontrolSens
2014-05-23 09:54:50.390696,tvSens
2014-05-23 09:54:59.348862,sofaSens
```

Figure 3.3: A slice of a sensor activation dataset.

3.3.2 Intuition behind the solution

The rationale for the designed solution to EAM learning comes when sensor activations are plotted in a three-dimensional space spanned by location, type and time, as shown in Figure 3.4. This three-dimensional space, named *activity space*, reflects the prior knowledge about activities. An activity has a concrete type that is shared with the object types used to perform that activity and is performed in a location and a time segment.

When sensor activations describing several activities are plotted in the activity space, several lumps or clusters can be distinguished. Intuition says that sensor activations, and hence actions, have to be close to each other when they form an activity (close in the activity space), i.e. if a user is making pasta, it is common sense to think that sensor activations describing that activity will be located in the kitchen, and the objects which are being used by the user will be for cooking purposes. As far as time regards, sensor activations will occupy a time segment which will denote the duration of the activity.

This grouping of actions in the activity space can be seen in Figure 3.4. Six activities can be identified there, namely MakeChocolate, BrushTeeth, WashHands, MakePasta, MakeCoffee and WatchTelevision. Due to time resolution in the graphic, actions composing BrushTeeth, WashHands and WatchTelevision cannot be properly distinguished. Those three activities are quite short in time, but depicted actions for each of them are respectively five, three and three. There are some actions that are not grouped in any activity. Those actions are generated by objects that have multiple types. For example, in the case of MakePasta, the kitchen tap is used. The kitchen tap can be used for cooking or for housework and that is why it appears in both positions.

It is interesting to see how Figure 3.4 supports the intuition about action distribution in the activity space. In general, actions can be easily grouped in activities,

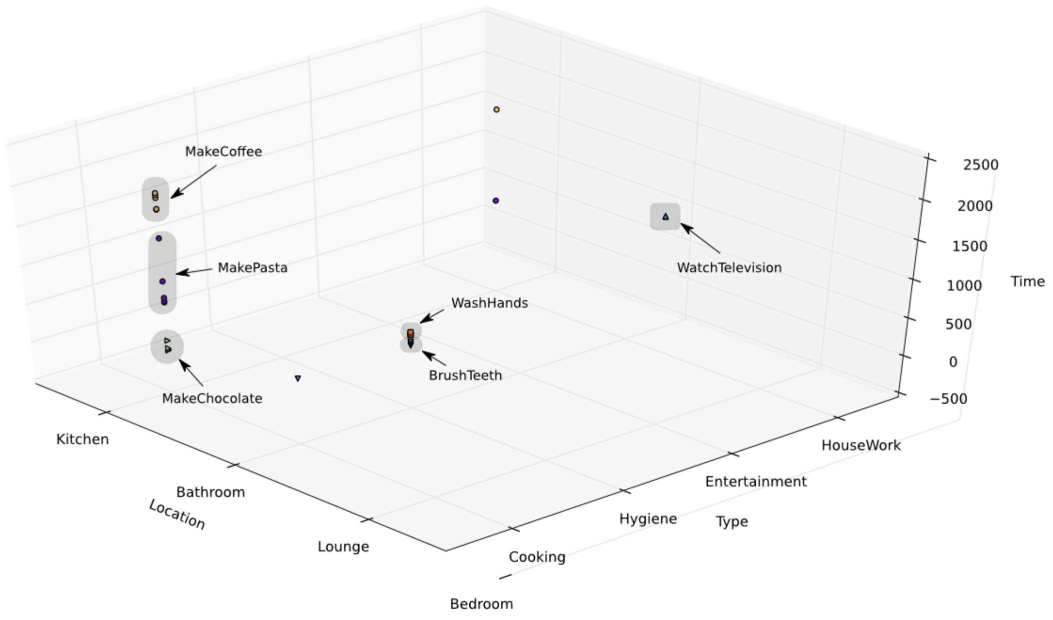


Figure 3.4: A plot of the actions executed by a user for performing several activities in the three-dimensional space spanned by location, type and time.

but there are also some tough cases. This is the case of actions composing activities BrushTeeth and WashHands, which are very close in time and are performed in the same location and with the same purpose. The last action of MakePasta seems also difficult to group, since it is closer in time to activity MakeCoffee, sharing again the same location and type.

In consequence, the idea is to design a clustering algorithm which takes advantage of location, type and time information to aggregate actions to clusters. The clustering process needs heuristics and metrics that take special care of those actions that are difficult to group in the corresponding activity. But this is not enough. Action clusters have to be identified and labelled with one of the defined activities. To identify action clusters with activity names, initial activity models in the context knowledge can be used. Initial activity models contain an incomplete sequence of actions. Despite being incomplete sequences, they can be used to identify activities. So the clustering algorithm has to produce several labelled action/sensor activation sequences per activity.

The clusters extracted in the clustering process may contain all the actions executed by a user to perform a concrete activity. Those clusters may be used to learn different ways of performing an activity and hence, to learn extended activity models. In summary, the key idea is to design a clustering algorithm that uses initial activity models to detect activities and some metrics in the activity space to aggregate actions to activities. Afterwards, using the clusters extracted and labelled, they are analysed to learn extended activity models for each activity.

3.3.3 Description of the solution

Following the rationale described in section 3.3.2, a solution for the EAM learning system has been designed. The devised system architecture is depicted in Figure 3.5. At a first glance, the two important blocks identified in section 3.3.2 can be distinguished in the architecture: a clustering process and an activity model learning process. The clustering process is implemented in two modules, namely the Semantic Activity Annotation module (SA^3) and the Action Aggregator module (AA). On the other hand, the activity learning process is implemented by a module called Activity Model Learner (AML).

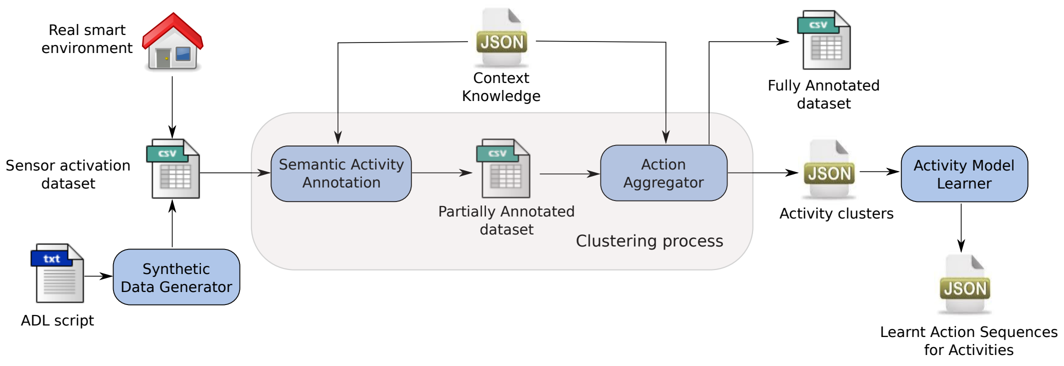


Figure 3.5: The detailed design architecture of the proposed approach to learn EAMs.

Two kinds of elements can be distinguished in the architecture depicted in Figure 3.5: files and software modules. The only exception to this rule is the *real smart environment*, which represents a sensorised environment where a user performs several activities. All the other elements can be classified into files or software modules. There are three kinds of files: plain text files (*ADL script*), JSON files (*context knowledge*, *activity clusters* and *learnt action sequences for activities*) and CSV files (*sensor activation dataset*, *partially annotated dataset* and *fully annotated dataset*). Files are used to exchange information between software modules. The developed software modules are four: *Synthetic Data Generator*, *Semantic Activity Annotation*, *Action Aggregator* and *Activity Model Learner*. All the software modules of the diagram have been implemented using Python 2.7[1] and its packages Numpy[2] for numerical computing and Pandas[3] for data analysis.

[1]https://www.python.org/
[2]http://www.numpy.org/
[3]http://pandas.pydata.org/

Once the elements of the architecture have been described, the solution itself will be detailed. Firstly, a sensor activation dataset has to be collected for a concrete user. A real smart environment or a synthetic data generator can be used to get this dataset. The sensor activation dataset contains all sensor activations registered for a user performing certain activities during a fixed period of time. Remember that sensor activations are unlabelled. Whatever method has been chosen to obtain it, the resulting sensor activation dataset is formatted in a CSV file and its information has already been described in Definition 2.

Afterwards, a novel clustering process is run on the sensor activation dataset, using the context knowledge provided by an expert and formatted in a JSON file. The proposed clustering process, which is extensively discussed in Chapter 4, is divided into two steps:

1. Semantic Activity Annotation algorithm (SA^3): this algorithm uses IAMs stored in the context knowledge to detect activities in the unlabelled dataset. It transforms sensor activations to actions according to the given context knowledge to apply a novel pattern recognition algorithm that has been designed and implemented to detect the occurrence of IAMs in the unlabelled sensor activation dataset. SA^3 also uses object and activity location information to infer from sensor activations where an activity is being performed and analyse consequently the feasibility of the detected pattern. SA^3 is an initialisation process from the clustering point of view, because it finds initial clusters in the time axis and identifies the label of those clusters. The output of SA^3 is given in the so called *partially annotated dataset*, where sensor activations and their resultant actions are labelled with activity names or the special label None, which indicates that the action could not be labelled with any known activity name. Notice that SA^3 only labels those actions that pertain to IAMs thus leaving the vast majority of actions without any label. This is the reason why the output of the algorithm is called *partially annotated*. A detailed description of SA^3 is given in Section 4.1.

2. Action Aggregator algorithm (AA): the input to the algorithm is the output of SA^3, i.e. the partially annotated dataset file. AA uses activity, object and action knowledge described in the context knowledge to expand initial action clusters detected by SA^3. To analyse those actions that remain unlabelled after SA^3, AA defines boolean functions based on location, type and time information of actions and objects. It analyses unlabelled actions that are close to initial clusters, checking the feasibility of those actions pertaining to those close initial clusters. The feasibility is computed using location, type and time information, implementing the idea that actions pertaining to an activity are performed in the same location, with a coherent purpose and

close in time. The AA algorithm's result is stored in another CSV file called *fully annotated dataset*, which assigns an activity label to all actions and their original sensor activations. The special label None is used for those actions that are not considered part of any activity. Those actions can be originated by sensor noise or user erratic behaviour, i.e. a user that interacts with an object that is not being used for performing an activity. Additionally, AA also generates a JSON file called *activity clusters*. This file contains all the clusters found by the algorithm for each activity. Clusters, which are action sequences, come with an occurrence frequency. The activity clusters file is also used to store some interesting statistics found by AA about the frequency of objects used and actions executed for each activity, frequency of locations where each activity has been performed and activity duration statistics (mean and standard deviation). An example of an activity clusters file can be seen in Figure 3.6. The AA algorithm is described in Section 4.2.

The main result of the clustering process is thus stored in the activity clusters JSON file. This file has the information of all action sequences executed by a user to perform a concrete activity. For example, in the file shown in Figure 3.6, in the "patterns" section, all the action sequences found by AA are stored. In this case, for activity ReadBook, only one cluster has been generated with 53 occurrences in the sensor activation dataset. According to AA, ReadBook has been performed by means of executing the action sequence *useFurniture*, *turnOnLamp* and *hasBook*. Nevertheless, AA generally finds many different clusters per activity. Those clusters are finally processed by Activity Model Learner (AML), which filters incorrect action sequences and outliers to learn final EAMs as a list of action sequences. AML uses action sequence similarity metrics to implement an outlier detection algorithm. Notice that all action sequences or clusters detected by AA are not valid activity models, since they may contain incorrectly labelled actions (due to clustering errors, sensor errors and/or user erratic behaviour) and missed actions (due to sensor errors). Those spurious action sequences have to be detected and removed, to discover those action sequences that are valid models for a given activity. Valid action sequences are the EAMs for activities and are stored in a JSON file called *learnt action sequences for activities*. All the details about AML are provided in Chapter 5.

3.4 Summary and Conclusions

A global solution has been designed to learn extended activity models from initial generic but incomplete activity models. The base of the solution is the ontology-based activity modelling, which provides a unified and reusable framework to

```
"ReadBook": {
    "locations": [
        [42, "Bedroom"],
        [11, "Lounge"]
    ],
    "actions": [
        [53, "useFurniture"],
        [53, "turnOnLamp"],
        [53, "hasBook"]
    ],
    "patterns": [
        [
            53,
            [
                "useFurniture",
                "turnOnLamp",
                "hasBook"
            ]
        ]
    ],
    "objects": [
        [42, "bed"],
        [42, "bedroom-lamp"],
        [42, "book-b"],
        [11, "sofa"],
        [11, "lounge-lamp"],
        [11, "book-a"]
    ],
    "occurrences": 53,
    "duration": [
        14.773584905660377,
        2.6467602797835856
    ]
},
```

Figure 3.6: Information stored in the activity clusters file for activity ReadBook.

model ADLs. The presented solution for learning aims at completing and specialising initial activity models provided by a domain expert.

For that purpose, this chapter introduces some important definitions and constraints, setting the scenario in which the learning system works. Afterwards, the rationale behind the solution design has been described. It has been shown that when actions pertaining to activities are plotted in the so called activity space - spanned by time, type and location - activities are arranged in clusters. The idea is to be able to identify those clusters and aggregate surrounding actions defining appropriate metrics and heuristics. Once those clusters have been identified, a learning algorithm will be deployed to learn extended activity models from identified clusters.

In conclusion, this chapter has established the ground in which the contributions of the dissertation stand. The addressed problem has been described and a solution for that problem has been proposed. The following chapters will show how this solution is deployed.

If you keep doing what you've always done, you'll keep getting what you've always gotten.

Neil Strauss

CHAPTER 4

A Clustering Process for Activity Annotation

A novel clustering process for activity annotation in an unlabelled dataset is described in this chapter. Using previous knowledge obtained from a domain expert, the clustering process identifies activities performed by a user in an unlabelled sensor activation dataset. The clustering process is claimed to be novel because it combines unsupervised learning techniques with domain knowledge. In contrast with a typical clustering algorithm, which finds a structure in a given dataset, the proposed algorithm can also link the extracted structures with a semantic activity name. This is possible because initial activity models are used, i.e. generic but incomplete activity models provided by a domain expert.

The clustering algorithm is divided into two steps, as introduced in Chapter 3: (i) a special pattern recognition algorithm that finds initial activity models' occurrences in the sensor activation dataset (SA^3), and (ii) a data aggregation step where object, sensor and action knowledge is used to define location, type and time based metrics which are used to aggregate actions to activities (AA). Following this division, the chapter is divided in three sections: Section 4.1 describes the Semantic Activity Annotation algorithm (SA^3), which is presented as the initialisation step of the clustering process. Section 4.2 presents the second step of the clustering process, where actions are aggregated to corresponding initial action clusters provided by SA^3 which describe an activity performance. Section 4.3 summarises this chapter and draws the extracted conclusions.

4.1 SA^3: Semantic Activity Annotation Algorithm

The objective of the SA^3 algorithm is to find sequences of actions that are describing an activity in an unlabelled sensor activation dataset. For that purpose, initial activity models (IAM) stored in the context knowledge are used. Remember that IAMs, as defined in Definition 6, are characterised by a sequence of necessary actions to perform an activity plus a duration estimation.

The problem can be stated more formally as:

Problem (SA^3)**.** *Given a context knowledge file where activities, sensors and objects are described and an unlabelled sensor activation dataset, find the occurrences of IAMs that form valid executions of activities.*

Considering that IAMs are sequences of actions and that sensor activation datasets are sensor sequences, the SA^3 problem can be seen as a pattern recognition problem, where IAMs act as the patterns to be recognised in the sensor activation dataset. Although SA^3 is actually a pattern recognition algorithm, there are some important features that make the problem special:

1. IAMs are sequences of actions, whereas sensor activation datasets contain sensor activations, i.e. the activation of a concrete sensor of the environment.

2. As IAMs are incomplete activity models, a user will generally execute more actions than those considered in the IAMs to perform activities. As a consequence a sensor activation dataset will generally include sensor activations and actions that are interleaved with IAM actions for the same activity.

3. IAMs have no information about the order in which actions are executed, thus if $IAM(A) = \{a, b, c\}$, a sensor activation dataset containing the sequence $\{c, a, b\}$ has to be identified as activity A, i.e. the elements of the pattern to be recognised might appear in varied orders.

4. Even though a sequence corresponding to an IAM is found, it does not necessarily describe a valid activity, since duration and location are important. For example, for an activity whose typical duration is 5 minutes, the corresponding actions are found but their distance is of 3 hours; it is then common sense to think that those actions do not form a valid activity. Hence, the pattern recognition algorithm needs to take action time distances and locations into account.

5. Sensors are prone to errors and in consequence, sensor activation datasets contain noise. The pattern recognition algorithm has to work thus in noisy environments. More concretely, the considered kinds of sensor noise are positive and missing sensor noise (Definitions 9 and 10).

In summary, SA^3 has to use IAMs as patterns to be recognised in an unlabelled sensor activation dataset, where actions are executed in varied orders, where actions that are not in IAMs can appear interleaved, where duration and location information is crucial for the validity of a pattern and where sensor noise exists.

In order to tackle the described problem a three-step algorithm has been designed and implemented (see Figure 4.1). The first step, named sensor-action transformation step, uses sensor information from the context knowledge to transform sensor activations into actions (Section 4.1.1). The second step, the activity sequence finding step, runs a pattern recognition algorithm using IAMs and object information from context knowledge, as described in Section 4.1.2. This step addresses the varied order of actions, interleaved actions not pertaining to IAMs, duration and location information for valid activities and sensor noise. Finally, the third step called correct activity sequence fitting fixes the overlapping activities generated in the second step (Section 4.1.3).

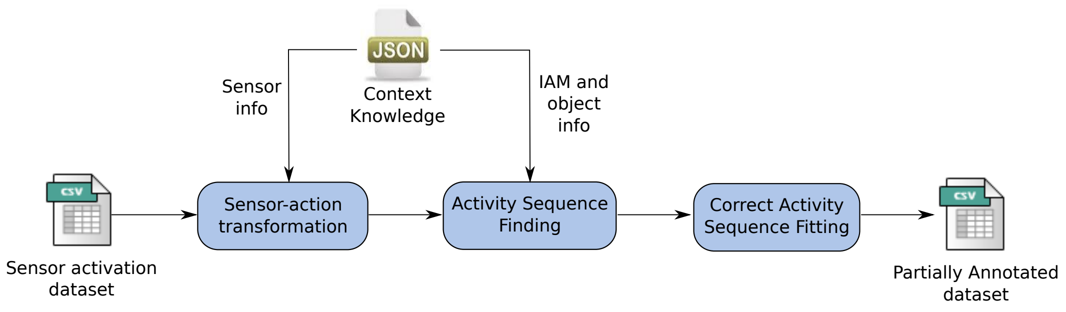

JSON
Context Knowledge
Sensor info
IAM and object info
CSV
Sensor activation dataset
Sensor-action transformation
Activity Sequence Finding
Correct Activity Sequence Fitting
CSV
Partially Annotated dataset

Figure 4.1: The three-step algorithm for SA^3.

The result of SA^3 is the partially annotated dataset. SA^3 can only label those actions pertaining to IAMs and thus it cannot infer a label for all the other actions of the sensor activation dataset. An example of how a partially annotated dataset looks like is provided in Table 4.1. The first column shows the timestamp of the sensor activation. The second column shows the activated sensor. Those two columns form the sensor activation dataset. SA^3 adds the third, fourth and fifth columns, where action, activity label and start/end tags are provided. Start tag refers to the first action pertaining to the IAM of a valid activity discovered by SA^3. End tag has the same meaning but for the last action. In consequence, start and end tags show the start and end times for a detected activity. For convenience, and even though any action not pertaining to the IAM of the detected activity cannot be labelled by SA^3, all the actions between start and end times are labelled with the detected activity name, and not with the special label None. This labelling criterion is applied to make visualization easier and it does not have any effect on the output of SA^3, neither for AA nor for evaluation purposes.

The next sections describe in detail the three steps of SA^3 depicted in Figure

Timestamp	Sensor	Action	Activity	Start/End
2014-05-23 07:42:17.106962	wsugarSens	hasFlavour	None	
2014-05-23 07:49:17.310460	bedSens	useFurniture	None	
2014-05-23 09:43:44.128079	cupSens	hasContainer	MakeChocolate	start
2014-05-23 09:47:33.984341	storeSens	openStore	MakeChocolate	
2014-05-23 09:47:39.333528	potSens	useCookingUtensil	MakeChocolate	
2014-05-23 09:47:52.750216	cookerSens	useCookingAppliance	MakeChocolate	
2014-05-23 09:48:07.764138	fridgeSens	openFridge	MakeChocolate	
2014-05-23 09:48:12.591836	wmilkSens	hasMilk	MakeChocolate	
2014-05-23 09:48:47.199512	chocoSens	hasChocolate	MakeChocolate	end
2014-05-23 09:54:11.553695	mugSens	hasContainer	None	

Table 4.1: Example of a partially annotated dataset, the output of the SA^3 algorithm shown in table format.

4.1. Finally, Section 4.1.4 shows the complete algorithm and illustrates it with an example.

4.1.1 The sensor-action transformation step

The first step takes as inputs the sensor activation dataset and the context knowledge file to transform every sensor activation into an action. For each sensor activation in the dataset, the corresponding sensor model is checked in the context knowledge file. As shown in Figure 3.2 every sensor has the action to which it has to be mapped in the context knowledge file provided by the domain expert. This simple transformation is possible due to the dense sensing-based activity monitoring approach, where sensor activations are directly linked to user-object interactions and thus to actions.

In consequence, a sensor activation sequence such as:

$$\{cupSens, wsugarSens, smilkSens\}$$

will be transformed to:

$$\{hasContainer(cup), hasFlavour(white\text{-}sugar), hasMilk(skimmed\text{-}milk)\}$$

For the sake of clarity, and given that action matching does not care about concrete objects used to execute that action, *hasContainer(cup)* will be used as *hasContainer*. But notice that the object information obtained through the sensor activation is not removed.

Sensor-action transformation step allows performing pattern recognition in the action space, abstracting from concrete sensor activations. This step can be kept simple thanks to the dense sensing-based approach. For different activity monitoring approaches, such as wearable sensors, more complex sensor-action transformation steps should be implemented. However, notice that as far as sensor information can be transformed to actions, SA^3 and the entire clustering process can work without any problem. This is why Constraint 1 is considered to be a weak constraint.

4.1.2 The activity sequence finding step

The objective of this step is to find all valid occurrences of IAMs in the action space obtained from the sensor-action transformation step. An iterative process is run over actions. Firstly, the algorithm checks whether the current action pertains to any of the IAMs stored in the context knowledge file. If the answer is no, the action is labelled with the None label and the next action is treated. But if the answer is yes, the IAMs of the activities in which the action appears are considered as possible activities. Notice that if the action pertains to more than one IAM, all the possibilities are treated. Assume the action a pertains to $IAM(A_1)$ and $IAM(A_2)$. Two action sequences are created $S(A_1) = \{a\}$ and $S(A_2) = \{a\}$. The algorithm iterates through the next actions, in order to find sequences of actions that form valid activities in correspondence of possible activities.

The process to find valid activities works as follows:

1. Current action b is added to the created action sequences: $S(A_i) = append(S(A_i), b)$.
2. If current action b pertains to an IAM of an activity for which no sequence has been open, open a new sequence.
3. Check whether candidate sequences form a valid activity in terms of *completion*, *duration* and *location*
4. If any of the candidate sequences forms a valid activity, close the sequence and follow with remaining sequences.
5. Iterate through actions until all sequences are closed (valid activities) or removed (invalid activities).

To fully understand the described process to check for valid activities, the *completion*, *duration* and *location* criteria have to be explained:

1. Completion criterion: an action sequence has to contain all the actions of the corresponding IAM to be considered a complete action sequence for the corresponding activity. An action sequence $S(A_i)$ is complete for activity A_i if and only if
$$IAM(A_i) \subseteq S(A_i) \tag{4.1}$$

2. Duration criterion: for an action sequence to fulfil the duration criterion, the duration of the action sequence has to be smaller than the duration estimation of the corresponding IAM. The duration of an action sequence $S(A_i)$ is calculated as the rest of the timestamps of the last action and the first action:

$$\Delta_t(S(A_i)) = t(S(A_i)_{last}) - t(S(A_i)_{first}) \tag{4.2}$$

where $t(S(A_i)_j)$ is the timestamp of the j-th action of the sequence $S(A_i)$. Duration estimations for activities are stored in the context knowledge file and are written as $\Delta_t(A_i)$. Hence, the duration criterion establishes that an action sequence fulfils the duration criterion if and only if

$$\Delta_t(S(A_i)) \leq \Delta_t(A_i) \tag{4.3}$$

3. Location compatibility criterion: the actions of sequence $S(A_i)$ which pertain to $IAM(A_i)$ have been transformed from corresponding sensor activations. Those sensor activations can be traced in the context knowledge to the used objects. And those objects have a location in their model. Hence, the location of actions can be inferred using the context knowledge file. Using this information, the location of the possible activity described by sequence $S(A_i)$ is inferred. The location of sequence $S(A_i)$ is inferred as the common locations of actions of the sequence pertaining to $IAM(A_i)$. If actions pertaining to the IAM do not share a common location, the sequence has no location and thus, it is not compatible with any activity in terms of location. On the other hand, if a common location is shared, that location has to be compatible with the locations of the corresponding activity, modelled in the context knowledge file. Notice that only locations of actions pertaining to an IAM are considered. All the other actions can be generated by noise or erratic behaviour and in consequence, their locations are not considered in SA^3. This can be seen better with an example: let $S = \{a, b, c, d\}$ be a sequence of actions, where a and d pertain to the IAM of activity A. For location inference, only actions that pertain to the IAM of the detected activity are considered, since all the other actions of the sequence S cannot be considered as part of the activity yet. For instance, some of those actions might be produced by positive noise. Hence, the sequence S is an activity A with location L if a and d have been executed in the same location and that location is compatible with A activity's locations. Formally expressed:

$$\begin{gathered} S = \{a, b, c, d\},\ A : \text{candidate activity and } IAM(A) = \{a, d\} \\ S \text{ is explained by } A \rightarrow loc(obj(a)) = loc(obj(d)) \in Locations(A) \end{gathered} \tag{4.4}$$

where $obj(a)$ returns the object which has been mapped to action a, $loc(o)$ returns the location of object o in the environment and $Locations(A)$ returns the list of possible locations of activity A according to the context knowledge file.

Action sequences that fulfil the completion, duration and location criteria will be considered as valid activities and stored as such. Action sequences that do not fulfil those criteria will be removed, since they do not form a valid activity description.

4.1.3 The correct activity sequence fitting step

Depending on the activity models, the sensor activation dataset and noise levels, valid activities can overlap each other. Moreover, as many IAMs will share some actions, activity overlapping will be a normal situation after the activity sequence finding step described in Section 4.1.2. But by virtue of Constraint 2 (only single user - single activity scenarios are considered), interleaving thus overlapping activities cannot appear in the sensor activation dataset. So a special step has to be designed to treat overlapping activities generated in the activity sequence finding step.

Having a set of overlapping activities, described through the corresponding action sequences, this step introduces a heuristic directly derived from Constraint 2 to return a set of non-overlapping activities which provides the best explanation for the generated situation. The heuristic states that in a set of overlapping action sequences, the maximum number of non-overlapping action sequences have to be returned.

The reason to adopt this heuristic is that the vast majority of sensor activations are originated by deliberated user-object interactions. In consequence, the vast majority of sensor activations must be part of an activity. In order to maximise the number of sensor activations which correspond to the execution of an activity, the heuristic chooses the maximum number of non-overlapping activities in a given sequence. Because this is the way to maximise the number of sensor activations in activities.

4.1.4 The complete SA^3 algorithm

Sections 4.1.1, 4.1.2 and 4.1.3 describe the first, second and third steps of the SA^3 algorithm. This three-step algorithm is depicted as pseudo-code in Algorithm 1. It has been designed to work with noisy sensor activations, varying order for activity executions and sensor activations that do not belong to any initial activity model. That flexibility allows applying the tool to many different datasets.

Algorithm 1 SA^3 algorithm for semantic activity annotation

```
Require: sensor_activation_dataset, context_knowledge
Ensure: partially_annotated_dataset
  $partially\_annotated\_dataset \leftarrow createDataset(sensor\_activation\_dataset)$
  $action\_dataset \leftarrow applyTransformFunction(sensor\_activation\_dataset,$
  $context\_knowledge)$
  $IAM\_list \leftarrow obtainIAMS(context\_knowledge)$
  for all $action \in action\_dataset$ do
    if $action \in initial\_activity\_models$ then
      $activities \leftarrow obtainActivities(action, IAM\_list)$
    end if
    for all $activity \in activities$ do
      // Use duration, completion and location criteria
      $valid\_activities \leftarrow findValidActivities(context\_knowledge)$
    end for
  end for
  $partially\_annotated\_dataset \leftarrow findNonOverlappingActivities(valid\_activities)$

  return $partially\_annotated\_dataset$
```

An illustrative example is shown next, to describe how SA^3 works. Timestamps are ignored in the example, assuming that the duration criterion is fulfilled by the shown action sequence and considered activities. This assumption helps making the example clearer. The illustrative example has been designed to show in detail how SA^3 works, so it does not have to be coherent with the reality.

Imagine initial activity models for MakeCoffee, MakeTiramisu, MakeWhippedCream and BrushTeeth are defined as follow:

$$\begin{aligned} MakeCoffee &= \{hasCoffee, hasContainer, hasFlavour\} \\ MakeTiramisu &= \{hasCream, hasContainer, hasCoffee\} \\ MakeWhippedCream &= \{hasFlavour, hasContainer, hasCream\} \\ BrushTeeth &= \{hasBrusher, hasToothPaste, turnOnTap\} \end{aligned}$$

Notice that there are many actions that are used for the IAMs of several activities. For instance, *hasContainer* is used in the IAMs of MakeCoffee, MakeTiramisu and MakeWhippedCream. Using the same actions for various activities makes the problem harder, since overlapping activities are more common.

Let us consider the following sensor activation sequence from the sensor activation dataset:

$$\{mugSens, creamSens, spoonSens, whitesugarSens, brusherSens,\\ afcoffeeSens, toothpasteSens, glassSens, btapSens\}$$

Applying the sensor-action transformation step using the context knowledge as described in Section 4.1.1, the following action sequence is obtained (concrete objects are ignored in the action sequence):

$$\{hasContainer, hasCream, useCookingUtensil, hasFlavour, hasBrusher,\\ hasCoffee, hasToothpaste, hasContainer, turnOnTap\}$$

Figure 4.2 shows all the activities found by the second step of SA^3 (Section 4.1.2). Activities overlap each other, because there are several actions that belong to several activities. Notice also that there are some actions that are not in any activity model, which is totally feasible for the approach.

```
hasContainer        |     |
hasCream            | MWC |
useCookingUtensil   |     |
hasFlavour          |     | MT
hasBrusher                |     |
hasCoffee                 |     |
hasToothpaste                   | BT
hasContainer                    |
turnOnTap                       |
```

Figure 4.2: Illustrative example of the output of the activity sequence finding step of SA^3. MWC refers to MakeWhippedCream, MT to MakeTiramisu and BT to BrushTeeth.

Assuming that the duration criterion is always fulfilled for this example, Figure 4.2 shows how the completion and location criteria work. For instance, action sequence $S = \{hasContainer, hasCream, useCookingUtensil, hasFlavour\}$ is a valid sequence to describe activity MakeWhippedCream, because it contains all the actions of $IAM(MakeWhippedCream)$ and it is location compatible. Actions *hasContainer, hasCream* and *hasFlavour*, which pertain to $IAM(MakeWhippedCream)$ are obtained by interacting with objects *mug, cream* and *white-sugar*. The location of all those objects is the kitchen. As activity location for MakeWhippedCream is also the kitchen, the action sequence and the activity are location compatible.

However, consider the following action sequence $S = \{ hasFlavour, hasBrusher, hasCoffe, hasToothpaste, hasContainer\}$. Assuming duration criterion is satisfied, it can be seen that the completion criterion is also satisfied for activity MakeCoffee. But sequence S does not form a valid activity because the last action *hasContainer* is produced by the object *glass*, which is located in the bathroom. As *hasContainer* is in $IAM(MakeCoffee)$, its location is taken into account for location inference and compatibility. In this case, sequence S does not have a unique location and hence, it is removed.

After the activity sequence finding step finishes, three overlapping valid activities are found, namely MakeWhippedCream, MakeTiramisu and BrushTeeth. Such a situation cannot happen in a single user - single activity scenario, so the third step has to be applied: the correct activity sequence fitting step (Section 4.1.3). Using the defined heuristic, two activities are returned:

$$\begin{aligned} MakeWhippedCream &= \{hasContainer, hasCream, useCookingUtensil, \\ &\qquad hasFlavour\} \\ BrushTeeth &= \{hasBrusher, hasCoffee, hasToothPaste, hasContainer, \\ &\qquad turnOnTap\} \end{aligned}$$

Those two activities correspond to the maximum number of non-overlapping activities found in the set of valid activities. The action sequence for MakeWhippedCream makes perfect sense. There is an action, *useCookingUtensil*, which is not in any IAM, but it is very coherent with the activity. Even though SA^3 labels this action with the MakeWhippedCream tag, remember there is no way for SA^3 to know whether this action is really part of the activity. Such a labelling is done only for convenience. On the other hand, the action sequence for activity BrushTeeth contains a strange action: *hasCoffee*. In a noisy scenario this action may be generated by sensor or communication infrastructure errors. The other action which is not in the IAM of BrushTeeth is *hasContainer*. It is generated by the *glass* sensor, which is located in the bathroom. The action *hasContainer* may refer to a glass used in the bathroom to rinse the mouth.

Looking at Figure 4.2, the heuristic defined in the correct activity sequence fitting step can be visually understood. As it can be seen, returning the maximum number of non-overlapping activities maximises also the number of actions inside activities.

4.2 AA: Action Aggregator

Initial clusters provided by SA^3 are expanded in this step, using context knowledge and time-based metrics for action aggregation. The inputs of the AA algorithm are the context knowledge file (see Section 3.3 and specially Figure 3.2) and the partially annotated dataset returned by SA^3 (see Table 4.1). The output of SA^3 is interpreted as the initialisation of the clustering process in the activity space spanned by location, type and time axis. SA^3 has been designed to provide the number of clusters in the dataset, their semantic label and their centroids in the time axis. Having this initialisation, AA is responsible for analysing all actions considering the three dimensions of the activity space and deciding which actions are aggregated to the initial clusters provided by SA^3.

Assume Figure 4.3 shows the output of SA^3 for a concrete sequence of actions. For simplicity, only the time axis is shown in the figure. Dashes represent time intervals without any action. Circles represent actions that are in one or more IAMs. Notice that two circles do not necessarily have to be the same action. The only property they have is that a circle action is in at least one of the IAMs of the context knowledge file. Crosses, on the other hand, are actions that are not included in any IAM. Once again, two crosses do not have to represent the same action.

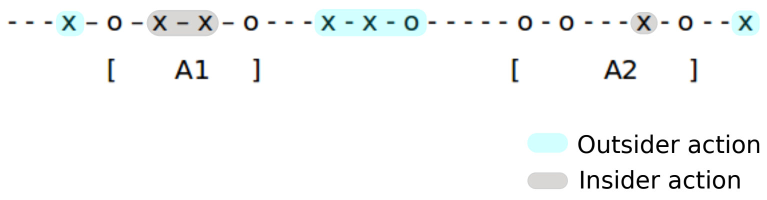

- - - X - O - X - X - O - - - X - X - O - - - - - O - O - - - X - O - - X
[A1]
[A2]
Outsider action
Insider action

Figure 4.3: Output of SA^3 for a concrete sequence of actions, where outsider and insider actions are identified.

Figure 4.3 shows that SA^3 detects activities A_1 and A_2 in that sequence of actions. As stated in Section 4.1 only actions that are in the IAM of the detected activity can be really considered part of that activity, i.e. only circles that are inside activities A_1 and A_2 are labelled with the corresponding activity labels.

Given that action classification, two new definitions can be assumed:

Definition 11 (Insider action)**.** Every action that is inside the initial clusters provided by SA^3 but is not in their IAMs is considered an *insider action* or *insider*, in short.

Definition 12 (Outsider action)**.** Every action out of the initial clusters provided by SA^3 is an *outsider action* or *outsider*, in short.

Due to single-user single-activity scenario constraint (Constraint 2), an insider may pertain to its wrapping activity or to none assuming the output of SA^3 is correct, i.e. it has been produced by noise or user erratic behaviour. There is no chance for an insider to pertain to an activity other than its wrapping activity because such an assumption would imply the existence of interleaved activities (remember this is true assuming that the output of SA^3 is correct, which is assumed in the AA algorithm). The situation for outsiders is different since an outsider can pertain to its previous activity, next activity or to none.

The inherent difference between insiders and outsiders demands a different treatment for both cases. AA first treats all insider actions and afterwards computes all outsiders using different approaches.

4.2.1 **Insider actions**

To decide whether an insider has to be added to its wrapping activity, a *compatibility function* between an activity and an action is defined. The compatibility function is a boolean function that captures whether an insider action has been executed in the same location as the activity and its purpose is coherent with the activity type:

$$Comp(A, a) = Loc(A, a) \wedge Type(A, a) \tag{4.5}$$

where A is an activity and a is an action. $Loc(A, a)$ is the location compatibility between an activity and an action, defined as in Section 4.1.2. The location of the concrete activity A is inferred using the same process as in SA^3, i.e. the common location of the actions pertaining to the IAM of the activity is set to be the location of the activity. $Loc(A, a)$ returns whether the inferred location for action a is the same as the inferred location for activity A. On the other hand, $Type(A, a)$ is the type compatibility between an activity and an action. Action type is inferred from the object used to execute the action. Every object in the context knowledge has a list of types. Activity type is unique and it is also retrieved from the context knowledge file. So $Type(A, a)$ is calculated as the intersection between the list of types of the action a and the type of the activity A. If the intersection is empty, the action and the activity are not type compatible.

Hence an insider action a will only be aggregated to its wrapping activity A, if $Comp(A, a) = True$, i.e. the insider has been executed in the same location of the activity, and its purpose is compatible with the activity type. If $Comp(A, a) = False$, the insider will be labelled with the None label, which means that it is not part of any activity.

4.2.2 **Outsider actions**

As an outsider can be aggregated to its previous or next activity, first of all the algorithm checks the feasibility of both options, defining another boolean function called *candidate function*:

$$Cand(A, a) = Comp(A, a) \wedge InRange(A, a) \tag{4.6}$$

$Comp(A, a)$ has already been defined in Equation 4.5, so the candidate function also takes into account location and type. The $InRange$ function is another boolean function that captures time feasibility. Thus the candidate function states that an activity A is a candidate activity for action a, if they are compatible ($Comp(A, a) = True$) and in range ($InRange(A, a) = True$).

As commented above, the $InRange$ function has been defined to capture the time feasibility. For example, if an action has been executed two hours before an activity whose estimated duration is three minutes, the action should not be aggregated to that activity, since it is almost impossible for the action to be related with the activity. This reasoning can be implemented in a boolean function using time distance and activity duration.

To capture time feasibility, the duration given by the expert in an IAM is interpreted as the standard deviation for concrete executions of that activity, i.e. the vast majority of the activities executed by any user, will lie inside the time area limited by the duration. This can be seen in Figure 4.4. So time distances among actions pertaining to a concrete activity are modelled by a Gaussian distribution, where the duration estimation given by the expert for that activity is the standard deviation. A Gaussian distribution has been selected since it captures perfectly the idea of activity duration as a time estimation where the majority of executions of that activity occur. The probability for actions of an activity to lie in the area limited by the duration estimation is very high and gets lower as it gets further from that duration.

Nevertheless, due to human behaviour variations, there will be some executions whose duration is outside the standard deviation. To capture those executions, the $InRange$ function considers all the actions lying inside k standard deviations. For the experiments run in this dissertation $k = 2$ has been set, but the value can be adjusted depending on the area which is desired to cover. For a concrete activity execution, the mean of the activity is calculated as the centre of the detected start and end of the activity, as given by SA^3. Using this mean calculation, the duration estimation given in the context knowledge is the standard deviation of the Guassian distribution for action time distances. Hence, in the case depicted in Figure 4.4, where Gaussian distributions are calculated as described, the outsider action represented with a star is in range with activities A_1 and A_2 for $k = 2$.

In consequence, using the candidate function, the following cases can be faced for an outsider action a and surrounding activities A_1 and A_2:

1. $Cand(A_1, a) = Cand(A_2, a) = False \rightarrow a$ is noise (None label)
2. $Cand(A_1, a) = True \wedge Cand(A_2, a) = False \rightarrow$ aggregate a to A_1
3. $Cand(A_1, a) = False \wedge Cand(A_2, a) = True \rightarrow$ aggregate a to A_2
4. $Cand(A_1, a) = Cand(A_2, a) = True \rightarrow$ need of a new heuristic

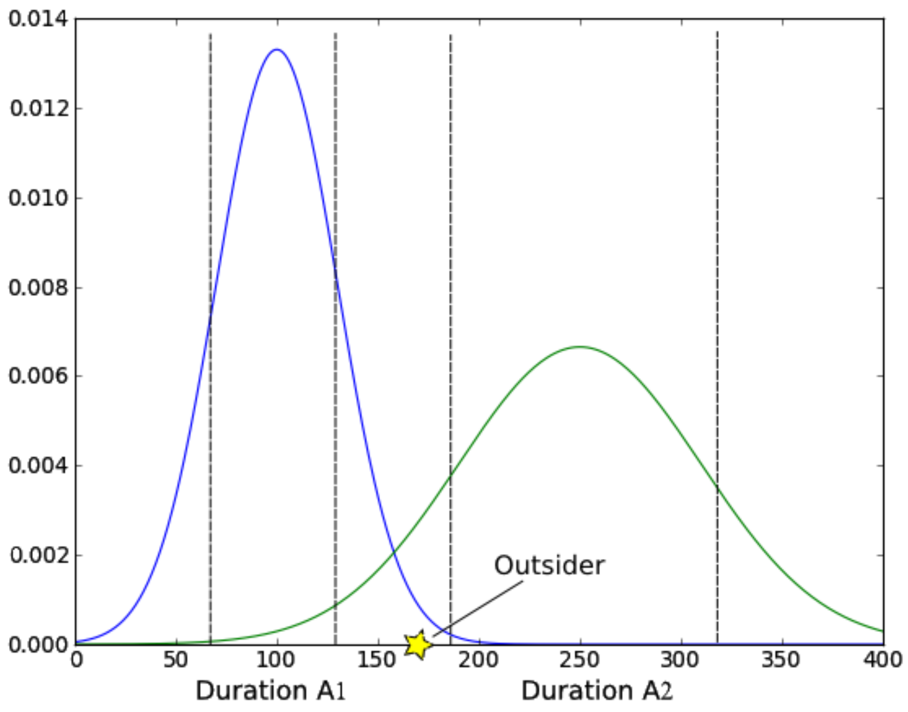


Figure 4.4: Gaussian distributions of activity durations for activities A_1 and A_2. Vertical lines show the standard deviation and the star represents an outsider action.

The first three cases provide a clear classification for an outsider. The candidate function determines unambiguously the label for action a. But for the fourth case, where previous and next activities are both candidate activities, a new heuristic has to be defined. There is no more information in the context knowledge than the location and type information to relate outsider actions to activities, thus context knowledge cannot be used to decide between activities A_1 and A_2. Hence a new heuristic is defined stating that an outsider action a will be aggregated to the time closest activity:

$$min\{\Delta_t(A_1, a), \Delta_t(A_2, a)\} \tag{4.7}$$

To implement this heuristic, a definition for $\Delta_t(A, a)$, the time distance between an activity and an action, is needed. As an activity has a duration and an action is described by a time instant, three time metrics are proposed:

1. Simple time distance: the distance between the centre of the activity as given by SA^3 and the action time coordinate.

$$\Delta_t(A, a) = |C_A - t_a|,\ where\ C_A = \frac{t_{end}^{SA^3} - t_{start}^{SA^3}}{2} \tag{4.8}$$

2. Normalised time distance: simple time distance normalised by the duration of the activity as given by the expert. This time metric, as opposed to the simple time distance, treats equally all activities regardless of their duration. Notice that the duration given by the expert is used rather than the duration detected by SA^3, since SA^3 may give varying durations depending on the order of executed actions that pertain to the IAM of the detected activity. Hence, the duration given by SA^3 cannot be a good reference. But notice also that to calculate the centre of the activity, the output of SA^3 is used. In this case, this information is the only one that allows calculating the centre. So the duration given by the expert is projected symmetrically around the centre calculated from SA^3:

$$\Delta_t(A, a) = \frac{|C_A - t_a|}{Duration_A},\ where\ C_A = \frac{t_{end}^{SA^3} - t_{start}^{SA^3}}{2} \tag{4.9}$$

3. Dynamic centre normalised time distance: only used for the previous activity, it dynamically calculates the centre of the activity depending on already aggregated actions.

$$\Delta_t(A, a) = \frac{|C_A - t_a|}{Duration_A},\ where\ C_A = \frac{t_{start}^{AA} + Duration_A}{2} \tag{4.10}$$

Let us explain the third time distance in detail. Activity start and end times provided by SA^3 are not generally trustful, since they depend on what actions are in the IAM of the detected activity and on the order of executions of those actions by the user. Imagine that $IAM(A) = \{a, b\}$ and that the action sequence processed by SA^3 is $S = \{d, b, a, c\}$. Assume that sequence S is the action sequence performed by the user for activity A. SA^3 will detect that activity A is being performed in sequence S, but it will only label actions b and a as pertaining to the activity. SA^3 does not have any information to know whether actions d and c pertain to activity A. So the start time of activity A for SA^3 will be the timestamp associated to action b and not to action d. The same happens for the activity end time.

Nevertheless, while executing the AA algorithm, starting times of activities can be estimated better, but with some limitations. In the previous example, as outsiders in AA are treated in time order for convenience, when treating outsider c, all the actions preceding action c have already been treated (in this case, action d, since actions b and a are in the IAM of activity A). AA has already determined that action d pertains to activity A. This means that the start of activity A, which is the previous activity for action c, has been fixed and it is different from the start time provided by SA^3 ($t^{AA}_{start} \neq t^{SA^3}_{start}$).

In contrast with time metrics 1 and 2, where activity start and end were given by SA^3 and activity duration was assumed to be located symmetrically around the centre of the activity, the third time metric uses the start time of the previous activity as found by AA. Afterwards, the centre of the activity is calculated projecting the duration from the starting point. This makes the previous activity treatment more accurate. However, notice that the same approach cannot be applied to the next activity, since it has not been treated yet. The estimation given by SA^3 cannot be improved, because surrounding activities have not been treated when action c is being analysed. Hence, the best guess is to keep using start and end times provided by SA^3.

4.2.3 The complete AA algorithm

To sum up, the AA algorithm takes the results of SA^3 as the initialisation step. First, it treats insiders for all the activities detected by SA^3, using the compatibility function (Equation 4.5). Afterwards, it treats outsiders in time order, using the candidate function (Equation 4.6) and defined three time metrics (Equations 4.8, 4.9 and 4.10). The complete algorithm is depicted as pseudo-code in Algorithm 2.

The algorithm returns two files: (i) a fully labelled sensor activation dataset, which is formatted in a CSV file in the current implementation, and (ii) a list of clusters, which has been implemented through a JSON file. Additionally, that file contains: (i) the frequency of each action cluster, (ii) duration statistics (mean and

Algorithm 2 *AA* algorithm for action aggregation

```
Require: partially_annotated_dataset, context_knowledge
Ensure: fully_annotated_dataset, activity_clusters_file
  $fully\_annotated\_dataset \leftarrow createDataset(partially\_annotated\_dataset)$
  $insiders \leftarrow obtainInsiders(partially\_annotated\_dataset)$
  $outsiders \leftarrow obtainOutsiders(partially\_annotated\_dataset)$
  for all $action \in insiders$ do
    $activity \leftarrow obtainWrappingActivity(action, partially\_annotated\_dataset)$

    if $Comp(activity, action, context\_knowledge)$ then
      $label(a, activity, fully\_annotated\_dataset)$
    else
      $label(a, None, fully\_annotated\_dataset)$
    end if
  end for
  for all $action \in outsiders$ do
    $previous\_activity \leftarrow obtainPreviousActivity(action, partially\_annotated\_dataset)$

    $next\_activity \leftarrow obtainNextActivity(action, partially\_annotated\_dataset)$

    $Prev \leftarrow Cand(action, previous\_activity, context\_knowledge)$
    $Next \leftarrow Cand(action, next\_activity, context\_knowledge)$
    if $\neg Prev \wedge \neg Next$ then
      $label(a, None, fully\_annotated\_dataset)$
    else if $Prev \wedge \neg Next$ then
      $label(a, previous\_activity, fully\_annotated\_dataset)$
    else if $\neg Prev \wedge Next$ then
      $label(a, next\_activity, fully\_annotated\_dataset)$
    else if $Prev \wedge Next$ then
      //Use time metrics
      $prev\_dist \leftarrow timeDist(action, previous\_activity, time\_metric)$
      $next\_dist \leftarrow timeDist(action, next\_activity, time\_metric)$
      if $prev\_dist \leq next\_dist$ then
        $label(a, previous\_activity, fully\_annotated\_dataset)$
      else
        $label(a, next\_activity, fully\_annotated\_dataset)$
      end if
    end if
  end for
  $activity\_clusters \leftarrow createClusters(fully\_annotated\_dataset)$
  return $fully\_annotated\_dataset, activity\_clusters$
```

standard deviation), (iii) frequencies of used objects and actions and (iv) frequencies of locations per activity.

An example of the fully annotated dataset can be seen in Table 4.2. AA takes the partially annotated dataset generated by SA^3 (columns Activity* and Start/End*) and adds two new columns: the label assigned by AA (Activity) and the start and end tags for activities (Start/End). Table 4.2 is taken from a real experiment and shows very well the difference between SA^3 and AA. The action sequence depicted shows a particular execution of the MakePasta activity, where due to sensor errors, a *rcontrolSens* activation occurred. It can be seen how SA^3 detects that a MakePasta activity is being performed in the action sequence. But activity start and end times differ substantially from the real ones. However, AA detects perfectly activity start and end times, using the outsider action treatment process. And for the insider action *hasRemoteControl*, AA assigns a None label, since the action is not type compatible with the activity. So in this case, AA completes its work perfectly, assigning the appropriate labels to every action and enhancing the results obtained by SA^3.

Timestamp	Sensor	Action	Activity*	Start/End*	Activity	Start/End
2014-05-23 13:34:09	storeSens	openStore	None		MakePasta	start
2014-05-23 13:34:15	potSens	useCookingUtensil	MakePasta	start	MakePasta	
2014-05-23 13:34:35	ktapSens	turnOnTap	MakePasta		MakePasta	
2014-05-23 13:35:01	cookerSens	useCookingAppliance	MakePasta		MakePasta	
2014-05-23 13:35:59	rcontrolSens	hasRemoteControl	MakePasta		None	
2014-05-23 13:36:15	macaroniSens	hasPasta	MakePasta	end	MakePasta	
2014-05-23 13:45:15	drainerSens	useCookingUtensil	None		MakePasta	
2014-05-23 13:45:54	fridgeSens	openFridge	None		MakePasta	
2014-05-23 13:46:00	baconSens	hasBacon	None		MakePasta	
2014-05-23 13:46:05	creamSens	hasCream	None		MakePasta	end

Table 4.2: Example of a fully annotated dataset, the output of the AA algorithm shown in table format. To visualise better, timestamps have been shortened. The asterisk refers to the result given by SA^3.

An example of the activity clusters file has already been shown in Figure 3.6. As it can be seen in that figure, several concepts per activity are stored in the file:

1. Locations: a list of different locations where the activity has been performed with occurrence frequencies. Locations are obtained using activity location inference.

2. Actions: a list of actions executed by the user to perform the activity with associated occurrence frequencies.

3. Patterns: patterns refers to different action sequences performed by the user for the activity, i.e. the action clusters detected by the clustering process. Every detected cluster has its occurrence frequency.

4. Objects: a list of objects used by the user while performing the activity. Once again, every object has its occurrence frequency.

5. Occurrences: the total number of instances of the activity captured by the clustering process, which has to be equal to the sum of all the occurrences of patterns.

6. Duration: the mean duration of the detected instances of the activity and the standard deviation, both in seconds.

This information will be used by the Activity Model Learner (AML) algorithm to learn EAMs from the clusters extracted by the clustering process. Additionally, the information stored in the activity clusters file may be very useful to domain experts, since they may be able to analyse how activities are carried out by different users in terms of the executed actions, used objects, different sequences of actions and duration. The potential applications of such information, specially considering that it can be updated periodically to monitor the evolution of the user, is out of the scope of this dissertation, but it can open the doors to detect cognitive impairment and its evolution, inappropriate behaviour or user bad habits.

4.3 Summary and Conclusions

A two-step clustering process has been developed and implemented in this chapter, in order to detect and identify activities in a sensor activation dataset using the context knowledge provided by a domain expert. The first step of the clustering process has been devoted to find the occurrences of IAMs in an unlabelled sensor activation dataset. For that purpose, a novel pattern recognition algorithm has been developed, which uses IAMs as patterns, but also takes into account duration, location and completeness criteria. The algorithm is called SA^3 and it can work in scenarios where actions can appear in varied orders and both positive and missing sensor noise exist.

The initial clusters detected by SA^3 are then expanded in the second step of the clustering process. The AA algorithm distinguishes between insider and outsider actions. For the former group of actions, a compatibility function has been defined to decide whether an insider belongs to its wrapping activity, whereas for the latter group, previous and next activities are analysed in terms of compatibility and time feasibility. For those actions which can be aggregated to the previous and the next activities, three time metrics have been defined. Using those metrics, the actions are aggregated to the time closest activity.

The results of the clustering process are finally stored in two files: the fully annotated dataset, where every sensor activation has an activity label, and the activity clusters file, where different clusters for each activity are depicted alongside other information. Even though the clustering process has to be understood inside the global solution designed to learn extended activity models, it can also be used for further objectives. For instance, it can be used to annotate sensor activation datasets. Notice that manual methods are usually used for annotating datasets that are used for activity recognition applications. However, manual annotation has a lot of problems, namely it is prone to errors and it is very time consuming, as shown by Rashidi and Cook in (Rashidi and Cook, 2011). Hence, the developed activity clustering process for unlabelled sensor activation datasets can be used as an activity annotator method, as far as the required previous knowledge can be provided.

The beautiful thing about learning is that nobody can take it away from you.

B.B. King

CHAPTER

5

Activity Model Learner

THE objective of this chapter is to describe and analyse the proposed learning algorithm for specialised and complete activity models which is called Activity Model Learner (AML). AML uses the results obtained by the activity clustering process described in Chapter 4, where different action sequences for every activity are identified in the form of clusters. The aim of AML is to learn extended activity models from the information given by the clustering process.

The learning algorithm presented in this chapter is a statistical algorithm. It uses the similarity between action sequences describing the same activity to identify spurious action sequences, based on statistical outlier detection techniques. As real world scenarios contain sensor errors and as the clustering process may not always label correctly the actions, some of the clusters provided to AML will not be valid. More concretely, some of the clusters will be spurious variations of the actual activity models for a given activity. AML detects those spurious action sequences and fuses them with the most similar action sequence which has been considered valid. The result of such a process is a set of specialised and complete activity models for every activity defined in the context knowledge.

The chapter is divided in four sections: Section 5.1 states clearly and concisely the objectives of the learning algorithm and the criteria followed to design it. Section 5.2 discusses the relevance of all the information provided by the clustering process to learn extended activity models, in order to identify what information can be used and why. Section 5.3 describes in detail the Activity Model Learner algorithm. Section 5.4 provides a summary of the chapter and presents the most relevant conclusions.

5.1 Objectives

The main objective of the Activity Model Learner is to learn extended activity models (EAM), which have been defined in Definition 7. An EAM of an activity represents complete and specialised models of the activity for a concrete user. Assuming that different users execute different actions to perform the same activity and that a concrete activity may be performed in different ways by the same user, a user adaptable activity modelling approach has to be able to capture activity models that fulfil these requirements. For example, making a coffee is a common activity for many users. However, a concrete user may prepare a coffee with milk sometimes and a black coffee other times. Both, coffee with milk and black coffee, are specialised sub-activities of making coffee, which will be characterised by different action sequences. So the objective of AML is to learn those different action sequences for a concrete user based on the output given by the clustering process described in Chapter 4.

Problem (AML)**.** *Given the activity clusters and fully annotated dataset files coming from the activity clustering process, learn extended activity models, i.e. complete and specialised activity models for a concrete user.*

Notice that in some cases, a user may only perform an activity in a single way. In that case AML would learn only one complete activity model, since for a model to be considered a specialised model, at least two complete models have to be learnt.

When extended activity models are learnt, they are presented to the domain expert, mainly due to two reasons: (i) the specialised models do not have a semantic tag as sub-activities of the detected activity - e.g. sub-activities MakeBlackCoffee and MakeCoffeeWithMilk may be learnt, but their names or semantic tags cannot be known by the learner - and (ii) to let the expert analyse and add the learnt models to the main activity ontology if convenient. The first reason refers to the fact that the AML cannot know whether an specialised model refers to making a black coffee or making coffee with milk. Nevertheless, having the specialised activity model, it is usually easy for an expert to decide the semantic label of the specialised activity model. The second reason is derived from the fact that the AML will not be able to deliver without any error, specially considering the limited previous knowledge provided and the noisy sensor scenario in which it has to work. An expert will analyse the EAMs learnt by the AML to filter them if necessary and add them to the activity ontology with appropriate specialised activity labels.

Based on the role of the domain expert and assuming that the action clusters obtained in the clustering process do contain all the activity variations for a concrete user, a conservative policy for activity learning is the best option. In this context, conservative means that it is better to learn false activity models as far as all real activity models are learnt. If an activity variation detected by the clustering process is removed in the learning stage, the expert will no be able to recover that variation from the information provided by the AML. However, false activity models can be purged by the expert, so the objective of the AML is to learn the 100% of real activity models, minimising the number of spurious models learnt.

5.2 Identifying Relevant Information

To address the problem addressed in Section 5.1, the information that the AML can use is the one stored in the output of the clustering process (activity clusters and fully annotated dataset) and the context knowledge file. Notice that the information from the context knowledge has already been exploited in the clustering process (Chapter 4), both by SA^3 (Semantic Activity Annotation Algorithm explained in Section 4.1) and AA (Action Aggregator, described in Section 4.2), so it cannot offer any new relevant information for AML. In other words, the output of the clustering process cannot be further analysed and refined using the information from the context knowledge file.

Thus, the AML can only rely on the activity clusters file and the fully annotated dataset. It is important to identify what information can be obtained from those files and whether they are relevant in order to learn EAMs from the action clusters which define activities. Basically, the activity clusters file is obtained as a summary of the fully annotated dataset. The only information which is not stored in the activity clusters file and it is in the fully annotated dataset is the start and end times for concrete executions of activities. Notice that the mean duration and standard deviation are calculated for each activity (not for each activity variation) and stored in the activity clusters file. However, all the clusters in the activity clusters file have been obtained applying the time feasibility $InRange$ function (Equation 4.6), so all of them are feasible models from the duration point of view. Hence, concrete activity durations are not relevant for AML and in consequence, the fully annotated dataset will not be used as an input to AML.

The activity clusters file is then the only input to the AML algorithm. The information provided in this file has already been shown in Figure 3.6 and described in Section 4.2.3. Now, this information will be analysed to assess its relevance to learn EAMs:

1. Locations: the locations contained for each activity are compatible with the locations defined in the context knowledge. The relative occurrences of those locations are not relevant to learn EAMs. They can be used to learn descriptive properties, for instance, but this is not the objective of AML, so locations will not be used.

2. Actions: the list of all actions used and their frequencies may serve to identify actions that are executed very rarely for a given activity. However, even being rare, an action could really be executed for a given activity by a concrete user. In those cases, the conservative policy selected dictates that frequency information of actions cannot be directly used to discard any of them.

3. Patterns: patterns represent the action sequences (clusters) with their associated frequencies. They contain all the varied ways a user performed an activity, so they are the key element to learn EAMs.

4. Objects: the AML aims to learn activity models as sequences of actions. The concrete objects used to execute those actions are not relevant. Objects may be used to learn descriptive properties, but not action properties, so they will not be used for the AML.

5. Occurrences: the total number of clusters for a given activity. It does not provide any significant information to learn EAMs.

6. Duration: it can be used to learn descriptive properties and update duration information provided by domain experts in the context knowledge. However, it cannot be used to learn action properties.

In consequence, the information stored in patterns is the only relevant information for the AML. Remember that patterns store all the different action sequences extracted by the clustering process for a given activity with associated frequencies.

5.3 The Learning Algorithm

Summing up the conclusions obtained in Sections 5.1 and 5.2, the task of learning EAMs can be seen as purging spurious action clusters from all the clusters extracted by the clustering process. The information that the AML can use for this task is the action clusters themselves and associated occurrence frequencies. So the main idea is that action clusters contain all the action sequences performed by a user for each activity, but some of those clusters are spurious, due to sensor noise, user erratic behaviour and/or clustering errors. The AML has to detect those spurious action sequences or outliers and remove them in order to keep only those action sequences that really represent an activity.

The designed learning algorithm works on the similarity between action sequences and their occurrence frequencies. A two-step process has been designed and implemented for that purpose: (i) the filtering step (Section 5.3.1), where action sequences will be treated in terms of repeated actions and varied order of actions, and (ii) the similarity-based outlier detection step (Section 5.3.2), where outlier action sequences will be detected in the similarity space and fused with the most similar action sequence taking their occurrence frequencies into account. The complete AML algorithm and its final output are described in Section 5.3.3.

5.3.1 The filtering step

If action clusters which define activities are carefully analysed, it can be seen that there are some clusters that contain repeated actions. Additionally, some clusters contain the same actions but in different order. In the ontology-based activity modelling approach described in Section 3.1, the activity models in terms of action properties do not contain repeated actions and the order in which actions are executed is not important (notice that descriptive properties do reflect this information, but they are not used when recognising activities). In consequence, two filtering steps are performed:

1. Remove repeated actions into a sequence: some action sequences contain repeated actions. For instance, consider the sequence $S = \{a, b, a, c\}$. As repeated actions do not add any new information for activity modelling, they are removed. S becomes $S' = \{a, b, c\}$. Notice that this step does not remove any action sequence of an activity. This step can be omitted depending on how activity models are used by the activity recognition system. The work presented in this paper is based on the knowledge-driven approach developed by Chen et al. in (Chen et al., 2012b). This activity recognition system does not consider repeated actions in the recognition phase, so activity models do not have this information. However, if the recognition system is sensitive to repeated actions, the learning algorithm can be easily modified to omit this first filtering step.

2. Fuse equal action sequences: some action sequences contain the same actions, but in different order. For example, $S_1 = \{a, b, c\}$ and $S_2 = \{b, c, a\}$. The order of actions is not important for activity models, so both sequences are fused. Two action sequences are equal if:

$$S_1 \cap S_2 = S_1 \cup S_2 \tag{5.1}$$

   Any two action sequences that fulfil Equation 5.1 are fused. Fusing implies removing one of the action sequences and to sum occurrence frequencies of both action sequences to assign it to the fused sequence. Thus from two sequences a resultant sequence will remain, reducing the number of valid action sequences.

As a consequence of the filtering step, the number of clusters for an activity will be equal or less than the clusters provided as input.

5.3.2 The similarity-based outlier detection step

The idea behind the similarity-based outlier detection is simple: all the action sequences describing the same activity will have their own similarity distribution depending on the varied ways of performing the activity by a concrete user. Spurious action sequences will be those which are very similar to valid action sequences, and their difference will be related to sensor noise or clustering errors. Hence, if outliers are detected in the similarity space between action sequences, those outliers will point to spurious action sequences.

The first thing needed to apply the idea introduced above is a similarity measure for two action sequences. The literature offers a lot of similarity measures to compare two sequences. However, many of them take into account the order of the elements of both sequences, such as the Levenshtein distance (Levenshtein, 1966), Hamming distance (Hamming, 1950) or most frequent k characters (Seker et al., 2014). As explained in Section 5.3.1, the order of actions is not relevant for ontology-based activity modelling, thus order-based similarity measures are not useful to detect outlier action sequences.

There are two similarity measures that do not care about the order of the elements of sequences, namely the Jaccard coefficient (Jain and Dubes, 1988) and the Sørensen-Dice coefficient (Sørensen, 1948). The Jaccard coefficient measures similarity between finite sample sets, and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$Jaccard(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.2}$$

It can be seen that $Jaccard(A, B) \in [0, 1]$. Moreover, if sequences A and B are equal, $Jaccard(A, B) = 1$. If A and B are empty sequences the Jaccard coefficient is defined to be 1.

The Sørensen-Dice coefficient is very similar to the Jaccard coefficient. It is denoted as QS and defined as the size of the intersection multiplied by two divided by the sum of the elements of both sequences:

$$QS(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{5.3}$$

Similarly to Jaccard coefficient, $QS(A, B) \in [0, 1]$. Since it does not satisfy the triangle inequality, the Sørensen-Dice coefficient can be considered a semimetric version of the Jaccard coefficient. This coefficient is mainly useful for ecological community data. Justification for its use is primarily empirical rather than theoretical, although it can be justified theoretically as the intersection of two fuzzy sets (Roberts, 1986).

For the outlier detection step, the Jaccard coefficient has been selected, because it is a theoretically well founded measure which can be seen as the basis of the Sørensen-Dice coefficient. So the similarity between two action sequences is given by the Jaccard coefficient of both sequences. The value 1 denotes that both sequences are equal, whereas 0 means that there is nothing in common between both sequences. Notice that due to the filtering step (Section 5.3.1) there will not be equal action sequences at this stage, hence value 1 will not appear. Similarly, having a 0 value is also impossible, since all the action sequences describing the same activity share at least the actions contained in their IAMs.

The similarity-based outlier detection is an iterative algorithm which uses the Jaccard coefficient to calculate the similarity between action sequences. The algorithm calculates the so called *Jaccard Matrix* (JM), which is a square matrix of all action sequences that remain from the previous iteration (in the case of the first iteration, action sequences coming from the filtering step in Section 5.3.1). $JM_{i,j}$ stores the Jaccard coefficient for action sequences S_i and S_j.

$$JM = \begin{pmatrix} Jaccard(S_0, S_0) & \cdots & Jaccard(S_0, S_N) \\ \vdots & \ddots & \vdots \\ Jaccard(S_N, S_0) & \cdots & Jaccard(S_N, S_N) \end{pmatrix} \tag{5.4}$$

The diagonal of JM is 1, since two equal action sequences' Jaccard coefficient is 1 ($\forall i, j : i = j \to Jaccard(S_i, S_j) = 1$). Diagonal values do not provide any information for outlier detection, so they are removed. Additionally, values from the upper and lower triangles of JM are the same, since $Jaccard(S_i, S_j) = Jaccard(S_j, S_i)$. So extracting only the values of the upper triangle from JM, $J\hat{M}$ is obtained, which is stored as a vector. The $J\hat{M}$ vector has the information of the similarity of all action sequences detected for a concrete activity. Figure 5.1 shows a plot of the similarity values between 11 action sequences, taken from a real example (they are representing the activity MakePasta for a concrete user). The idea of the similarity-based outlier detection is to identify similarity values that stand out among everyone else; then, take the action sequences that are represented by those outlier similarity values and fuse them.

To detect outlier similarity values, a statistical approach has been defined. Fixing a static threshold value for every activity is not appropriate, since some activities may contain many actions, whereas others are performed by means of few actions. So a difference of a single action in a "short" activity cannot have the same weight as in a "long" activity (short and long are here used to refer to the number of actions of activities). Furthermore, some users may execute very different action sequences to perform the same activity, while others may only add slight variations. In consequence, the outlier detection method has to calculate a threshold for every action sequence group describing an activity.

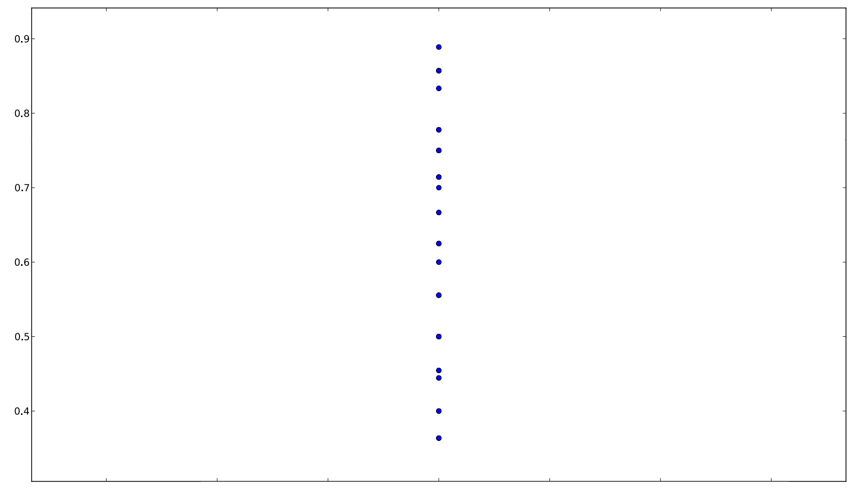


Figure 5.1: Jaccard coefficient-based similarity values for 11 action sequences detected for a concrete activity and a concrete user.

The strategy followed to calculate the threshold is based on the median value of the $J\hat{M}$ vector. The median is the numerical value separating the higher half of a data sample from the lower half. The median of a finite list of numbers can be found by arranging all the observations from lowest value to highest value and picking the middle one. If there is an even number of observations, then there is no single middle value; the median is then usually defined to be the mean of the two middle values. The median has been chosen rather than the mean, because the median is a robust estimator which is less influenced by outliers than the mean value. Using the median allows detecting the similarity "centre of gravity" minimising the influence of outliers.

Once the median of $J\hat{M}$ measures the central tendency, another statistic is needed to capture the dispersion of similarity in $J\hat{M}$. Two good candidates to measure the dispersion in univariate datasets have been found in the literature (Hoaglin et al., 1983). The first one is the median absolute deviation (MAD) and it is defined as the median of the absolute deviations from the data's median:

$$MAD = median_i(|X_i - median_j(X)|) \tag{5.5}$$

The MAD is a measure of statistical dispersion. Moreover, the MAD is a robust statistic, being more resilient to outliers in a dataset than the standard deviation. In the standard deviation, the distances from the mean are squared, so large deviations are weighted more heavily, and thus outliers can heavily influence it. In the MAD, the deviations of a small number of outliers are irrelevant.

There is another dispersion measure, which is known as the average absolute deviation about median (ADM). The ADM offers a direct measure of the scale of a random variable about its median and it is defined as:

$$ADM = E_i|X_i - median(X)| \tag{5.6}$$

where $E_i|X|$ stands for the mean value of X. Both the MAD and the ADM can be used to establish a proper threshold and identify outlier similarity values. However, in the current implementation the ADM has been chosen. It is true that the ADM is more influenced by outliers than the MAD, but from the point of view of the conservative policy selected for learning, the ADM is more suitable. The reason is that the MAD can have a value of zero if there are some similarity values with the same values as the median. This fact makes that the median itself could be taken as the threshold, which is quite aggressive. On the other hand, the only way the ADM can be zero is to have all similarity values being equal to the median. Moreover, it can be shown mathematically that the MAD of a given dataset will always be less or equal to the ADM, so the MAD is more aggressive in all the cases.

Due to those reasons, the ADM has been chosen to establish a statistical threshold to separate normal similarity values from outlier values. The threshold is defined as:

$$\theta_S = median(J\hat{M}) + ADM(J\hat{M}) \tag{5.7}$$

As the threshold in Equation 5.7 has been calculated using statistical approaches, it is denoted as θ_S. Applying Equation 5.7 to the similarity values plotted in the example of Figure 5.1, a threshold can be calculated. This threshold is depicted in Figure 5.2. Similarity values above the threshold can be considered as outliers.

Statistical approaches work well when "a lot of" data is available. But when "few" data is available, statistic-based approaches may extract weak conclusions. For example, imagine the following action sequences are being analysed by AML for the activity MakePasta (this example is extracted from a real experiment):

$$\begin{aligned} S_1 = \{&turnOnTap, useCookingAppliance, hasPasta, \\ &useCookingUtensil, hasPesto\} \\ S_2 = \{&hasPasta, useCookingAppliance, turnOnTap, \\ &hasBacon, hasCream, useCookingUtensil\} \\ S_3 = \{&turnOnTap, useCookingAppliance, hasPasta, \\ &useCookingUtensil, hasTomato\} \end{aligned}$$

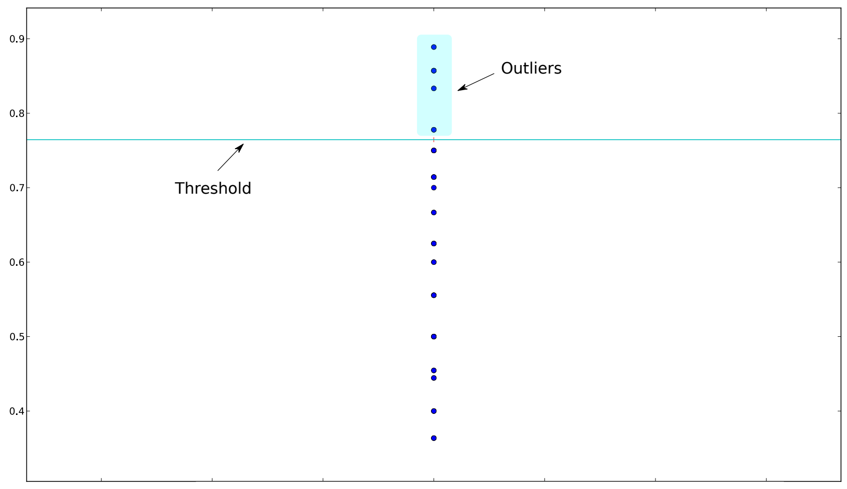


Figure 5.2: The threshold calculated for the similarity values of 11 action sequences representing a concrete activity.

The difference between S_1 and S_3 is that the user adds pesto in the first and tomato in the second, i.e. only one action out of five. All three action sequences are really performed by the user and they should not be removed or fused. However, if θ_S is calculated for the group of action sequences, a value of 0.6 is obtained. Considering that $Jaccard(S_1, S_3) = 0.66$, action sequences S_1 and S_3 should be fused, thus removing one of the action sequences which has actually been performed by the user. Notice that detecting outliers using statistical approaches on three samples is not generally feasible. Removing one of the action sequences would contradict the conservative policy selected for the AML, where the priority is given to learning all real action sequences performed by a user. Hence, a defensive mechanism for those cases is adopted. The threshold to identify outliers is defined as:

$$\theta = max\{\theta_S, \lambda\} \tag{5.8}$$

where λ is a fixed value. The idea of this approach is to state a similarity value that acts as a minimum value for action sequences to be fused. In the experiments performed in this dissertation $\lambda = 0.75$ (Chapter 6). If λ is set to higher values, the AML will be even more conservative, whereas decreasing λ will make the AML rely more on the statistic-based approach (θ_S). Figure 5.3 shows both thresholds, θ_S and λ, for the example of MakePasta activity and its three action sequences. Notice that $Jaccard(S_1, S_2) = Jaccard(S_2, S_3)$ and thus, only one point is depicted in the graphic. It can be seen clearly how using only θ_S, $Jaccard(S_1, S_3)$ is an outlier and in consequence, sequences S_1 and S_3 would be incorrectly fused. But using Equation 5.8 and taking the maximum of both thresholds, there is not any outlier and hence all three sequences are kept as valid activity models.

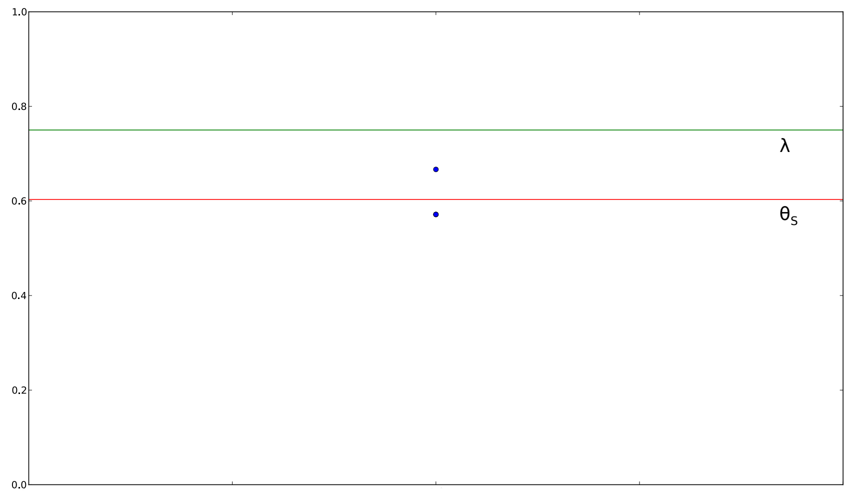


Figure 5.3: Both thresholds depicted in an example of few available data: θ_S and λ.

The similarity-based outlier detection algorithm is an iterative process that fuses sequences whose Jaccard coefficient is higher than θ, until no sequences can be fused. Fusing implies defining a function which returns a single action sequence having as input two different action sequences. In the case of the AML a fusing heuristic has been adopted, which states that given two action sequences to be fused, the lower frequency sequence is a spurious variation of the higher frequency sequence. In consequence, the higher frequency action sequence is the valid one and the function will return it as the fused action sequence.

$$Fusion(S_i, S_j) = \begin{cases} S_i, & \text{if } f_{S_i} \geq f_{S_j} \\ S_j, & \text{otherwise} \end{cases} \tag{5.9}$$

where f_{S_i} is the occurrence frequency of action sequence S_i. The fused action sequence will have a new occurrence frequency which is calculated as the sum of the occurrence frequencies of fused sequences: $f_{S_{fused}} = f_{S_i} + f_{S_j}$.

5.3.3 The complete AML algorithm

Combining the filtering step described in Section 5.3.1 and the similarity-based outlier detection step described in Section 5.3.2, the complete AML algorithm can be built. Its inputs are the activity clusters file, where the result of the clustering process is given, and the fixed threshold λ. Algorithm 3 shows the AML algorithm written in pseudo-code.

Algorithm 3 AML algorithm for learning extended activity models

```
Require: activity_clusters, $\lambda$
Ensure: learnt_action_sequences
  $activities \leftarrow obtainActivities(activity\_clusters)$
  for all $activity \in activities$ do
    $action\_sequences \leftarrow obtainSequences(activity, activity\_clusters)$
    // The filtering step
    $action\_sequences \leftarrow filteringStep(action\_sequences)$
    // The similarity-based outlier detection step
    repeat
      $JM \leftarrow jaccardMatrix(action\_sequences)$
      $\hat{JM} \leftarrow extractUpperTriangle(JM)$
      $\theta_S \leftarrow calculateStatisticThreshold(\hat{JM})$
      $outliers \leftarrow detectOutliers(\hat{JM}, \theta_S, \lambda)$
      $action\_sequences \leftarrow fuseSequences(action\_sequences, outliers)$
    until $outliers = \emptyset$
    $learnt\_action\_sequences \leftarrow addActivityModel(activity, action\_sequences)$
  end for
  return $learnt\_action\_sequences$
```

The output of the AML algorithm is the so called learnt action sequences file, stored in a JSON file in the current implementation. This file contains the extended activity models for every activity defined in the context model. An example can be seen in Figure 5.4, where two complete and specialised models for activity MakePasta are provided. The example is extracted from a real experiment, where the user prepares pasta *carbonara* more or less the 50% of times and pasta with tomato sauce also around 50% of times. Looking at the action sequences, the different ways of preparing pasta can be distinguished. In the first way, labelled by the user as pasta *carbonara*, bacon and cream are used by the user (actions *hasBacon* and *hasCream*). In the second way, action *hasTomato* appears, which makes reference to the usage of tomato sauce. Notice also that the first action sequence contains an *openFridge* action, which suggests that some of the ingredients are stored in the fridge. This does not happen in the second action sequence, where all the ingredients are taken from the food store (action *openStore*).

```
"MakePasta": {
    "1": [
            0.5074626865671642,
            [
                  "hasPasta",
                  "useCookingAppliance",
                  "hasBacon",
                  "turnOnTap",
                  "openFridge",
                  "openStore",
                  "hasCream",
                  "useCookingUtensil"
             ]
          ],
    "2": [
            0.49253731343283574,
            [
                  "hasTomato",
                  "hasPasta",
                  "useCookingAppliance",
                  "turnOnTap",
                  "openStore",
                  "useCookingUtensil"
            ]
          ]
    },
```

Figure 5.4: Example of the output of the AML algorithm. Two specialised and complete models for MakePasta activity are depicted. As their semantic tags are not known, they are tagged with numbers 1 and 2.

5.4 Summary and Conclusions

An algorithm to learn extended activity models has been designed and developed in this chapter. The algorithm takes the results of the clustering process described in Chapter 4, assuming that the action clusters have all the action sequences executed by a user to perform a concrete activity. Based on this assumption, it has been shown that learning extended activity models from action clusters can be seen as purging spurious action sequences which appear due to sensor noise, clustering errors and/or user erratic behaviour (Section 5.1). It is very important to highlight the conservative policy adopted by the AML algorithm. The highest priority is given to keep all real action sequences, even though this may imply not removing all spurious action sequences. The AML algorithm has been designed around this conservative policy.

Before describing the AML algorithm, all the available information has been analysed in order to identify what is relevant to learn extended activity models (Section 5.2). It has been shown that the action clusters per activity and their associated occurrence frequencies are the only relevant available information.

With the objective clearly defined and the relevant input information identified, Section 5.3 describes how the AML algorithm has been developed. It has two main steps: the filtering step (Section 5.3.1), where action sequences are filtered based on the repeated actions and their order, and the similarity-based outlier detection step (Section 5.3.2), where the Jaccard coefficient is used to compute the similarity between action sequences to calculate a threshold which allows distinguishing between inlier and outlier similarities; the action sequences whose similarity is higher than the threshold are fused, taking their occurrence frequencies into consideration. The complete AML algorithm and the output it generates have been described in Section 5.3.3.

One of the most surprising facts about the AML might be that it does not use the occurrence frequencies of action sequences in the outlier detection step. It can be argued that if spurious action sequences are generated by sensor noise, clustering errors and/or user erratic behaviour, their occurrence frequency should be relatively low. This is generally true. However, using frequencies to detect outliers has a pernicious effect: it can identify as spurious those action sequences that even being valid, have been executed few times by the user. Imagine, for instance, that a user prepares coffee frequently. Only in some special occasions, this user likes preparing Irish coffee. If frequency is taken into account for outlier detection, the action sequences describing the preparation of an Irish coffee might be removed. This contradicts the conservative policy adopted for the AML.

Indeed, the first metrics used for outlier detection combined Jaccard-based similarity and occurrence frequencies. It was observed that when the executed action sequences for a concrete activity had big differences in their occurrence frequencies, those with lowest frequencies were fused incorrectly with other action sequences. After defining and testing many metrics combining similarity and frequency, the conclusion was that frequencies should not be used for detecting outliers. As described in Section 5.3.2, AML uses frequency only in the fusion function (Equation 5.9) implementing the heuristic that for two action sequences to be fused, once their similarity has been identified as being an outlier, the lower frequency action sequence is a spurious variation of the higher frequency one.

The output of AML, stored in the learnt action sequences file (Figure 5.4), is presented to the domain expert, who can analyse the extended activity models extracted by the solution and add them to the knowledge base with appropriate activity names, if convenient. As such, a human expert will always have the last decision about the learnt activity models. That is one of the main reasons why the conservative approach for AML has been adopted. If an activity model is removed incorrectly, a human expert will not have any way to recover that activity model. But if a spurious activity model is learnt, the domain expert may remove it easily, looking at the action sequence and occurrence frequencies.

Science is a way of trying not to fool yourself. The first principle is that you must not fool yourself, and you are the easiest person to fool.

Richard Feynman

CHAPTER 6 Evaluation

THIS chapter describes and analyses the proposed evaluation methodology for the extended activity model learning system described through Chapters 3, 4 and 5. To test the learning system properly, detailed data from various users is needed, containing different ways of performing the same activity by the same user. In order to achieve such datasets, the proposed evaluation methodology is based on surveys to users and a synthetic dataset generator tool. Surveys allow capturing how users perform activities in terms of actions, while the synthetic dataset generator uses survey information to generate sensor activation datasets, introducing different kinds of sensor noise.

Once the methodology is described and discussed, evaluation scenarios and metrics will be introduced. To assess the performance of the learning system in various situations, several evaluation scenarios have been prepared, considering different kinds of sensor noise and set-ups. Similarly, and based on the available literature, the most significant performance metrics have been chosen to measure the performance of the approach.

Applying the evaluation methodology on the prepared scenarios and selected metrics, results are obtained. Those results are widely discussed in this chapter and compared to the objectives defined in Chapter 1. As the problem tackled in this dissertation has not been approached by any other researcher (see Section 1.1), comparing different approaches has not been possible.

The chapter is divided in three sections: Section 6.1 introduces the conventional evaluation methodology used for activity recognition, detects its drawbacks and proposes a new evaluation methodology which is more appropriate to validate the learning system. Section 6.2 presents evaluation scenarios, metrics, results and discussions divided in three main parts: SA^3 performance (Section 6.2.1), the complete activity clustering performance (Section 6.2.2) and finally the EAM learning system performance (Section 6.2.3). All three sections discuss the results and compare them with the objectives identified at the beginning of this dissertation (Chapter 1). To conclude, Section 6.3 provides a global view of the system performance, summarises the contents of the chapter and presents the most relevant conclusions.

6.1 Evaluation Methodology

Even though activity recognition is very diverse in terms of sensor approaches and algorithmic choices, evaluation is usually carried out applying a very well known methodology (Riboni and Bettini, 2011a), (Rashidi and Cook, 2011), (Tapia et al., 2004) which can be summarised in the following steps:

1. Choose a target environment and deploy sensors to acquire and process information about human activities.
2. Select a group of people who can perform target activities in the prepared environment.
3. Select a dataset labelling system so datasets generated by users can be used as a ground truth, both for the activity recognition system and the evaluation process.
4. Run experiments with users and label the obtained activity datasets.
5. Use the same datasets to test the activity recognition system and store the labels produced by it.
6. Compare the labels of the activity recognition system with the ground truth using appropriate metrics.

Each of the enumerated steps may vary depending on the activity recognition approach and the available resources. The described methodology, which will be called *standard methodology* for the rest of the dissertation, is conventionally the reference for any group working on human activity recognition.

Nevertheless, there are some problems that make very difficult to implement the so called standard methodology. For instance, (i) it is not always possible to own an environment and install sensors and processing systems, due to economic reasons, (ii) running experiments with human beings imply ethical and legal issues that can slow down the research process or sometimes make it impossible, (iii) dataset labelling systems are not perfect, since most of them rely on users' memory or discipline to annotate every activity carried out, and (iv) regulatory limitations on the use of human subjects prohibit the collection of extensive datasets that can test all scenarios and theories under all circumstances.

There are several research groups who have made the datasets that they obtain in their pervasive environments publicly available, following different variants of the standard methodology. A Wiki Page called BoxLab[1] contains a detailed list of public datasets collected in home environments. It contains a description of the deployed sensors, whether it is an annotated dataset and the availability of the dataset. Those public datasets can be a very good alternative for those research groups which cannot run their own experiments. Furthermore, public datasets can also be used for benchmarking different activity recognition systems.

6.1.1 Analysis of the viability of the standard methodology for the extended activity model learning system

To evaluate properly the EAM learning system, activity datasets that fulfil several conditions are needed. Namely:

1. Contain a lot of samples of different activities to enable a proper learning process. To make data-driven learning a meaningful process, many samples of the same activities are needed, specially to learn noise free activity models. If few samples are available and those samples are affected by noise, it is very difficult for the learning process to distinguish between noisy and noiseless models.

2. Sensor activations must be labelled in order to provide a solid ground truth. An accurate evaluation process needs an accurately labelled dataset, where all sensor activations produced due to an activity execution are annotated with that activity name, whereas sensor noise is not included in the activity.

3. Specific objects used for activities have to be monitored (dense sensing scenario). Instead of getting data from context sensors such as thermometers, humidity sensors or even movement sensors, datasets have to contain sensor activations monitoring concrete objects such as cups, dishes, food and so on, in order to be able to produce accurate activity models.

4. At least some of the activities have to be performed in varied ways to see whether the presented learning process can capture all those variations. Different ways of performing the same activity are needed, since the EAM learning system aims at capturing sub-activities. If all the activities are performed

[1]http://boxlab.wikispaces.com/List+of+Home+Datasets

the same way in terms of executed actions, the dataset will only serve to evaluate the learning of complete models, but not specialised models.

5. Several users are needed as the approach aims at capturing personalised activity models for concrete users. Every user has his/her own way to perform the same activities and one of the objectives of the EAM learning system is to be able to model those personalised models. Thus, to show that the EAM learning system can really deal with different users, datasets of several users are required.

Following the standard methodology to generate such datasets implies a high-cost and lengthy process with many ethical and legal constraints regarding the use of people for experiments. That cost, both economical and time cost, could not be assumed during the development of the current dissertation. Due to this limitation, public dataset repositories were analysed carefully to find an appropriate dataset.

Unfortunately, datasets that meet all the enumerated conditions could not be found in those public repositories. Some datasets used different activity monitoring approaches, such as vision- and wearable-based monitoring. Some other datasets could be used for the dense sensing scenario since they monitor specific objects, but activities are performed from a list where no variations can be found. Yet other datasets are more focused on monitoring the contextual information, rather than concrete user-object interactions.

As Helal et al. state in their paper (Helal et al., 2011):

> *Access to meaningful collections of sensory data is one of the major impediments in human activity recognition research. Researchers often need data to evaluate the viability of their models and algorithms. But useful sensory data from real world deployments of pervasive spaces are very scarce. This is due to the significant cost and elaborate groundwork needed to create actual spaces. Additionally, human subjects are not easy to find and recruit. Even in real deployments, human subjects cannot be used extensively to test all scenarios and verify multitudes of theories. Rather, human subjects are used to validate the most basic aspects of the pervasive space and its applications, leaving many questions unanswered and theories unverified.*

This is the case of the presented approach for EAM learning. It turns out that no datasets were found from real pervasive environments to verify the exposed theory. However, Helal et al. propose a solution to this problem in the same paper (Helal et al., 2011):

> *It is thus necessary to develop alternative and practical approaches to studying human activity recognition. Powerful and realistic simulation*

> *tools could be used to support the growing demand for test data. Simulations enable researchers to create focused synthetic replications of important events and activities under study. It can be easily changed and refined allowing researchers efficiently to experiment, analyze and fine-tune their models and associated algorithms. Simulation also allows a wider community of researchers to engage and collaborate to solve a specific problem. Hence, a design based on preliminary simulation studies would most likely to be a more robust and inclusive design. Also, a simulation model that mimics an existing real world pervasive space is most likely to answer more questions (and generate much more data) than the target actual space.*

Following those ideas, simulation tools have already been used for activity recognition. For example, Okeyo et al. use a synthetic data generator tool to simulate time intervals between sensor activations (Okeyo et al., 2012). Their research is focused on sensor data stream segmentation, so the tool generates varying patterns of sensor activations in order to verify their approach. Liao et al. combine simulation tools and real data for activity recognition in (Liao et al., 2006). A more elaborated simulator has been developed by Bruneau et al. in (Bruneau et al., 2009): DiaSim. The DiaSim simulator executes pervasive computing applications by creating an emulation layer and developing simulation logic using a programming framework. However, it is more focused on simulating applications such as fire situations, intrusions and so on to identify potential conflicts. In consequence, DiaSim cannot be directly applied to activity recognition. Finally, Helal et al. propose to develop powerful and realistic simulation tools to study human activity recognition. They develop a simulator called *Persim*, which has been enhanced in the new version *Persim-3D* (Helal et al., 2012). Persim is an event driven simulator of human activities in pervasive spaces. Persim is capable of capturing elements of space, sensors, behaviours (activities), and their inter-relationships. Persim is becoming a very complete simulator tool for activity recognition in pervasive environments. However, it is still under development and one of the main limitations is that it does not provide a way to model realistically human behaviour. Authors have already identified this limitation and they are currently working on programming by demonstration approaches to overcome the problem.

As can be seen in the literature review done in the paragraph above, simulation tools can be used for activity recognition, since they provide accurate enough datasets to verify some theories. However, none of the references given above specify a concrete methodology to use simulators to evaluate activity recognition approaches. There is no information about how activities should be defined, how different users can be modelled, sensor error models and so forth, which are key issues when using a simulator. Therefore, there is a lack of a sound methodology that addresses the usage of simulation tools for activity recognition evaluation.

6.1.2 Hybrid evaluation methodology approach

The hybrid evaluation methodology has been specially designed for activity recognition systems which assume the dense sensing paradigm (see Constraint 1). Even though the methodology itself is not limited to concrete scenarios, the implementation presented in this document works for single user - single activity scenarios, i.e. only one user is considered and concurrent or interleaved activities are not taken into account (see Constraint 2).

The methodology has been named hybrid because it combines real users' inputs and simulation tools. The key idea is to circulate surveys among target users with the objective of capturing how they perform certain activities of daily living. Additionally, users are also requested to describe how their days are in terms of defined activities. For example, a user might make a coffee and brush her teeth in week days between 7:00 and 7:30 AM. So the aim of those surveys is to model real human behaviour, covering one of the major weaknesses of simulation-based evaluation methodologies. Using the information collected by surveys, individual scripts are prepared, which are then processed by a synthetic dataset generator tool to simulate arbitrary number of days and generate perfectly labelled datasets of activities. To get as close as possible to real world settings, the synthetic dataset generator uses probabilistic sensor noise models and probabilistic time lapses.

Based on those constraints and ideas, the proposed hybrid evaluation methodology has the following steps (see Figure 6.1):

1. Design activity surveys: to capture how users perform activities and model their behaviour, a proper survey has to be designed. A detailed explanation of how surveys are designed for this dissertation can be found in Section 6.1.2.1.

2. Select target users: depending on the objectives of the research, several user groups can be selected. For example, if the system aims at providing help to elderly people, selecting members of that target group is recommended.

3. Distribute surveys among target users: a suitable way to distribute surveys has to be used, which guarantees users' anonymity. The distribution method can also be influenced by target users. For example, using web-based surveys can be a bad idea if surveys are directed to elderly people, who can be unfamiliar with those technologies. Personal interviews may be a good alternative for those cases.

4. Translate surveys to scripts: appropriate criteria have to be adopted to translate the answers obtained from surveys to scripts for the synthetic dataset generator - or any other simulator -. It is very important not to alter or lose the information provided by users.

5. Model sensor noise: sensor noise has to be modelled in order to achieve realistic activity datasets. Real sensors are not perfect and depending on their technological base, error models have to be provided.

6. Run synthetic dataset generator: using the scripts obtained from surveys and sensor error models, the synthetic dataset generator is executed. The output of the tool is a labelled activity dataset which will serve as the ground truth for evaluation.

7. Develop the activity modelling and/or recognition system: researchers have to develop the activity modelling and/or recognition system in order to be tested. Notice that datasets generated by the synthetic dataset generator can also be used in this step, specially for data-driven approaches.

8. Compare results: finally, the results obtained by the activity modelling and/or recognition system have to be compared with the ground truth, using appropriate metrics.

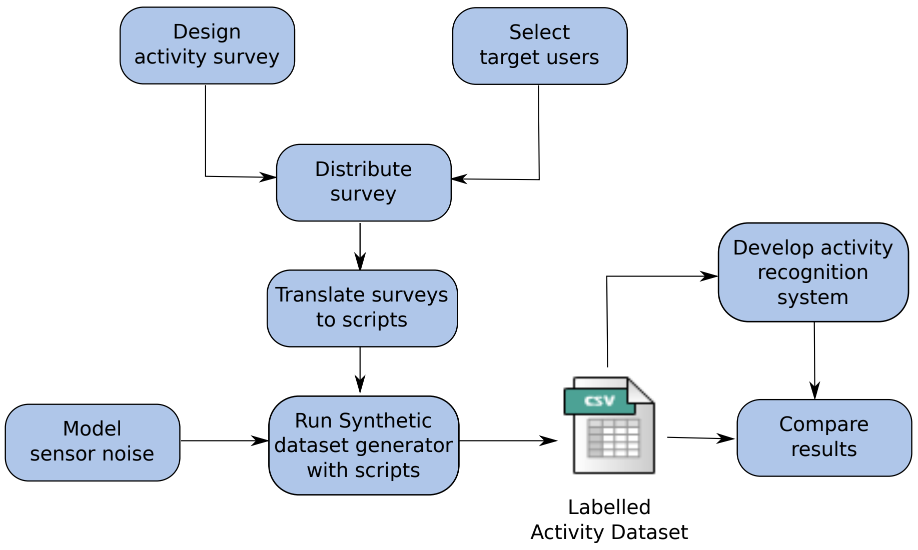

Design activity survey
Select target users
Distribute survey
Translate surveys to scripts
Develop activity recognition system
Model sensor noise
Run Synthetic dataset generator with scripts
CSV
Compare results
Labelled Activity Dataset

Figure 6.1: The hybrid evaluation methodology steps depicted in a flowchart.

6.1.2.1 Survey for activities of daily living

One of the main advantages of considering dense sensing scenarios is that activities are described in terms of the objects which have been used to perform that activity. Furthermore, as only sensor activations (and not de-activations) are important for the approach (see Definition 1), to model an activity, it is enough to know which objects are used by the user and the order of usage of those objects. This information is easy to obtain in a survey and will be named *activity model*.

Definition 13 (Activity model)**.** An activity model is a sequence of objects used by a user to perform an activity. A user might provide several activity models per each defined activity, because the same activity can be performed in several ways. Activity models also provide a typical duration given by the user.

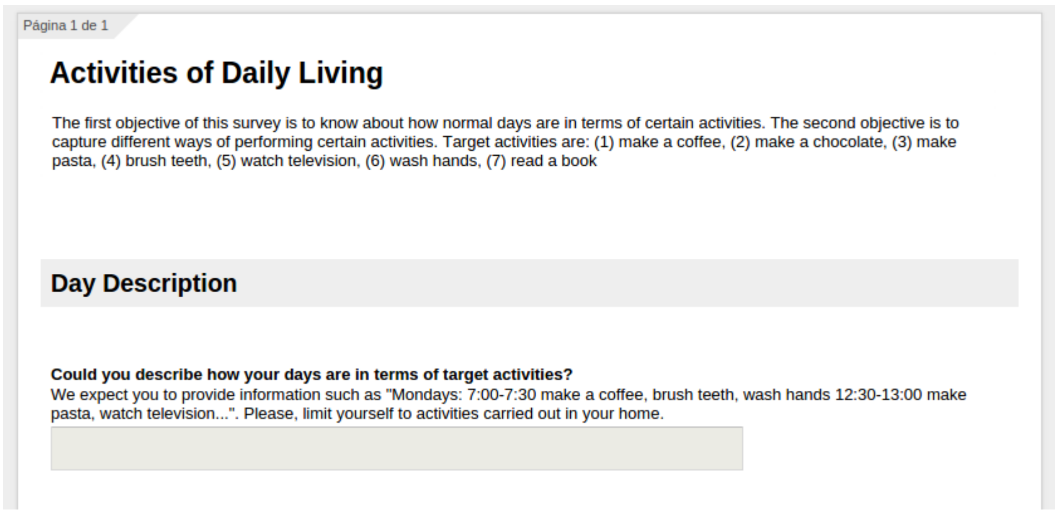
On the other hand, to model human behaviour appropriately, acquiring activity models is not enough. It is very important to know what activities are performed by a concrete user in a daily basis, alongside with the time slots and time lapses between contiguous activities.

Definition 14 (Behaviour model)**.** A behaviour model is a sequence of activities with associated time slots and time lapses. A user might provide several behaviour models, as every day can be different in terms of performed activities and times.

The main objective of the survey is to obtain activity and behaviour models from target users. Hence, the survey for activities of daily living has two main parts. The first part is devoted to capture what activities are performed in different days, i.e. behaviour models (see Definition 14). The second part, on the other hand, asks users about how they perform those activities based on user-object interactions, i.e. activity models. The concrete survey used for the experiments carried out in this dissertation can be found in the web[1].

As can be seen in Figure 6.2, the survey begins with a brief explanation for target users, where the aims of the survey are stated and the target activities are presented. In this case, the target activities are seven: (i) make a coffee, (ii) make a chocolate, (iii) make pasta, (iv) brush teeth, (v) watch television, (vi) wash hands and (vii) read a book. Afterwards, under the heading of "Day Description", users are asked to describe their week days in terms of activities. They are expected to provide information about time slots and activity sequences performed in those time slots. Users are also asked to provide time relations between two consecutive activities. For example, between 7:00 and 7:30 AM a user might make a coffee and ten minutes later might brush her teeth. This first part has been designed to obtain behaviour models for target users.

[1]http://goo.gl/etCNyi


Página 1 de 1

Activities of Daily Living

The first objective of this survey is to know about how normal days are in terms of certain activities. The second objective is to capture different ways of performing certain activities. Target activities are: (1) make a coffee, (2) make a chocolate, (3) make pasta, (4) brush teeth, (5) watch television, (6) wash hands, (7) read a book

Day Description

Could you describe how your days are in terms of target activities?
We expect you to provide information such as "Mondays: 7:00-7:30 make a coffee, brush teeth, wash hands 12:30-13:00 make pasta, watch television...". Please, limit yourself to activities carried out in your home.

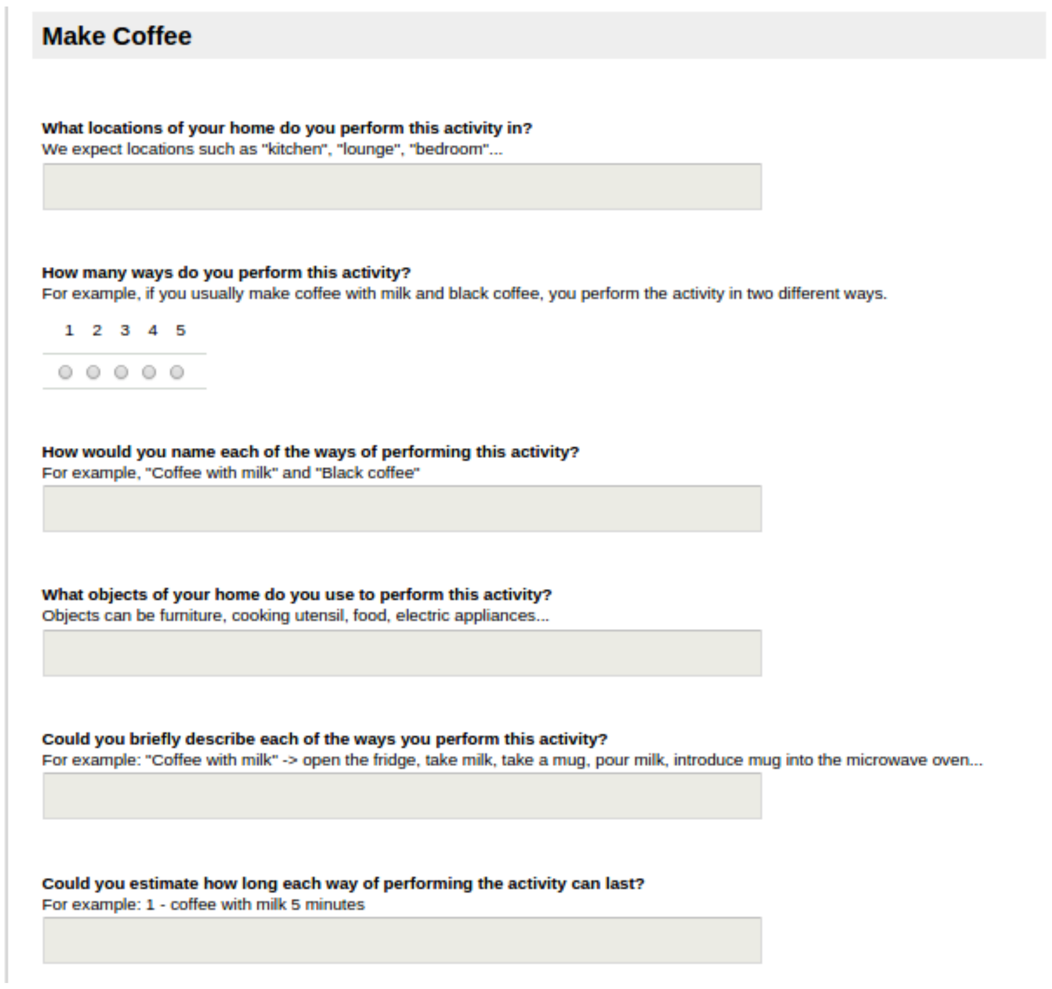
Figure 6.2: The first part of the survey. A brief introduction can be found where the aim of the survey is explained, continuing with the behaviour model part.

The second part of the survey is longer. Target activities are presented one by one. For each activity, several questions are asked to users, to capture the locations of activities, the ways activities are performed, the objects used for each activity, a description of how those objects are used and typical duration estimations. An example of those questions can be found in Figure 6.3 for the activity MakeCoffee.

As Figure 6.3 shows six questions are asked per activity. The first question is to know where the activity is performed by the user. As stated in the brief explanation under the question, expected locations are home locations such as kitchen, lounge and so forth. Each activity may be performed in several locations, in concordance with the context knowledge model shown in Figure 3.2. Notice also that the expected locations fulfil with the location definition given in Definition 5.

The second question deals with different ways of performing an activity, i.e. sub-activities. Users are asked to provide a variation name, which will define the name of the sub-activity. The next question asks about the objects used to perform the activity. This will serve not only to model the activity itself, but also to model the context knowledge, where objects and attached sensors are represented (see Section 3.3.1). Afterwards, the most important question for activity modelling comes: a description of how the enumerated objects are used to perform the activity. Descriptions are requested for each activity variation. From those descriptions, object usage order and time lapses will be obtained. Finally, the last question aims at modelling typical durations for the variations of the target activity.

As described in the steps of the hybrid evaluation methodology in Section 6.1.2, it is also important to decide the way to circulate the survey and to guarantee user



Make Coffee

What locations of your home do you perform this activity in?
We expect locations such as "kitchen", "lounge", "bedroom"...

How many ways do you perform this activity?
For example, if you usually make coffee with milk and black coffee, you perform the activity in two different ways.

1	2	3	4	5
○	○	○	○	○

How would you name each of the ways of performing this activity?
For example, "Coffee with milk" and "Black coffee"

What objects of your home do you use to perform this activity?
Objects can be furniture, cooking utensil, food, electric appliances...

Could you briefly describe each of the ways you perform this activity?
For example: "Coffee with milk" -> open the fridge, take milk, take a mug, pour milk, introduce mug into the microwave oven...

Could you estimate how long each way of performing the activity can last?
For example: 1 - coffee with milk 5 minutes

Figure 6.3: The questions of the survey to capture the activity model of MakeCoffee.

anonymity. In our current experiments, we use the Google Forms[1] service, mainly for three reasons:

1. Easy circulation: surveys can be sent by e-mail to target users, who receive a link to the web-based survey. This makes survey circulation easy and clear.
2. Anonymity: the answers of users are completely anonymous. When a user submits his/her answers, only a tag like "user 1" appears. There is no way to link user answers to a concrete user.
3. Simple and centralised answer management: the Google Forms service generates a centralised table where all the answers are collected. The table is automatically updated whenever a new answer arrives. This web-based table system for answers makes data management much easier.

Summarising, the survey for activities has different questions in order to obtain activity and behaviour models according to their definitions (Definitions 13 and 14). Surveys are circulated among target users using Google Forms, which offers convenient tools to send them by e-mail and collect anonymous answers in a centralised manner.

6.1.2.2 **Synthetic dataset generator**

Although the hybrid methodology for evaluation could be used in principle with any simulator for human activity recognition, a custom simulation tool has been developed to evaluate the EAM learning system. Available simulators such as Persim do not cover all the conditions to evaluate the EAM learning system appropriately. It may be very useful in the future, but nowadays it does not have the tools to model sensor errors and different variations of activities.

Following the ideas of Okeyo et al. (Okeyo et al., 2012), instead of developing a simulator tool that provides visual interaction like Persim, a synthetic dataset generator has been developed. The tool presented by Okeyo et al. does a very good job simulating time relations between sensor activations and activities, so their ideas regarding time management have been borrowed. But the simulator tool developed for this dissertation has more capabilities, allowing researchers to introduce different sensor activation sequences for activities with occurrence probabilities, activity sequences which occur only with a given probability and different ways to model sensor errors.

The synthetic dataset generator tool has been implemented in Python 2.7[2]. The inputs to the synthetic dataset generator are a script called ADL script, where activity and behaviour models for a concrete user are represented, and the context

[1]http://www.google.com/google-d-s/createforms.html
[2]https://www.python.org/

knowledge file, where some sensor error models are provided. As sensor error models are linked to their type (pressure, tilt, contact and so on), it is a natural choice to place them where all the sensors of a concrete environment are represented, i.e. in the context knowledge file. This part of the context knowledge file has not been introduced in Section 3.3.1 because it was not important for the EAM learning system. However, for simulation purposes, sensor error models play a crucial role, and including them in the context knowledge file was the straightforward solution. The considered sensor error modalities are two: sensor positive noise (see Definition 9) and missing noise (see Definition 10).

Figure 6.4 shows a high-level design for the synthetic dataset generator. As can be seen in the figure, activity and behaviour models and sensor positive noise are represented in the ADL script. On the other hand, sensor missing noise models are obtained from the context knowledge file. Using probabilistic time management tools, the synthetic dataset generator creates a sensor activation dataset, where all sensor activations are properly labelled to use it as ground truth. Sensor activations which are part of an activity are labelled with the activity name. But sensor activations which appear due to sensor noise are labelled with the special label None.

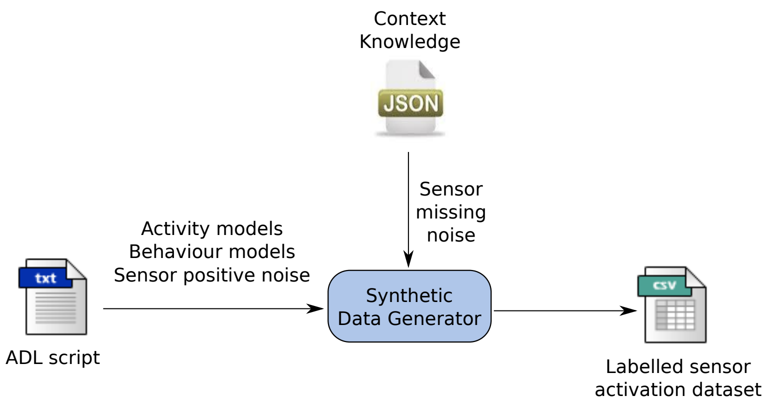


Figure 6.4: High-level design of the synthetic dataset generator tool.

One of the design decisions was to separate the representation of both sensor error models between two different files. The reason is that sensor missing noise is completely linked to sensor technology and the pervasive infrastructure (wireless receivers, communication, and system sampling and conversion mechanisms), whereas sensor positive noise is more related to environmental aspects, such as the distribution of objects and human behaviour. Hence while sensor missing noise can be considered a property of a sensor, sensor positive noise is more influenced by the inhabitant and the environment. That is why the missing error models are included in the context knowledge file depending on the sensor type and positive error models are represented in the ADL script which represents a concrete user.

To make this decision the research carried out by Chen et al. in (Chen et al., 2012b) has been considered. They show some experiments for activity recognition in a smart home environment using the dense sensing activity monitoring approach. Throughout the experiment of 144 activities, a total of 804 user-object interactions were performed. They used the so called User-object Interaction Recognition Accuracy (UoIR), defined as the system's correctly captured interactions against the total occurred interactions, as the metric to evaluate the reliability and performance of the activity monitoring mechanism. This metric takes into account not only unfired or misread interactions caused by faulty sensors but also those circumstances caused by the pervasive infrastructure. As such it is more accurate to reflect the system monitoring performance. Table 6.1 shows the UoIR for different types of sensors with an overall average UoIR of 96.89%. They conclude that these data prove the monitoring and acquisition mechanism of the system as being very reliable.

Sensor type	Total interactions	Captured interactions	Accuracy (%)
Contact	624	611	97.92
Tilt	126	119	94.44
Pressure	36	32	88.89
Sound	18	17	94.44

Table 6.1: Interaction recognition rate as shown in (Chen et al., 2012b).

However, no positive sensor noise has been identified in (Chen et al., 2012b), even though they simulate it in some of their experiments. This is quite reasonable, since the normal state of the sensor represents no interaction. It is very complicated from the technological point of view to change the state of a sensor when no interaction occurs, so it can be concluded that spontaneous sensor activations are very rare. If positive sensor noise is registered, it has to be mainly caused by undesired interactions that actually occur, even though they are not part of the activity. Those undesired interactions can be due to human erratic behaviour (see Definition 8) or interactions among objects caused by their distribution and casual movements.

Given that according to literature spontaneous sensor activations are rare and interactions among objects usually occur due to human intervention, it can be concluded that sensor positive noise is mainly caused by user erratic behaviour. Thus it has been included in the ADL script rather than in the contextual knowledge file.

ADL script

The ADL script defines activity models, behaviour models and sensor positive noise for a concrete user. It is currently implemented as a plain text file which has its own syntax. A parser function has been implemented to parse the file and obtain all the models defined on it.

The first information given in the ADL script refers to the number of days that has to be simulated. A natural number is provided there.

The next part of the file is for defining *sensor activation patterns* for activities. Sensor activation patterns are used to describe how activities are performed in terms of sensor activations and thus represent activity models in terms of sensors. An activity can have an arbitrary number of sensor activation patterns, which are specified with an occurrence probability and a sequence of sensor activations with relative time lapses. An example of sensor activation patterns for activity MakeCoffee can be found in Figure 6.5.

```
MakeCoffee 2
0.7 coffeePotSens@0 ktapSens@10 afcoffeeSens@30 cookerSens@20
    cupSens@180 fridgeSens@10 smilkSens@5 wsugarSens@10
0.3 coffeePotSens@0 ktapSens@10 afcoffeeSens@30 cookerSens@20
    cupSens@180 wsugarSens@10
```

Figure 6.5: Sensor activation patterns for MakeCoffee activity obtained from a real user. The activity has two activation patterns with different occurrence probabilities.

First of all, the name of the activity is defined. The number that comes after the name specifies the number of sensor activation patterns for that activity. The next line represents the first sensor activation pattern, which begins with an occurrence probability $p \in [0, 1]$. Notice that the occurrence probabilities of all sensor activations for a given activity must sum to 1. The probability number is followed by a sequence of sensor activations and relative time lapses. The first sensor activation's time has to be 0, indicating that it is the first sensor activation of the activity. The values that come after the '@' symbol represent the time in seconds between the previous sensor activation and the current one. In consequence, in the example given in Figure 6.5, *cookerSens@20* means that sensor activation *cookerSens* occurs 20 seconds after the *afcoffeeSens* sensor activation. The specific way the synthetic dataset generator treats those time lapses will be explained later, when the simulation loop is described.

Once all activity models are represented using appropriate sensor activation patterns, behaviour models are defined, which represent different days of the user in terms of performed activities (see Definition 14). Two kinds of behaviour models are defined:

1. *Sequences:* where a time slot is given with a sequence of activities and relative time lapses between two consecutive activities. Sequences are used to define those activity sequences that are always performed by a user in concrete days.

2. *Alterations:* where a probability value is assigned to an activity to be performed in a concrete time slot. Alterations represent a different kind of behaviour model. Some users might perform an activity regardless of the week day. For example, a user might watch television in evenings with a certain probability. Some days the user watches television, but some days does not. It does not depend on the day, but on some other causes (mood, last hour plans and so forth).

Specific week days are not represented in behaviour models. Instead of that, the probability of a concrete list of sequences and alterations is given. A list of sequences and alterations models a day. So if such a day model occurs 2 days in a week, i.e. in weekends, the assigned probability will be $2/7 \simeq 0.29$. An example is depicted in Figure 6.6. A typical day of a user is described, with an occurrence probability of 0.29, since the activity pattern describes a weekend day. In this case, the user reported that (s)he sometimes reads a book in the afternoon. Alterations allow modelling this kind of behaviour.

```
Prob 0.29 4
S 9:00-10:00 MakeCoffee@0 BrushTeeth@1800 ReadBook@120
S 13:30-14:30 MakePasta@0 BrushTeeth@600
S 22:00-23:00 BrushTeeth@0 WashHands@10
A 18:00-20:00 ReadBook 0.5
```

Figure 6.6: An example of a behaviour model for a concrete day, which has an occurrence probability of 0.29 and it is composed of three sequences and an alteration.

As it happens with sensor activation patterns, the occurrence probabilities of behaviour models must sum to 1.

The last part of the script is to define sensor positive noise (see Definition 9). As sensor positive noise is mainly caused by user erratic behaviour it is very complex to model it accurately. Besides, obtaining those models from user surveys is impossible, since users cannot tell how they interact with objects unpurposely. For those reasons, a simple sensor error model has been adopted which guarantees noise generation independently from ongoing activities. A probability value can be assigned to concrete sensors to get activated in an hour interval using a uniform probability distribution. For example, sensor *cupSens* can be assigned an activation probability of 0.1, which means that each hour, the sensor has a 0.1 probability of getting activated.

Context knowledge file

The context knowledge file has been described in Section 3.3.1, where an example of such a file was provided in Figure 3.2. But sensor missing noise models were not shown in that example. As sensor missing noise is mainly related to the sensor type, another part is added to the context knowledge file. More concretely, Figure 6.7 shows an example of how such error models are implemented for contact and tilt sensors. For each sensor type, a missing probability $p \in [0, 1]$ is provided.

```
"error_models": {
    "contact": {
          "prob": 0.0208
    },
    "tilt": {
          "prob": 0.0556
   },
   ...
}
```

Figure 6.7: Example of sensor missing noise models for contact and tilt sensors.

In consequence, the context knowledge file has the information about activities, objects, sensors and sensor missing noise models. It is mandatory for the synthetic dataset generator to keep the coherence between the sensors used in the ADL script and in the context knowledge file. The tool itself makes sure that coherence exists. If there is a sensor activation in the ADL script which is not represented in the context knowledge file, the synthetic dataset generator raises an error.

Simulation loop

Using the ADL script and the context knowledge file, the synthetic dataset generator creates a CSV file where each sensor activation has an associated timestamp and is labelled with an activity name or with the special label None if it is caused by noise. Additionally, activity start and end times are marked in the dataset.

For the purpose of generating realistic sensor activation datasets, the synthetic dataset generator has a simulation loop which has been represented in a flowchart in Figure 6.8. First of all, the simulator fixes the first day to start the simulation. In the current implementation the same day the simulator is launched is used as the first day. Afterwards, for the selected day, sensor positive noise is generated. For that purpose, the simulator generates a random number per each sensor with a probability greater than zero. If the sensor has to be activated, the simulator chooses a concrete time inside the current hour using a uniform distribution. The process finishes when the 24 hours of the day have been treated.

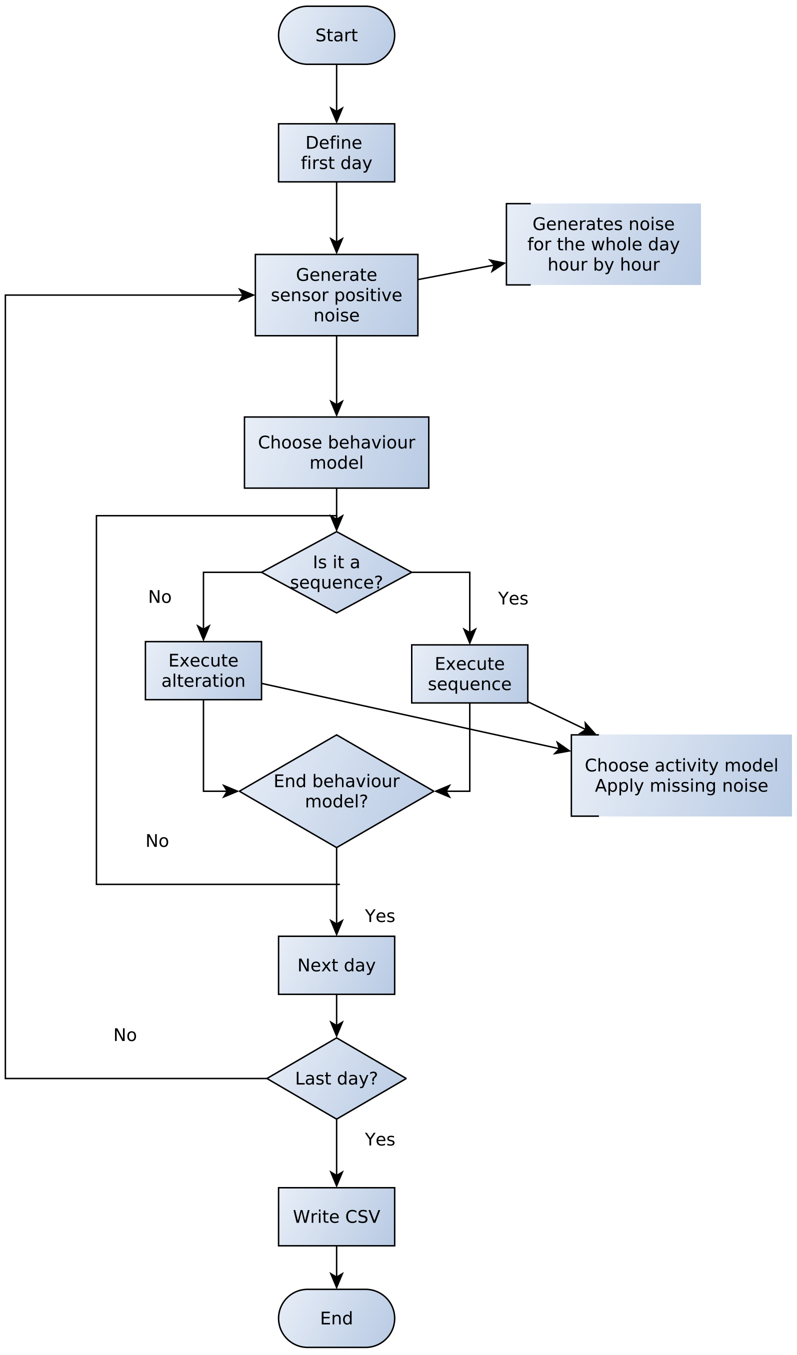


Figure 6.8: A flowchart of the simulation loop of the synthetic dataset generator.

Once sensor positive noise has been generated for the whole day, a behaviour model is chosen, taking into account the probabilities of each model. The behaviour model is a list of sequences and alterations. The first element of the list is taken. If it is a sequence, the activities of the sequence are executed in the specified time slot.

Let us show an example of how a sequence is executed by the simulator. Assume the sequence to be executed is such that

```
S 9:00-10:00 MakeCoffe@0 WatchTelevision@30 BrushTeeth@1800
```

In that case, the simulator generates a start time for the sequence in the provided time slot, using a uniform distribution. Afterwards, it picks the first activity (MakeCoffee) and looks for the sensor activation patterns of that activity. The simulator probabilistically chooses one of the sensor activation patterns of the activity and executes it. While executing the activity itself, two main aspects are taken into account:

1. Time lapses between sensor activations: the time lapses provided in the script are used as the mean values of Gaussian distributions, whose standard deviation is fixed to a 25% of the mean by default. So the time lapse is generated probabilistically using as reference the value given in the script. This makes varying and realistic time lapses for consecutive sensor activations.

2. Sensor missing noise: before generating the sensor activation, its missing probability is consulted in the context file. The simulator uses the missing probability to decide whether to generate the activation.

Similarly to sensor activations, time lapses between activities are treated through Gaussian distributions. At the end of the process, the whole sequence will be executed, with probabilistically chosen time lapses and sensor missing errors.

To execute an alteration, the simulator uses its occurrence probability. If it has to be executed, the activity is performed as in sequences. When the behaviour model is fully executed, i.e. all the elements of the list have been treated, the simulator generates the next day and repeats the whole process until the last day is reached. When this happens, the generated dataset is written in a CSV file, where properly labelled timestamped sensor activations can be found.

6.1.3 **Discussion about the hybrid methodology**

The hybrid methodology explained in Section 6.1.2 is based on the limitations and solutions identified in the literature (Helal et al., 2011), (Helal et al., 2012), (Okeyo et al., 2012). It has been specially designed to evaluate the EAM learning system and thus, it has some specific tools to adapt the methodology to the constraints of the system. However, the methodology itself, apart from concrete implementations, can be used to evaluate other activity recognition systems and has several advantages over the standard methodology explained in the beginning of Section 6.1. Let us enumerate and justify the advantages:

1. The hybrid methodology is cost effective, both from economic and time perspectives: from the economic point of view, the hybrid methodology does not need any pervasive environment, which can become an important investment. Furthermore, the cost of the experiments is also avoided, such as paying to participants, preparations and so on. From the time point of view, savings can be very important too, specially if a high number of experiments is planned. Preparing, distributing and processing surveys among users needs a variable amount of time, but it is unarguably less than preparing and running real experiments with humans.

2. A lot of users' information can be used: as is based on surveys, it is generally easy to achieve a great number of users for the tests, in contrast with what happens using the standard methodology. When running experiments with humans, finding volunteers is usually very difficult, specially if the experiment is long in time.

3. Ethical and legal issues are much softer: as there are no experiments with human beings many ethical and legal aspects can be avoided. The only important point to be considered is the anonymity of users regarding the information they provide in the surveys.

4. Datasets can be generated on demand: using the synthetic dataset generator or any other convenient simulator, arbitrary number of datasets can be generated as needed.

5. Perfectly labelled datasets can be obtained: the synthetic dataset generator labels all sensor activations according to the given script and sensor error models. In consequence, the generated dataset is a perfect ground truth. On the other hand, when using the standard methodology, dataset annotation is a real issue. As noted by Rashidi and Cook in (Rashidi and Cook, 2011): '*An aspect of activity recognition that has been greatly under-explored is the method used to annotate sample data that the scientist can use to train the activity model.*' Several annotation methods have been used by researchers, but all of them have their weaknesses. As a consequence, those datasets are not perfectly labelled and they can include spurious sensor activations in activities as ground truth.

6. Any kind of scenarios can be implemented: the synthetic dataset generator allows preparing experiments where no sensor noise exists, where only a concrete kind of sensor noise exists or where conditions are as close as possible to realistic settings. The chance of implementing all those varieties of scenarios allows researchers to test deeper their activity recognition systems, since they can see the influence of any factor they consider relevant.

The core ideas of the hybrid methodology are using surveys to capture user behaviour and simulators to generate synthetic data. The concrete implementations for the EAM learning system are derived from the constraints and features of the system, such as focusing on dense sensing scenarios, single user - single activity scenarios and so forth. However, when constraints are different, different tools can be used inside the same evaluation methodology.

In this case, the constraints and features of the EAM learning system make feasible the usage of surveys and simulators. The dense sensing paradigm for activity monitoring establishes that user-object interactions are monitored through simple sensor activations, which are used for activity modelling. So sensor activations are boolean valued, making simulation much easier. For example, Persim introduces sensor value generation functions, because the simulated sensors are broader and can produce different values depending on the context.

Another simplification comes from the activity modelling approach, which is based on action sequences mapped from sensor activations which monitor objects. This means that sensors like thermometers, humidity sensors and so on do not have to be modelled and simulated for the EAM learning system. Only sensors attached to objects. The same activity modelling approach makes feasible obtaining accurate activity models from users, since it is easy for any user to describe what objects are used to perform activities.

The single user - single activity constraint also simplifies the whole evaluation process. Firstly, users do not have to provide information about overlapping and concurrent activities, which is very complicated. Additionally, activity models and behaviour models are kept simple for simulation. Modelling and simulating concurrent activities is a challenge that is avoided in this case.

Finally, time lapses between actions and activities can also be simulated appropriately, as shown by Okeyo et al. (Okeyo et al., 2012). Users report approximate time lapses in the surveys and the synthetic dataset generator uses Gaussian random number generators in order to generate user based varying time lapses.

However, there are some disadvantages too and they are mainly related to human behaviour modelling. The first handicap is that modelling user erratic behaviour is complex. It is very difficult for any user to specify what objects (s)he interacts with unpurposely or to say how an activity is aborted because another thing has to be done. The synthetic dataset generator offers a very straightforward way to model this kind of interactions, but it is the researcher who has to assign probability values to specific sensors without having an appropriate way to model user erratic behaviour. The strategy followed in this dissertation is to introduce high levels of random positive noise to compensate the lack of appropriate models. As it will be shown in Section 6.2, the difference in the number of sensor activations between noiseless and noisy scenarios are greater than 35% in average, which means that the introduced sensor positive noise is more than a third of the original dataset.

Another disadvantage refers to the information provided in surveys regarding activity models. Some users are very precise in their answers, but some are not. Activity descriptions given by some users might be discarded due to their vagueness. But this is not generally a big problem, since obtaining more answers is usually easy. Some other users may omit important details of activities in their answers, forgetting for example an object which has been used, hence the precise way of performing activities cannot always be captured. This is certainly a problem while modelling activities, but from the EAM learning evaluation perspective, it is a minor problem. The aim of the EAM learning system is to learn activity models which are represented in the datasets. Whether those datasets are really an accurate version of an activity is not that important as far as the learning system captures and models them as they appear.

In consequence, it can be claimed that the hybrid evaluation methodology with its current implementation, is appropriate to evaluate the proposed EAM learning system. Its advantages are many and very important, while the disadvantages can be coped with specific strategies.

6.2 Evaluation Scenarios, Results and Discussions

Using the tools and methodologies described in Section 6.1, several evaluation scenarios have been prepared to test and validate the most important aspects of the EAM learning system. More concretely, three major evaluation scenarios can be distinguished:

1. SA^3 evaluation scenarios: SA^3 is a very important part of the whole EAM learning system, since it is the initialisation step of the clustering algorithm. The performance of SA^3 has to be carefully analysed to understand how it works, its main advantages and strong points as well as its main weaknesses. Remember that the AA algorithm works on the results of SA^3, adding -or not- actions to the initial action clusters detected by SA^3. But if SA^3 fails at detecting an activity, AA cannot recover it. Thus, the performance of SA^3 in all possible situations is key for the performance of the clustering process and in consequence, the performance of the EAM learning system.

2. Activity clustering evaluation scenarios: the activity clustering process is the sum of the SA^3 and AA algorithms. As a result of the process itself, every sensor activation of a dataset is labelled and several clusters for the defined activities are obtained. Action clusters' quality is directly linked to labelling performance. Remember that those action clusters are finally processed by the AML algorithm to learn extended activity models, so assessing the performance of the clustering process by means of labelled sensor activations is very important.

3. EAM learning evaluation scenarios: finally, the EAM learning system has to be evaluated. The system is composed by the sequential use of SA^3, AA and AML. Evaluation scenarios have to be set-up in order to validate that the EAM learning system is able to learn extended activity models for different users. The results obtained in those evaluation scenarios are the most important ones, because they give a clear vision of the performance of the whole system.

6.2.1 SA^3 performance

6.2.1.1 Evaluation scenarios and metrics

To assess the performance of SA^3 exhaustively, the features of the synthetic dataset generator will be used to set up different kinds of scenarios. To prepare those scenarios, surveys will not be used. In the case of SA^3, the main interest is to measure the performance in presence of severe constraints, e.g. increasing noise, activities with several variations, varied order of actions, using IAMs that share the same actions, activity sequences that are very close in time and so on. As SA^3 alone is not going to be used for real applications, the usage of surveys is not necessary. SA^3 makes sense inside the complete activity clustering algorithm, whose performance will be tested using real users' inputs.

With this purpose, four scenarios have been prepared:

1. The ideal scenario: SA^3 will be tested in ideal conditions, i.e. there is no sensor noise in the dataset.

2. Missing sensor noise scenario: increasing missing error probability is introduced to two sensors that are mapped to actions of the IAMs of two activities, to understand how the detection of those activities evolves compared to noiseless activities. It is also important to see whether the missing noise affecting to some activities influence the detection of the others.

3. Positive sensor noise scenario: increasing positive noise is introduced to assess the performance of SA^3 in presence of positive noise.

4. Demanding activities scenario: the fourth scenario tests how SA^3 performs when IAMs of several activities share many actions among them.

The base of the first three scenarios is an ADL script prepared to offer some common features to all experiments. From the activity models point of view: (i) several variations for every activity, (ii) different probability distributions between activity variations, (iii) varied order of actions in activities and (iv) varied time lapses between consecutive sensor activations. From the behaviour model perspective: (i) different day types, (ii) activities which are very close in time and (iii) combinations of sequences and alterations.

```
MakeCoffee 2
0.5 mugSens@0 smilkSens@20 microwaveSens@20 afcoffeeSens@120
    wsugarSens@20
0.5 cupSens@0 ktapSens@10 microwaveSens@15 wsugarSens@90
    afcoffeeSens@30
MakeChocolate 2
0.8 mugSens@0 wmilkSens@20 microwaveSens@20 chocoSens@120
0.2 cookerSens@0 potSens@5 wmilkSens@20 chocoSens@30
    mugSens@200
MakePasta 2
0.8 potSens@0 ktapSens@20 cookerSens@30 macaroniSens@120
    ftomatoSens@600
0.2 spaghettiSens@0 potSens@20 ktapSens@25 cookerSens@30
    baconSens@50 creamSens@600
BrushTeeth 2
0.7 brusherSens@0 toothpasteSens@5 glassSens@30 btapSens@5
0.3 brusherSens@0 toothpasteSens@5 glassSens@30 btapSens@5
    dentalflossSens@15
WatchTelevision 2
0.5 sofaSens@0 rcontrolSens@5 tvSens@10
0.5 rcontrolSens@0 tvSens@5 sofaSens@5
WashHands 2
0.85 btapSens@0 bsoapSens@15 handcreamSens@40
0.15 btapSens@0 bsoapSens@15
ReadBook 2
0.9 bookbSens@0 bedSens@10 blampSens@5
0.1 bookaSens@0 sofaSens@10
```

Figure 6.9: Sensor activation patterns for the defined activities as given to the synthetic dataset generator.

The prepared ADL script reflects all those features and tries to be realistic in the definition of the activity and behaviour models. However, notice that at this point, realism is not important (the following evaluation scenarios will address realism properly). Figure 6.9 shows the defined seven activities of daily living with all the sensor activation patterns and occurrence probabilities. As can be seen in Figure 6.9, the defined activities are: MakeCoffee, MakeChocolate, MakePasta, BrushTeeth, WatchTelevision, WashHands and ReadBook. Different probability distributions can be found for different activities. For example, MakeCoffee has an even occurrence probability for its two variations (50% - 50%). However, ReadBook is very unbalanced (90% - 10%). Varied order of actions and time lapses can also be seen in the sensor activation patterns of each activity. Finally, some activities contain a lot of sensor activations - six in the case of MakePasta - whereas some others are very short - WashHands or ReadBook with only two -. In general, activity models reflect all the desired features to test the performance of SA^3, as identified in the paragraph above.

Behaviour models defined for the first three evaluation scenarios are depicted in Figure 6.10. Three typical days are defined with different occurrence probabilities. The first one has three sequences and one alteration. Activity WashHands is very close in time to activity BrushTeeth, for example. The second day type tries to simulate a typical day where the inhabitant spends a short time in home, showing a big time difference between both activity sequences. Finally, the third day type contains two alterations and three new activity sequences where some activities are again very close in time.

```
Prob 0.43 4
S 7:00−7:30 MakeChocolate@0 BrushTeeth@120 WashHands@30
S 13:00−13:30 MakePasta@0 MakeCoffee@60 BrushTeeth@1800
S 20:00−20:30 MakePasta@0 BrushTeeth@200 ReadBook@150
A 18:00−19:30 WatchTelevision 0.8
Prob 0.28 2
S 7:00−7:30 MakeChocolate@0 BrushTeeth@100 WashHands@30
S 20:30−21:00 BrushTeeth@0 ReadBook@50
Prob 0.29 5
S 9:00−10:00 MakeChocolate@0 WatchTelevision@30 BrushTeeth@1800
S 13:30−14:30 MakePasta@0 BrushTeeth@150
S 22:00−23:00 BrushTeeth@0 WashHands@10
A 15:00−16:00 WatchTelevision 0.75
A 18:00−20:00 ReadBook 0.5
```

Figure 6.10: Behaviour models for a given user as given to the synthetic dataset generator.

Another common element of the first three evaluation scenarios is the IAMs of the activities. All seven activities have the same IAMs alongside the three scenarios. Figure 6.11 shows the IAMs used for the experiments. It is very important to keep the same IAMs in all the experiments, since IAMs aim at describing incomplete but generic activity models for any user.

The fourth evaluation scenario is different from the other three scenarios. The idea is to use some activities whose IAMs share the same actions. For that purpose, and in order to keep the realism, the following seven activities have been defined: MakeCoffee, MakeWhippedCream, MakeTiramisu, BrushTeeth, WatchTelevision, WashHands and ReadBook. The first three activities' IAMs are formed by the same four actions. More concretely:

```
IAM(MakeCoffee) = {hasContainer, hasCoffee, hasFlavour}
IAM(MakeWhippedCream) = {hasContainer, hasCream, hasFlavour}
IAM(MakeTiramisu) = {hasContainer, hasCream, hasCoffee}
```

The other activities are defined as in the previous scenarios. So the aim of this evaluation scenario is to see how SA^3 behaves with challenging IAMs. To focus only on the pattern recognition performance, no noise is added to this fourth evaluation scenario.

For all four evaluation scenarios the same metrics have been used. The most important aspect for SA^3 to be evaluated is whether activities are well detected. So labels given to sensor activations are not important. Rather, a comparison between the start and end times of the activity detected by SA^3 and the real activity has to be done. Assume A_R is the real activity and A_{SA^3} the activity detected by SA^3. A_{SA^3} is a correct activity detection if A_R contains A_{SA^3} and both activities have the same label:

$$\begin{gathered} t_{start}(A_R) \leq t_{start}(A_{SA^3}) \text{ and } t_{end}(A_{SA^3}) \leq t_{end}(A_R) \\ Label(A_R) = Label(A_{SA^3}) \end{gathered} \tag{6.1}$$

This evaluation criterion is called *activity-based evaluation*. Using activity-based evaluation criterion, true positives, false positives and false negatives are calculated for every activity, comparing the output of SA^3 with the ground truth. Notice that true negatives are ignored, since they are equivalent to computing the true positives for the None activity, which is not interesting for evaluation purposes.

6.2.1.2 **Results**

Ideal scenario

To test the performance of SA^3 and have a base reference, the ideal scenario has been designed. The ADL script has already been described (Section 6.2.1.1) and it contains varying order of sensor activations per activity, varied occurrence probability distributions and varied time lapses between sensor activations and activities.

```
"MakeCoffee": {
    "type": ["Cooking"],
    "location": ["Kitchen"],
    "IAM": ["hasContainer", "hasCoffee"],
    "duration": 360
},
"MakeChocolate": {
    "type": ["Cooking"],
    "location": ["Kitchen"],
    "IAM": ["hasContainer", "hasChocolate"],
    "duration": 500
},
"MakePasta": {
    "type": ["Cooking"],
    "location": ["Kitchen"],
    "IAM": ["hasPasta", "useCookingAppliance",
            "useCookingUtensil"],
    "duration": 1200
},
"BrushTeeth": {
    "type": ["Hygiene"],
    "location": ["Bathroom"],
    "IAM": ["hasBrusher", "hasToothpaste", "turnOnTap"],
    "duration": 130
},
"WatchTelevision": {
    "type": ["Entertainment"],
    "location": ["Lounge"],
    "IAM": ["hasRemoteControl", "useTV"],
    "duration": 40
},
"WashHands": {
    "type": ["Hygiene"],
    "location": ["Bathroom"],
    "IAM": ["turnOnTap", "hasSoap"],
    "duration": 90
},
"ReadBook": {
    "type": ["Entertainment"],
    "location": ["Bedroom", "Lounge"],
    "IAM": ["hasBook", "useFurniture"],
    "duration": 30
}
```

Figure 6.11: The IAMs of all seven defined activities for the first three evaluation scenarios.

The aim of the ideal scenario is to obtain a base reference of the performance of SA^3 in noiseless conditions. Activity-based evaluation is used and true positives, false positives and false negatives are measured.

With the objective of getting reliable results, 130 days have been simulated. More concretely, the sensor activation dataset contains 4093 sensor activations describing a total number of 1027 activity instances. The results are shown in Table 6.2. For each activity, the number of instances and the percentages of true positives, false positives and false negatives are given.

Activity	Instances	True Positives (%)	False Positives (%)	False Negatives (%)
MakeChocolate	130	100	0	0
WatchTelevision	103	100	0	0
BrushTeeth	350	100	0	0
WashHands	130	100	0	0
MakePasta	146	100	0	0
ReadBook	112	100	0	0
MakeCoffee	56	100	0	0

Table 6.2: Results of the SA^3 algorithm for the ideal scenario.

Table 6.2 shows that the 100% of all activities are properly captured by SA^3 under ideal conditions. There are neither false positives nor false negatives. Remember that the activity-based evaluation does not check for the labels of every sensor activation. The fact that a 100% of all activities are properly captured indicates that SA^3 gets activity time locations robustly.

Sensor missing noise scenario

For the next experiments, assessing the performance of SA^3 under sensor missing error noise is the objective. In principle, the completion criterion of the activity sequence finding step of SA^3 suggests that if a sensor activation which is mapped to one of the IAMs fails, SA^3 will fail to detect that activity (see Section 4.1.2 for a detailed explanation). In order to test that suggestion, two of the seven activities have been chosen. Sensor activations which are mapped to the IAMs of those activities are given an increasing missing probability, to see how this missing noise affects the detection of those activities. The rest of the activities are kept noiseless, to measure the effect of missing noise in other activities. Table 6.3 shows the results for the sensor missing noise scenario. The activities which suffer missing noise are marked with an asterisk (MakeChocolate and BrushTeeth). Five experiments are run, each of them simulating 130 days. Sensor missing probabilities are 0.01, 0.02, 0.05, 0.07 and 0.1. For each experiment, true positives, false positives and false negatives are measured, using the activity-based evaluation criterion.

As can be seen in Table 6.3, the performance of SA^3 for the rest of activities does not vary. It keeps on producing 100% of true positives, while false positives

Activity	Sensor missing probability														
	0.01			0.02			0.05			0.07			0.1		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
MakeChocolate*	99.2	0	0.8	99.2	0	0.8	94.6	0	5.4	92.3	0	7.7	89.2	0	10.8
WatchTelevision	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
BrushTeeth*	98.9	0	1.1	98.6	0	1.4	96	0	4	93.8	0	6.2	89.4	0	10.6
WashHands	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
MakePasta	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
ReadBook	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
MakeCoffee	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0

Table 6.3: Results of SA^3 for the sensor missing noise scenario, where TP: true positives, FP: false positives and FN: false negatives. The activities affected by the noise have an asterisk.

and negatives are null. However, true positive rate of activities MakeChocolate and BrushTeeth seems to be linearly reducing with sensor missing noise probability. The reduction of true positives generates the appearance of false negatives, since those activities that are not detected, are labelled as None. Thus, they are considered false negatives. There is no effect in false positives, but the effect on true positives is clear, as expected. To make the linearity more clear, Figure 6.12 plots true positive rates for both activities for measured sensor missing noise probabilities.

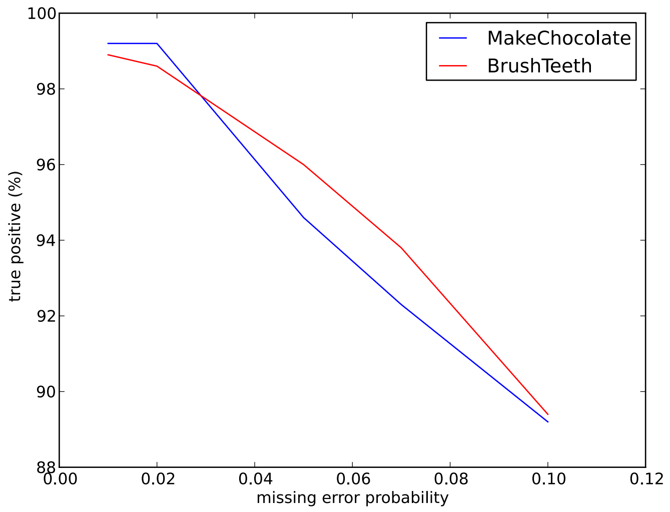


Figure 6.12: The true positive percentages of SA^3 for activities MakeChocolate and BrushTeeth in presence of increasing sensor missing noise.

Taking into account that noise generation is probabilistically accomplished in the synthetic dataset generator, Figure 6.12 is a confirmation of the inverse linear relation between sensor missing noise and SA^3 activity detection performance. When missing error increases, activity detection decreases.

Sensor positive noise scenario

To test the sensor positive noise scenario, five experiments have been run. For each of the experiments, six sensors are picked up randomly and they are assigned increasing sensor positive noise probabilities. This strategy allows measuring the performance of SA^3 in presence of positive noise. For each of the experiments 130 days are simulated again. To have a reference of the level of noise, sensor activations in each dataset scale from 5054 to 8850. The noiseless scenario presented 4093 sensor activations, so the noisiest dataset contains around 46% more sensor activations. Table 6.4 shows the results for the sensor positive noise scenario. Each experiment is identified by its noise level, where 0.05 means that all randomly selected sensors have a positive noise probability of 0.05 per hour. For each experiment, true positives, false positives and false negatives are depicted.

Activity	**Sensor positive probability**														
	0.05			0.1			0.15			0.2			0.25		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
MakeChocolate	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
WatchTelevision	100	0	0	100	0	0	99.1	0.9	0	100	0	0	100	0	0
BrushTeeth	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
WashHands	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0
MakePasta	100	0	0	100	0	0	100	0	0	100	0	0	99.3	0.7	0
ReadBook	100	0	0	100	0	0	97.5	2.5	0	100	0	0	98.1	1.9	0
MakeCoffee	100	0	0	100	0	0	100	0	0	100	0	0	100	0	0

Table 6.4: Results of SA^3 for positive sensor noise scenario; TP: true positives; FP: false positives; FN: false negatives

Table 6.4 shows that sensor positive noise affects true positives and false positives. For the first one, a slight reduction for some activities can be seen. For the second one, some activities show rates that even being greater than zero, are still very low, 2.5% being the highest for those experiments. Exploring the obtained data deeper, it was found that in some cases, the false positives are generated overlapped with the real activity, i.e. a false sensor activation appears close to the real activity and changes the real time assignments for that activity. This happens when a sensor activation which is mapped to an action in the IAM of a given activity is generated by chance very close in time from a real execution of that activity. The effect is that the SA^3 fails at annotating all real occurrences of a given activity and produces false positive annotations. This can be seen in activities WatchTelevision and MakePasta in Table 6.4, for example. To understand the problem better, Figure 6.13 shows a representation of the situation. Assuming that dashes represent time lapses without sensor activations, circles represent real sensor activations and crosses those sensor activations generated due to positive noise, the problem can be seen clearly. A false sensor activation which is mapped to the IAM of the real activity A_R is generated by chance just before the beginning of the activity. SA^3 detects it as part of the activity and thus, the start time of the activity is not detected correctly. The consequence is that a real activity is not detected and a false positive activity is generated.

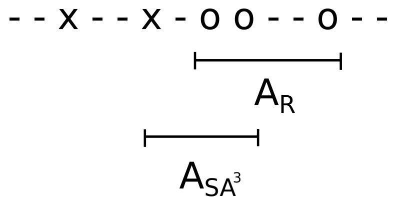


Figure 6.13: A representative situation encountered in positive sensor noise scenarios. Dashes represent time intervals without sensor activations, circles represent real sensor activations and crosses represent false sensor activations. While A_R stands for real activity, A_{SA^3} stands for the activity detected by SA^3.

In some other cases, false positives do not affect the annotation of correct activities (look at ReadBook for 0.15 and 0.25 noise levels, where all instances of the activity were perfectly annotated). In those cases, the problem is that the IAM of activity ReadBook is composed by two actions: *hasBook* and *useFurniture*. As sensor activations for two sensors monitoring furniture and books were chosen randomly for noise generation, it turned out that both activations were given in such circumstances that they were detected as activities. However, they do not affect any other activity detection. This situation is represented in Figure 6.14. The real activity is correctly detected by SA^3, but another false activity is also detected due to the false generation of two sensor activations that are mapped to actions in an IAM.

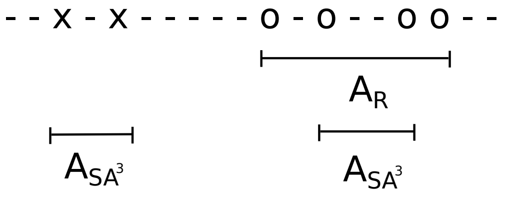


Figure 6.14: Another representative situation encountered in positive sensor noise scenarios. Dashes represent time intervals without sensor activations, circles represent real sensor activations and crosses represent false sensor activations. While A_R stands for real activity, A_{SA^3} stands for the activity detected by SA^3.

Demanding activities scenario

The final scenario aims at testing SA^3 under demanding activities. As it has been explained in Section 6.2.1.1, demanding activities are those which share many common actions in their IAMs. It can be thought that as SA^3 uses a pattern recognition algorithm where some specific domain-based criteria are applied, its performance could be affected by very similar IAMs. Those conditions are simulated using seven activities, where MakeCoffee, MakeTiramisu and MakeWhippedCream share the same four actions in their IAMs. Remember that even being very similar, they are still different and unique. For this special scenario, a total number of 1262 activity instances have been produced. No sensor noise has been produced for the experiment, since the objective is to see the effect of demanding activities. The obtained results can be seen in Table 6.5.

Activity	Instances	True positives (%)	False positives (%)	False negatives (%)
MakeCoffee	222	100	0	0
MakeTiramisu	120	100	0	0
MakeWhippedCream	72	100	0	0
WashHands	150	100	0	0
BrushTeeth	420	100	0	0
ReadBook	127	100	0	0
WatchTelevision	151	100	0	0

Table 6.5: Results of SA^3 for demanding activity models scenario.

Results show that SA^3 is not affected by the similarity of IAMs, as long as they are unique. True positives are 100% for all activities, whereas false positives and negatives are zero. The situation is exactly the same as in the ideal scenario. Notice that assuming unique IAMs for each activity is very reasonable, since if there are two equal IAMs for two activities, those two activities should actually be one.

6.2.1.3 Discussion about SA^3

The results depicted in Section 6.2.1.2 show that SA^3 can handle varying order of actions in activity performance without any problem. All four set-ups contain different activation patterns, where actions are executed in different orders. This can be seen in Figure 6.9: every activity has different sensor activation sequences with varying orders. As the ideal scenario shows (Table 6.2), the true positive rate of the algorithm for every activity is 100% and neither false positives nor negatives are observed. Notice that sensor activation sequences used for each activity contain many actions which are not used to define any of the IAMs, but the algorithm still performs perfectly when detecting activities. Hence, SA^3 can also handle non-considered actions.

As far as noisy scenarios regard, results suggest that missing sensor activations cannot be properly handled by the algorithm, when those activations are linked to actions that are used to define IAMs, i.e. if a sensor activation which is mapped to one of the actions used to define the IAM of a concrete activity fails, that activity cannot be detected by SA^3. The non-captured activities are proportional to the missing activation probability of those actions, as shown in Figure 6.12. This is due to the completeness criterion used in the second step of SA^3, as explained in Section 4.1, which demands the presence of all actions in the IAM of an activity to consider an action sequence a valid activity. However, missing sensor noise only affects true positive and false negative rates of the activities which are faultily registered by sensors. False positive rates keep behaving as the ideal scenario. Moreover, a non-detection of an activity does not have any other effect on the other activities. Following this reasoning, a good way to minimise the impact of sensor missing noise is to monitor key objects for IAMs using robust sensors such as contact sensors.

Independently from such robust monitoring strategies, it is reasonable to think whether algorithmic solutions to mitigate sensor missing noise can be adopted. The completion criterion is very restrictive in that sense, but it is also very reasonable, since if not, distinguishing real activity executions from those ones generated by positive noise would be very difficult, specially considering the low number of actions used in the IAMs (see Table 6.22). Further approaches have not been analysed in this dissertation, because the low level of missing sensor noise in real environments makes the detection rates very good. However, some ideas are suggested in Chapter 7 for their future development.

For sensor positive noise, the scenario is different. SA^3 has shown to perform reliably even with high levels of noise. For example, even having certain sensors with faulty activation probabilities of 0.25 per hour, average true positive rate over all activities is 99.9%. Simultaneously, false positive rates are still very low, concretely 0.3% in average, while false negative rates are 0. False positive activity detections are due to random occurrences of actions in time instants which make them compatible with the duration criterion of an activity. The probability of having such false positives increases as the noise level increases and the number of actions in IAMs decreases. However, the results obtained with the highest level of noise of the experiments show how robust SA^3 is in such scenarios.

Finally, a test involving challenging activity models has shown that as far as activity models are unique, SA^3 does not have any problem detecting them. Average true positive rate for all activities is 100% and false positive and negative rates are 0, as can be seen in Table 6.5. Remember that the experiments for challenging or demanding activity models were performed without any noise. So the obtained results suggest that the performance is exactly the same as for the ideal scenario.

In conclusion, SA^3 has been tested for all the desired requirements listed in Section 6.2.1.1. From the activity models point of view: (i) several variations for every activity, (ii) different probability distributions between activity variations, (iii) varied order of actions in activities and (iv) varied time lapses between consecutive sensor activations. From the behaviour model perspective: (i) different day types, (ii) activities which are very close in time and (iii) combinations of sequences and alterations. All those features were present in all four evaluation scenarios, where ideal conditions, noisy scenarios and demanding activity models were also tested. The obtained results suggest that SA^3 is a very reliable tool to detect activity executions. It does not correctly label all the sensor activations, but it accurately captures activity time locations. Thus, it can be concluded that SA^3 serves as a initialisation step for an activity clustering process.

6.2.2 Activity clustering performance

6.2.2.1 Evaluation scenarios and metrics

Testing the activity clustering algorithm, i.e. $SA^3 + AA$, requires the usage of the hybrid evaluation methodology. The activity clustering algorithm can be seen as an activity annotator algorithm which labels each sensor activation with an activity label. Notice that activity annotation and recognition are not the same thing. Activity recognition is performed online, while sensor activations are being generated. But activity annotation happens offline, once all sensor activations have already been produced. So annotating is typically easier than recognising, since it does not have time restrictions.

Contrary to the evaluation of SA^3, realism is a key issue when testing activity clustering. That is why the hybrid evaluation methodology is used to set up two evaluation scenarios: the *ideal* and the *complete* scenario. The first one does not contain any sensor noise, which makes easier the annotation process. The complete scenario is closer to reality since it has sensor noise, which makes annotating more demanding. Sensor missing noise errors are obtained from the statistics given in (Chen et al., 2012b) and shown in Table 6.1. Noise error models have been specified depending on sensor type. For example, pressure sensors have a missing probability of around 10%. This means that whenever the activity script contains a pressure sensor activation, the synthetic dataset generator will perform it with a 90% of probability. On the other hand, to model sensor positive noise, the strategy introduced in Section 6.1.3 has been followed, i.e. to introduce high levels of random positive noise to compensate the lack of appropriate models for user erratic behaviour. For all experiments, 6 sensors are randomly chosen to assign them a positive noise probability of 0.05 per hour. And other five sensors are also chosen randomly to assign them a 0.1 probability.

ADL surveys have been circulated among people from DeustoTech - University of Deusto. The target users are researchers, students and professors, with ages ranging from 20 to 60 years old. The circulated ADL survey can be found in the web[1]. A total number of 12 answers have been received, but 4 of them were discarded because they provided too vague information to model properly the activity and behaviour models. Thus, 8 real users are finally used for the experiments.

It is worth to remember that the AA algorithm has three different time metrics (see Section 4.2). To test which of them is better, all the generated datasets are treated three times, using a different time metric for each experiment.

So the experimental set-up designed consists of:

1. 8 real users.
2. 7 activities of daily living labelled as MakeCoffee, MakeChocolate, MakePasta, BrushTeeth, WatchTelevision, WashHands and ReadBook.
3. 2 scenarios: the ideal scenario and the complete scenario.
4. 3 executions of the activity clustering process, using the three different time metrics.

The datasets used for the experiments carried out to evaluate activity clustering are available in the web[2]. As a summary, 16 different datasets can be found there, 2 per user: the datasets for the ideal and the complete scenarios. All datasets simulate 60 days according to the behaviour models provided by each user.

The evaluation criterion in this case, is the so called *action-based evaluation*: compare the labels given by the clustering process to every sensor activation/action with the ground truth produced by the synthetic dataset generator tool by means of true positives, false positives and false negatives (true negatives are ignored again, because they represent actions which have been correctly labelled as None). This evaluation criterion assesses the performance of the clustering process as an activity annotator.

6.2.2.2 **Results**

Ideal scenario

The 8 datasets generated from the ADL surveys for the ideal scenario contain 2128 sensor activations in average. For each of the experiments, the average true positives, false positives and false negatives are depicted per activity. The averages are calculated over the 8 users. A comparison between the actions labelled by SA[3] and AA is depicted in the results. Remember that those results cannot be compared to

[1]http://goo.gl/etCNyi

[2]http://www.morelab.deusto.es/pub/synthetic_adl_datasets/

the results shown for SA^3 evaluation (Section 6.2.1.2), since the evaluation criteria are different. Table 6.6 shows the results when simple time distance is used in AA (Equation 4.8). Similarly, Table 6.7 depicts the results when normalised time distance is applied (see Equation 4.9). Finally, Table 6.8 shows how the clustering process performs when the dynamic centre normalised time distance is used for the previous activity (see Equation 4.10) and the normalised time distance for the next activity.

Activity	**Clustering Results**					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	55.05	98.67	0	0	44.95	1.33
WatchTelevision	75.12	100	0	0	24.88	0
BrushTeeth	90.91	96.8	0	0	9.1	3.2
WashHands	74.88	92.93	0.07	6.94	25.05	0.13
MakePasta	55.74	99.73	0	0	44.26	0.27
ReadBook	89.08	100	0	0	10.91	0
MakeCoffee	62.63	99.06	0	0.1	37.37	0.84

Table 6.6: Average results for 8 users of the clustering process for the ideal scenario using simple time distance.

Activity	**Clustering Results**					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	55.05	98.67	0	0	44.95	1.33
WatchTelevision	75.12	100	0	0	24.88	0
BrushTeeth	90.91	96.57	0	0	9.1	3.43
WashHands	74.88	92.32	0.07	7.55	25.05	0.13
MakePasta	55.74	99.8	0	0	44.26	0.2
ReadBook	89.08	100	0	0	10.91	0
MakeCoffee	62.63	99.16	0	0	37.37	0.84

Table 6.7: Average results for 8 users of the clustering process for the ideal scenario using normalised time distance.

To get a clearer comparison between three time metrics, Table 6.9 shows the average precision and recall for each time metric. The average is calculated over all activities. As it can be seen, both precision and recall are very high for all three time metrics. However, the combination of dynamic centre normalised and normalised time distances yields the best results, specially for precision. The difference for recall is not very significant.

Complete scenario

The 8 datasets for the complete scenario are obtained from the same ADL surveys and scripts. But for these datasets, sensor error models are applied. As a result,

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	55.05	98.67	0	0	44.95	1.33
WatchTelevision	75.12	100	0	0	24.88	0
BrushTeeth	90.91	98.27	0	0	9.1	1.73
WashHands	74.88	96.69	0.07	3.03	25.05	0.27
MakePasta	55.74	99.78	0	0.02	44.26	0.2
ReadBook	89.08	100	0	0	10.91	0
MakeCoffee	62.63	99.09	0	0	37.37	0.91

Table 6.8: Average results for 8 users of the clustering process for the ideal scenario using dynamic normalised time distance for previous and normalised time distance for next activity.

	Avg. Precision	Avg. Recall
t1	98.98%	99.17%
t2	98.91%	99.14%
t3	99.56%	99.36%

Table 6.9: Comparative between the usage of the three time metrics for the clustering process in the ideal scenario. t1 refers to simple time distance, t2 to normalised time distance and t3 to using dynamic centre normalised time distance for previous activity and normalised time distance for next activity.

datasets have an average of 2942 sensor activations. Compared to the ideal scenario, those datasets have 38.25% more sensor activations in average, which gives a clear idea of the high level of sensor positive noise. Notice that this increment comes even having missing errors, which are applied to every sensor activation. The combination of missing and positive noise gives datasets containing 38.25% more sensor activations, thus sensor positive noise is very high.

Results are shown in the same manner as for the ideal scenario. As such, Table 6.10 shows the results when using simple time distance, Table 6.11 when using normalised time distance and finally, Table 6.12 when combining dynamic centre normalised and normalised time distances. Once again, a summary of the performance of the three variants is depicted in Table 6.13, where average precision and recall are shown.

6.2.2.3 **Discussion about the clustering process**

The first discussion point for the clustering process is the difference in performance of the three time metrics. Tables 6.9 and 6.13 show a clear comparative, both in the ideal and complete scenario. It can be seen that the difference between simple time distance and normalised time distance is not significant. But using dynamic centre normalised time distance for previous activity and normalised time

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	54.47	96.6	0.52	1.2	45.01	2.2
WatchTelevision	71.13	100	0	0	28.87	0
BrushTeeth	90.95	96.64	0.25	0.35	8.8	3.01
WashHands	75.74	91.96	0.51	6.66	24.75	1.38
MakePasta	53.48	97.07	0.73	2.64	45.79	0.29
ReadBook	82.56	94.37	0.39	0.29	17.05	5.33
MakeCoffee	58.53	97.91	0.9	1.91	40.57	0.17

Table 6.10: Average results for 8 users of the clustering process for the complete scenario using simple time distance.

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	54.47	96.6	0.52	1.18	45.01	2.22
WatchTelevision	71.13	100	0	0	28.87	0
BrushTeeth	90.95	96.37	0.25	0.35	8.8	3.28
WashHands	75.74	91.27	0.51	7.33	24.75	1.39
MakePasta	53.48	97.19	0.73	2.64	45.79	0.17
ReadBook	82.56	94.37	0.39	0.29	17.05	5.33
MakeCoffee	58.53	98.07	0.9	1.76	40.57	0.17

Table 6.11: Average results for 8 users of the clustering process for the complete scenario using normalised time distance.

Activity	Clustering Results					
	True Positive (%)		False Positive (%)		False Negative (%)	
	SA^3	AA	SA^3	AA	SA^3	AA
MakeChocolate	54.47	96.69	0.52	1.09	45.01	2.22
WatchTelevision	71.13	100	0	0	28.87	0
BrushTeeth	90.95	98.08	0.25	0.35	8.8	1.57
WashHands	75.74	95.6	0.51	2.81	24.75	1.59
MakePasta	53.48	96.58	0.73	2.83	45.79	0.59
ReadBook	82.56	94.37	0.39	0.29	17.05	5.33
MakeCoffee	58.53	97.65	0.9	1.59	40.57	0.75

Table 6.12: Average results for 8 users of the clustering process for the complete scenario using dynamic normalised time distance for previous and normalised time distance for next activity.

	Avg. Precision	Avg. Recall
t1	98.1%	98.19%
t2	98.03%	98.17%
t3	98.7%	98.25%

Table 6.13: Comparative between the usage of the three time metrics for the clustering process in the complete scenario. t1 refers to simple time distance, t2 to normalised time distance and t3 to using dynamic center normalised time distance for previous activity and normalised time distance for next activity.

distance for next activity, yields better results. Notice that even though differences are not very big, the higher precision of the third approach is due to lower false positive rates. For some activities, the first two approaches produce even higher true positives, but at the expense of generating more false positives. Recall values are very similar for all three time metrics in both scenarios. So results suggest that as far as sensor activation labelling regards, combining the dynamic centre normalised time distance and normalised time distance is the best option.

The small differences between three time approaches shown in Tables 6.9 and 6.13 mean that for outsider actions, the candidate function solves the vast majority of the cases (Equation 4.6). For those outsiders that cannot be classified by the candidate function, the three time metrics play a role. But their effect is minimised because of the low number of such outsiders.

For the rest of the discussion about the clustering process, the third time approach will be considered, since it seems to be the best approach. Tables 6.7 and 6.11 show the good performance of the activity clustering process using the context knowledge. True positive rate is very high for all activities. The lowest rate is found for activity ReadBook in the complete scenario: 94.37%. This is due to two factors: (i) the missing sensor probability for pressure sensors is around 10% and (ii) the IAM for ReadBook has the action *useFurniture* ($IAM(ReadBook) = \{hasBook, useFurniture\}$); all the objects that are mapped to action *useFurniture* are monitored by pressure sensors, for example, sofa, chair or bed. When pressure sensors fail, SA^3 cannot detect the activity and hence, true positive rate decreases. Notice that this happens only because the missing action is part of the IAM of the activity. All other activities, even in the complete scenario, show true positive rates higher than 95%, reaching in some cases 100% rates. Those high rates are accompanied with very low rates of false positives and negatives. For example, the highest false positive rate has been found for activity MakePasta in the complete scenario with 2.83%.

It is worth to point out the behaviour of the clustering process, divided into two steps. Tables 6.7 and 6.11 show how SA^3 labels correctly varying number of sensor activations. This number depends on the relation between the number of sensor activations performed by a user and the number of actions in the IAMs. For instance, if the IAM of activity A has two actions and a concrete user performs in average 9 actions for that activity, SA^3 will only label correctly those actions that lie inside the two actions of the IAM. It can be seen that for low action number activities like BrushTeeth and ReadBook, SA^3 shows quite a high true positive rate and low false negative rate. However, activities like MakeChocolate or MakePasta have a true positive rate below 60% and high false negative rates. In any case, in the second step run by AA, true positives rise, false negatives get very low and false positives slightly increase, giving similar results for all activities. This means that SA^3 discovers activities' time locations very accurately and AA treats insider and outsider actions properly to achieve very good rates building on the results of SA^3.

As a conclusion, the clustering process depends on the results of SA^3, its initialisation step. As such, the clustering process is sensitive to sensor missing noise which affects to actions used in IAMs. However, the complete scenario shows sensor missing noise levels extracted from real deployments and results are very good. Notice that experiments are carried out for eight real users, each of them executing activities in many different ways. The high true positive rates and low false positive and negative rates obtained in such realistic set-ups suggest that the activity clustering process annotates very accurately each sensor activation and thus, action, giving as a result robust action clusters, even in noisy scenarios. Remember that while sensor missing noise models have been extracted from real deployments, sensor positive noise has been exaggerated in order to provide very challenging conditions. Even in those conditions, many of the built action clusters will correspond to real activity models, and some others will contain spurious actions, due to false positive annotations. But looking at the low rate of false positives, it can be concluded that correct action clusters will be dominant.

6.2.3 EAM learning performance

6.2.3.1 Evaluation scenarios and metrics

For the evaluation of the EAM learning system, the same datasets used and described for the evaluation of the activity clustering system have been used (see Section 6.2.2.1). The idea is to run the AML algorithm on the clusters extracted by the activity clustering process, using the same activity datasets for the same 8 real users. In consequence, the evaluation scenarios are exactly the same, but they are summarised for convenience:

1. 8 real users.

2. 7 activities of daily living labelled as MakeCoffee, MakeChocolate, MakePasta, BrushTeeth, WatchTelevision, WashHands and ReadBook.

3. 2 scenarios: the ideal scenario and the complete scenario.

4. 3 executions of the activity clustering process, using the three different time metrics. AML is subsequently executed on all three clustering processes independently.

The chosen evaluation criterion to assess the performance of the EAM learning system is named *model-based evaluation*. It compares the learnt activity models with the models provided by users in their answers to the survey. Activity models are compared action-wise, i.e. if a user states that activity A is performed by action sequences S_1 and S_2, those sequences constitute the ground truth. The sequences resulting from the learning process (clustering process + AML) are compared with this ground truth. In this example, it should be checked whether the learning process learns S_1 and S_2 for activity A. The indicators used to show the performance of the EAM learning system are: learnt correct action sequences, learnt spurious action sequences and learnt total action sequences.

Using the described evaluation scenarios and the *model-based evaluation* criterion, the EAM learning system's capability of learning activity models is measured. For all experiments, IAMs have been defined previously and they are always the same. Indeed, the IAMs defined in Figure 6.11 are the ones used for those experiments. This is important, since IAMs aim at being generic and incomplete activity models for every user. Using those IAMs, the EAM learning system should be able to learn extended activity models for the defined activities, and those models should be the same as the models provided by the users in their surveys. Such results would prove the main claim of this dissertation, i.e. complete and specialised activity models can be learnt from user generated data for previously defined incomplete activity models.

6.2.3.2 **Results**

Ideal scenario

The datasets used for those experiments have already been described in Section 6.2.2.1. Remember that the ideal scenario does not contain any sensor noise. AML is run, taking as inputs the results of the clustering process generated in Section 6.2.2.2. As a reference, using a laptop computer equipped with an Intel i5 processor with 4 cores running at 1.6 GHz and 8 GB of RAM memory, the complete EAM learning system needs around 30 seconds to process such datasets. The obtained results are summarised in the following tables: Table 6.14 shows the results for the ideal scenario using the simple time distance in the clustering process; analogously, Tables 6.15 and 6.16 show the results in the same scenario, but applying

Activity	**Learning Results**			**Average Number of Patterns**
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	0	1	1
WatchTelevision	1.14	0	1.14	1.14
BrushTeeth	1.25	0.31	1.56	1.25
WashHands	1	0.37	1.37	1
MakePasta	2	0	2	2
ReadBook	1.12	0	1.12	1.12
MakeCoffee	1.71	0.24	1.71	1.71

Table 6.14: Average results for 8 users of the EAM learning process for the ideal scenario, using the simple time distance in the clustering process.

Activity	**Learning Results**			**Average Number of Patterns**
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	0	1	1
WatchTelevision	1.14	0	1.14	1.14
BrushTeeth	1.25	0.31	1.56	1.25
WashHands	1	0.37	1.37	1
MakePasta	2	0	2	2
ReadBook	1.12	0	1.12	1.12
MakeCoffee	1.71	0	1.71	1.71

Table 6.15: Average results for 8 users of the EAM learning process for the ideal scenario, using the normalised time distance in the clustering process.

Activity	**Learning Results**			**Average Number of Patterns**
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	0	1	1
WatchTelevision	1.14	0	1.14	1.14
BrushTeeth	1.25	0.47	1.72	1.25
WashHands	1	0.25	1.25	1
MakePasta	2	0	2	2
ReadBook	1.12	0	1.12	1.12
MakeCoffee	1.71	0	1.71	1.71

Table 6.16: Average results for 8 users of the EAM learning process for the ideal scenario, using dynamic centre normalised time distance for the previous activity and normalised time distance for the next activity.

the normalised time distance in the first and the combination of the dynamic centre normalised distance and normalised distance in the second.

All the tables contain the same information. Tables are divided in two main columns: (i) *Learning Results*, where the results of the EAM learning system are depicted using the indicators of correct, spurious and total models, and (ii) *Average Number of Patterns*, which shows the average number of different ways to perform the activity by all users. For instance, if a user executes two different action sequences for an activity, and another user executes only one action sequence for the same activity, the average number of patterns for that activity is $(2 + 1)/2 = 1.5$. Average number of patterns is obtained from the surveys to users and constitutes the ground truth for the learning system. On the other hand, learning results are depicted in the first three columns, providing three average values per activity: the correct models learnt by the system, the spurious models learnt and the total number of learnt models, which is the sum of the previous two values. Remember that those values are average values for all 8 users.

Finally, a comparative table is shown in Table 6.17. Average precision and recall are calculated for all activities, using different time distances in the clustering process.

	Avg. Precision	Avg. Recall
t1	91.46%	100%
t2	93.25%	100%
t3	93.25%	100%

Table 6.17: Comparative of applying AML to the clustering process with the three time metrics in the ideal scenario. t1 refers to simple time distance, t2 to normalised time distance and t3 to using dynamic centre normalised time distance for previous activity and normalised time distance for next activity.

Complete scenario

In the following experiments, AML has been run in the complete scenario, where sensor missing and positive noise are applied. Table 6.18 shows the results using simple time distance in the clustering process. Table 6.19 does the same using the normalised time distance. Finally, Table 6.20 presents the obtained numbers using dynamic centre normalised time distance for the previous activity and normalised time distance for the next activity. All three tables show the correct, spurious and total models, alongside the average number of patterns per each activity.

As for the ideal scenario, Table 6.21 provides a summary of the performance of each time distance by means of average precision and recall.

Activity	Learning Results			Average Number of Patterns
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	1.2	2.2	1
WatchTelevision	1.14	0.89	2.03	1.14
BrushTeeth	1.25	1.17	2.42	1.25
WashHands	1	0.75	1.75	1
MakePasta	2	1.12	2.12	2
ReadBook	1.12	0.14	1.26	1.12
MakeCoffee	1.71	1.34	3.05	1.71

Table 6.18: Average results for 8 users of the EAM learning process for the complete scenario, using simple time distance in the clustering process.

Activity	Learning Results			Average Number of Patterns
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	1.2	2.2	1
WatchTelevision	1.14	0.89	2.03	1.14
BrushTeeth	1.25	1.17	2.42	1.25
WashHands	1	0.75	1.75	1
MakePasta	2	1.12	3.12	2
ReadBook	1.12	0.14	1.26	1.12
MakeCoffee	1.71	1.59	3.3	1.71

Table 6.19: Average results for 8 users of the EAM learning process for the complete scenario, using normalised time distance in the clustering process.

Activity	Learning Results			Average Number of Patterns
	Correct Models	Spurious Models	Total Models	
MakeChocolate	1	1.2	2.2	1
WatchTelevision	1.14	0.89	2.03	1.14
BrushTeeth	1.25	1.17	2.42	1.25
WashHands	1	0.75	1.75	1
MakePasta	2	1.12	3.12	2
ReadBook	1.12	0.14	1.26	1.12
MakeCoffee	1.71	1.71	3.42	1.71

Table 6.20: Average results for 8 users of the EAM learning process for the complete scenario, using dynamic centre normalised time distance for the previous activity and normalised time distance for the next activity.

	Avg. Precision	Avg. Recall
t1	59.87%	100%
t2	59.28%	100%
t3	59.02%	100%

Table 6.21: Comparative of applying AML to the clustering process with the three time metrics in the complete scenario. t1 refers to simple time distance, t2 to normalised time distance and t3 to using dynamic centre normalised time distance for previous activity and normalised time distance for next activity.

As the complete scenario is the most realistic one, yet another table is shown to see how many actions are learnt by the EAM learning system compared to the number of actions in the IAMs of each activity. Table 6.22 shows such results using the combination of dynamic centre normalised and normalised time distance, since results for all three time distances are very similar. Once again, average numbers over 8 users are provided. This table is useful to see how far are the IAMs of some activities from real executions by users.

Activity	Actions in IAM	Average Number of Learnt Actions
MakeChocolate	2	5.6
WatchTelevision	2	2.55
BrushTeeth	3	3.5
WashHands	2	2.79
MakePasta	3	6.63
ReadBook	2	2.37
MakeCoffee	2	6.36

Table 6.22: Average number of learnt actions compared to the number of actions in the IAMs of defined activities. Results are obtained for 8 users in the complete scenario, using dynamic centre normalised distance for the previous activity and normalised time distance for the next activity.

6.2.3.3 **Discussion about the EAM learning system**

The most important objective of this dissertation is to evaluate the performance of the complete EAM learning system, which is composed by the activity clustering process (SA^3 + AA) and the AML algorithm. For that purpose, AML is run using as inputs the clusters obtained in the clustering process. It has to be seen whether extended activity models for every user are properly learnt by the system. To answer this question, the results obtained in Section 6.2.3.2 are analysed.

First of all, it is important to remember that the objective of the EAM learning system is to learn properly all activity models really executed by a user. To check whether the learning system accomplishes this objective, the learnt correct models should be equal to the average number of patterns per activity, i.e. if a user has executed one single action sequence to perform an activity, the learning system should learn correctly one action sequence (remember that the results provided in the tables show the average values). The equality between correctly learnt models and the average number of patterns is obtained in all the experiments, regardless of the applied noise and used time metrics for the clustering process. Hence, the EAM learning system is shown to correctly learn the 100% of the varying ways of performing an activity by all users.

Having three time approaches for the clustering process places the question of what the effect of those time approaches is in the EAM learning system. Table 6.17

shows a comparison for the ideal scenario and Table 6.21 for the complete one. While the third time approach is better for the ideal scenario, the same does not happen in the complete one. However, the difference in average precision for the complete scenario is not significant, ranging from 59.87% to 59.02%. Notice that as performed different activity models per activity are not high - the highest value is two for MakePasta -, learning an additional activity model for a user makes quite a big difference in the calculated rates. That is why the slight differences in average precision between three time approaches cannot be considered significant.

The obtained results suggest that AML behaves very robustly regardless the time approach used. Even though the initial number of clusters for time approaches one and two are more than for time approach three, AML is able to detect outliers and fuse them to get always very similar results. So from the point of view of the EAM learning system, it is not easy to say which time approach is the best one. Nevertheless, taking into account that time approach three behaves better for clustering and for EAM learning in the ideal scenario, it seems natural to choose it as the reference time approach. From now on, the discussion will be focused thus on the third time approach.

The desired results for correctly learnt activity models come with the learning of some spurious models, specially in the complete scenario. Learning some spurious or false models was expected, since the objective of the learning algorithm is to avoid removing any activity pattern that has been actually performed by the user. It is preferable to get false action sequences than removing any activity pattern that has been really performed. For that purpose, the learning process is conservative when removing and fusing activity patterns. Even with this conservative approach, it has been observed during experiments that the learning process can reduce clusters provided by the clustering process from 17 to 3 in some cases, thus removing many false action sequences and keeping the 100% of correct action patterns.

In addition to the low number of spurious activity models learnt, it has to be said that those false models are usually easy to discard for an expert for two reasons: (i) they have very low occurrence frequencies and (ii) they usually contain actions that are not generally executed for those activities. For example, for activity MakeChocolate, actions like *hasBacon* have been seen. Notice that those actions cannot be discarded by AA during the clustering process, since they are type and location compatible with MakeChocolate. Such an action is produced by sensor positive noise, which can be observed in the slight false positive increments from ideal scenario to complete scenario in Tables 6.7 and 6.11. Either those actions are insiders that are not properly aggregated by the compatibility function (Equation 4.5), or outsiders that fulfil the candidate function (Equation 4.6).

Adding more knowledge to the context knowledge would allow discarding such actions in AA. For example, if object types state that meat (bacon or sausages) is only used to prepare meals, the algorithm could infer that bacon cannot be used for activity MakeChocolate, which is a sub-class of activity MakeDrink and disjunct of activity MakeMeal. But this brings the initial knowledge balance problem: how much knowledge should be initially provided to such a learning system? The answer depends a lot on the domain. If obtaining and modelling knowledge for a concrete domain is easy, adding knowledge is a good idea. However, obtaining and modelling knowledge can be very expensive in certain domains. The approach presented in this dissertation follows the philosophy of minimising initial knowledge as much as possible, presenting the *worst case scenario*. We believe that the results shown in Section 6.2.3.2 support this decision.

Concluding, the EAM learning system is able to learn complete and specialised activity models for different users in realistic scenarios. As demanded in Chapter 1, all the activity variations executed by a user are learnt. But this is achieved by means of conservative approaches that also generate false activity models. It has been observed that such spurious models are not generally a problem for an expert. It has to be said though, that there are some other spurious models that are very difficult to discard even by an expert. For those cases, adding some spurious models to the knowledge base should not be a big problem, since if the user does not perform that activity model, the recognition system will not detect it.

The EAM learning system is a solid step towards dynamic and personalised knowledge-based activity models. If the learning approach was applied periodically, the learnt activity models would be dynamic. Whenever a user performs an activity differently, the learning system will be able to capture that variation and provide the corresponding activity model. The experiments shown in this dissertation are based on batch learning, but adapting the learning system for incremental learning is only an implementation issue. On the other hand, the EAM learning system also supports personalised activity models. It has been shown that eight real users' personalised ways of performing activities are properly modelled using the same IAMs. Using previous incomplete knowledge about activities, complete and specialised activity models in terms of executed actions are shown to be learnt with user generated data.

6.3 Summary and Conclusions

After analysing the most used evaluation methodology in the community of activity recognition (Section 6.1), it has been concluded that using such methodology to properly assess the performance of the EAM learning system is not feasible. In consequence a novel evaluation methodology has been described in this chapter (Section 6.1.2), following the trend of several researchers who have already analysed the benefits of using simulated environments for activity recognition. Combining a synthetic dataset generator tool with surveys to real users, realistic experimental set-ups can be prepared, introducing varying time lapses and sensor noise. One of the most important novelties of the presented evaluation methodology is to provide a tool to model human behaviour reliably using specially designed surveys.

To test all the relevant parts of the EAM learning system, several experiments or evaluation scenarios have been prepared in Section 6.2. Those experiments have been prepared using the hybrid evaluation methodology. The performance of every algorithm has been measured based on well defined metrics.

The results obtained for all the experiments for SA^3, the complete clustering process and for the EAM learning system can be considered very good, even having high levels of sensor noise - remember that the complete scenario contains more than 38% more sensor activations due to sensor noise than the ideal scenario - . For instance, experiments for SA^3 show how accurate is the algorithm when detecting time locations of performed activities. It cannot label correctly every action of a dataset, but it can very accurately distinguish between action sequences describing different activities. This initialisation, which is specially robust to sensor positive noise, is the key step to make sure that all action sequences will be properly captured in the end of the clustering process. The obtained results show that the AA algorithm analyses every action suitably after the initialisation step, using all the knowledge represented in the context knowledge file. Even though false positives can be generated at this step, the vast majority of actions are properly tagged, taking advantage of action type, location and the defined three time metrics. This can be seen in the high precision and recall values obtained in both scenarios. For example, precision is above 97% in the complete scenario using the third time approach, whereas recall is above 98% (see Table 6.13).

The combined results of both algorithms, SA^3 and AA, allow capturing effectively all the action sequences which have been executed by a user for a concrete activity. At this stage, the most important thing is that the action clusters obtained for each activity, actually contain the real action sequences executed by users. However, there are also spurious variations of those real sequences due to sensor noise and clustering errors. Based on those action clusters AML implements a similarity-based outlier detection algorithm. Such algorithm encodes action sequence information using the Jaccard coefficient and limits the problem of finding spurious models to detect outliers in the similarity space. Using conservative approaches to avoid removing any real action sequence, a statistical outlier detection approach is run. The approach works well because spurious action sequences are very similar to real action sequences. More concretely, as spurious action sequences are slight variations of real action sequences, their similarity values are higher than expected for the similarity distribution of a given activity. AML establishes a threshold using statistics that are robust to outliers. This threshold separates those similarity values that can be considered normal from anomalous or outlier similarity values. The result is that all real activity models are correctly learned, while the number of spurious or false models is acceptable.

Alea jacta est.

Gaius Julius Caesar

CHAPTER 7

Conclusions and Future Work

A general description of the main results and contributions presented in this dissertation is given in this chapter. To finalise the dissertation, the objectives posed in Chapter 1 are reviewed to see at what extent they have been achieved. Furthermore, a summary of all contributions done during the research work is provided, alongside a list of publications which prove how this work has been validated with the research community. The chapter ends with some ideas for future research in the area of activity modelling and some final remarks.

The rest of this chapter is divided in the following sections: Section 7.1 presents a summary of work and the conclusions obtained from the research work and results. Section 7.2 lists all the contributions done through the dissertation. Section 7.3 shows how the objectives and goal of the dissertation are achieved, thus validating the hypothesis with the obtained results. Section 7.4 reviews all relevant scientific publications related to the development of the dissertation. Section 7.5 provides some ideas for future research, extracted from the limitations and weaknesses of the EAM learning system. Section 7.6 makes some final remarks.

7.1 Summary of Work and Conclusions

Activity modelling is a very important step for activity recognition. It provides the models which will be used in the recognition step and thus encode the exact way an activity is performed. There are several desirable features for activity modelling approaches, including:

1. Provide generic activity models which can be applied to any user.
2. Provide personalised activity models which capture the special way an activity is performed by a concrete user.
3. Provide tools to make activity models evolve in time as users vary their behaviour.
4. Provide human-understandable models, to make the usage of such models easy for the development of other applications for intelligent systems.

With the objective of achieving an activity modelling process which fulfils all the listed requirements, a general framework was presented in Chapter 1, represented in Figure 1.1. As a summary, the process starts with an expert who provides generic activity models. Those models are used in the learning system to identify activities in sensor datasets and learn personalised models. The expert reviews the obtained personalised models and includes them in the knowledge base. Expert intervention is convenient because the activity model learning approach will often lead to errors. Repeating those steps continuously in time, dynamic activity models are obtained.

One of the main gaps encountered to achieve the proposed activity modelling process was to learn new actions for already known activities. So the objective of the dissertation was to fill in that gap, combining knowledge-based activity models with data-driven learning techniques. The models learnt would be personal activity models with specific action sequences per user obtained from the sensor datasets generated while performing activities.

This dissertation has introduced a novel approach to acquire complete and specialised knowledge-based activity models through data-driven learning techniques. The approach makes possible using generic but incomplete knowledge-based activity models to learn specialised and complete models, which represent the personal way an activity is performed by a concrete user. Central to the approach is the two-step clustering process (Chapter 4), which has the singularity of using previous knowledge in order to find action clusters and label them with the appropriate activity class. Those clusters are then treated by the learning algorithm in order to acquire extended activity models for different users (Chapter 5).

The results as shown in Chapter 6 indicate that the approach works well in realistic experimental set-ups, with real users' inputs, realistic time intervals for activity sequences and sensor noise. It has to be stressed that all the varying ways of performing an activity by a user are correctly captured by the approach for all users and scenarios. Specialised and complete models for initial generic incomplete activity models can be properly learnt automatically with minimum previous context knowledge. In consequence, the presented approach can be used to make knowledge-based activity models evolve with user behavioural data, offering the tools to solve two of the problems of knowledge-driven approaches: the generic and static nature of activity models.

So it can be concluded that the extended activity model learning system developed in this dissertation has served to validate the hypothesis posed in Section 1.2. It has been shown that personalised activity models can be learnt accurately. Accuracy, in this case, refers to the ability to learn all the actions executed by a user to perform a concrete activity. The 100% of correct activities achieved in the experiments backs the statement.

It is convenient though to remember the scope of the presented learning system. First of all, it works only for the single user - single activity scenario, so interwoven activities are not addressed. The problem of learning new activities is not addressed in this dissertation either. Finally, the approach has been designed for the dense sensing activity monitoring paradigm. In principle, the learning system should be able to work on top of any monitoring approach as far as concrete actions can be obtained.

The evaluation methodology used relies on simulation tools and user surveys (Chapter 6). There are some limitations, since users might omit some details in their answers and the synthetic dataset generator cannot accurately simulate all the possible situations. For example, simulating user erratic behaviour is a challenge. That is why the level of noise introduced in the experiments is so high (around 38% in average). The idea is to introduce high levels of positive sensor noise, much higher than those seen in real pervasive environments (see (Chen et al., 2012a)). The results obtained using this evaluation methodology can be deemed as relevant because it combines: (i) real users' inputs regarding activities, objects, time lapses and locations, (ii) realistic sensor error models obtained from real environments and (iii) high levels of positive sensor noise to properly cover the effects of user erratic behaviour. The most important thing is to be able to capture what users describe in their surveys, showing that new actions can be learnt and different ways of performing the same activity can be identified and properly modelled. The evaluation methodology designed and described in Section 6.1.2 guarantees this. Nevertheless, a final validation on real data would be desirable. Despite the technical difficulties of running such experiments, we are currently working on it. The first idea is to run some experiments with actors and pre-defined scripts in order to obtain some variations for a given list of activities. Afterwards, contacting research groups that own pervasive environments has to be considered, with the objective of running large scale experiments.

In conclusion, the EAM learning system designed and developed throughout this dissertation, is composed by the activity clustering process (Chapter 4) and the Activity Model Learner (AML, Chapter 5). The activity clustering process is divided into two steps: the initialisation of the clustering, provided by the Semantic Activity Annotation Algorithm (SA^3, Section 4.1) and the Action Aggregator algorithm (AA, Section 4.2). This learning system has been designed to cope with the problem of learning personalised activity models in terms of actions, using generic but incomplete activity models provided by a domain expert. Such a problem has been identified in the literature as one of the gaps to achieve an activity modelling process to obtain dynamic and personalised models. Once results are carefully analysed, it can be claimed that the EAM learning system accomplishes its objectives, being able to learn extended activity models for several users. So the EAM learning system is an important step towards a dynamic and personalised knowledge-driven activity modelling process.

7.2 Contributions

A summary of the contributions explained in this dissertation is presented in this section:

- The design architecture for learning extended activity models was described in Chapter 3. This contribution addresses objective 2 and the design part of objective 3 (Section 1.2).
  - It offers a convenient modular architecture which separates key concepts and allows the implementation of different approaches.
  - Communication between different modules is carried out by means of files which follow standard formats, such as JavaScript Notation (JSON) and Comma Separated Values (CSV).
  - It uses a light-weight knowledge representation and processing framework based on JavaScript Object Notation (JSON) technology, presenting compact and easy-to-extend structures for activities, objects, locations and sensors, to achieve environment independence.
- A novel two-step activity clustering process which uses previous knowledge in unlabelled sensor datasets is explained in Chapter 4. This contribution tackles part of objective 3, defined in Section 1.2.
  - The clustering algorithms found in the literature allow including prior knowledge in form of constraints or some points' memberships, limiting substantially the kind of knowledge that can be provided to the

algorithms. However, the approach presented in this dissertation specifies prior knowledge as incomplete models, allowing higher levels of expressiveness.

  - It allows grouping data in clusters and recognises each cluster's label using previously given incomplete models. Even though it is an unsupervised approach, the activity clustering algorithm provides the semantic labels of each cluster. No previous work shows clustering approaches with labelling capacities, due to the way prior knowledge is represented and incorporated.
  - It works on the activity space, which is composed by location, type and time axes, providing a meaningful low-dimensional space for efficient action clustering.
  - Typical clustering algorithms define distance functions in the corresponding space to group points that are close to each other. Defining continuous distance metrics in the activity space is not possible, so heuristic functions which exploit domain knowledge and time distance metrics are used to process actions. Heuristic functions process the information provided by discontinuous location and type axes, whereas time distance metrics work on the continuous axis of time. The activity clustering approach is a good example of how clustering can be performed in discontinuous spaces.

- An activity model learning algorithm was presented in Chapter 5, which uses action clusters that define activities to extract activity models. With this contribution, objective 3 is completed (Section 1.2)

  - It has a flexible filtering step to pre-process the action clusters in form of action sequences found by the activity clustering algorithm.
  - It implements a model similarity-based outlier detection statistical algorithm to detect spurious action sequences. The main novelty of the approach is to interpret outliers as pairs of a valid action sequence and its spurious variation. Hence, the algorithm fuses both action sequences, taking into account the occurrence frequencies of both action sequences.
  - It outputs complete and specialised activity models for a concrete user alongside with their occurrence frequencies.

- A hybrid evaluation methodology was designed and implemented to properly evaluate the learning system in Chapter 6. The methodology can also be used for further activity modelling and recognition approaches. This contribution addresses objective 4 (Section 1.2).

- Following previous work on simulation for pervasive environments, the hybrid evaluation methodology offers a realistic and cost-efficient way to evaluate activity modelling and recognition approaches.
- It addresses one of the major problems of simulation-based approaches, which is the problem of modelling human behaviour. Surveys to users are designed in order to capture how activities are performed by them and model effectively their behaviour.
- Using surveys, it allows obtaining the information of any number of users minimising time-cost.
- It provides a special simulator to generate sensor datasets, namely the synthetic dataset generator. Its usage allows generating on demand any kind of dataset representing different activity executions. The synthetic dataset generator allows modelling two kinds of sensor errors using probabilistic models: sensor positive and missing errors. The reviewed previous work does not address the simulation of sensor noise for dense sensing monitoring scenarios.
- Combining surveys and simulation tools, any kind of experiment can be prepared generating as much data as needed, in contrast with standard methodologies, where experiments are very expensive and testing different situations is very difficult.

7.3 Hypothesis and Objective Validation

In the beginning of this dissertation, concretely in Section 1.2, a hypothesis was posed, which stated the following:

Hypothesis. *Using domain experts' previous knowledge as generic but incomplete activity models and data-driven learning techniques on user generated data, it is possible to learn accurately new actions to obtain personalised activity models for every user.*

In order to be able to validate this hypothesis, a goal was also defined, which is also shown below for convenience:

Goal. *To design and implement a learning system that uses generic but incomplete activity models to analyse unlabelled activity sensor datasets and acquire personalised models which contain new actions.*

To achieve such a goal, some specific and measurable objectives were identified in Section 1.2. It is the moment now to go through all those objectives and see how they have been addressed in this dissertation.

1. *To study the current state of the art on knowledge- and data-driven activity modelling and recognition.* Chapter 2 provides a deep and critical study of the current state of the art, analysing activity monitoring, modelling and recognition. Special attention has been paid to activity modelling, presenting data-driven, knowledge-driven and even hybrid approaches and identifying their advantages and disadvantages.

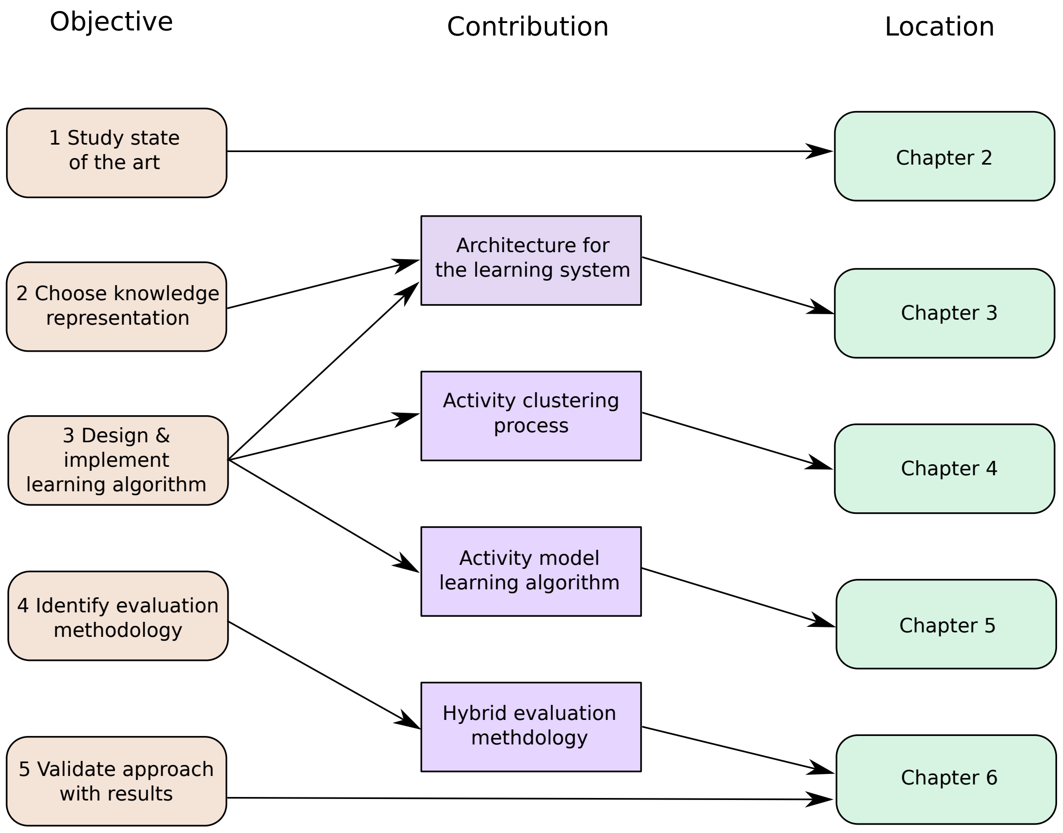
2. *To choose a knowledge representation formalism and design proper structures to represent domain experts' knowledge.* This objective has been addressed in Chapter 3. A light-weight knowledge representation formalism has been chosen, namely the JavaScript Object Notation framework (JSON). Using JSON, the context knowledge file shows structures to represent activities, objects and sensors.

3. *To design and implement a multiple step learning algorithm which uses previous knowledge and user generated data to obtain personalised activity models.* The design of the learning system and its rationale have been presented in Chapter 3. Moreover, implementation details of the system architecture have been provided, showing samples of the used files. The modules identified in the architecture, which constitute the multiple step learning process have been explained in the following chapters: Chapter 4 introduces the activity clustering process and Chapter 5 describes in detail the activity model learner algorithm. All the implementation work has been done using Python 2.7.

4. *To identify the evaluation methodology for the learning system which better addresses the requirements of the system.* Chapter 6 analyses the so called standard methodology for activity modelling and recognition. It is concluded that using the standard methodology is not feasible to properly validate the learning system. In consequence, a novel hybrid methodology is developed and presented in Section 6.1.2, which combines real users' inputs and a simulator.

5. *To validate the obtained results quantitatively, with the objective of capturing the 100% of real activity models performed by a user.* The results of several experiments are presented in Chapter 6. It has been shown that the 100% of action sequences executed by several users to perform the same seven activities are captured and modelled (Section 6.2.3.2). These results are obtained in scenarios which contain high levels of sensor noise.

For the sake of clarity, Figure 7.1 provides a diagram where the objectives, the contributions that are related to these objectives and their location in this dissertation are shown.



Objective
Contribution
Location
1 Study state of the art
2 Choose knowledge representation
3 Design & implement learning algorithm
4 Identify evaluation methodology
5 Validate approach with results
Architecture for the learning system
Activity clustering process
Activity model learning algorithm
Hybrid evaluation methdology
Chapter 2
Chapter 3
Chapter 4
Chapter 5
Chapter 6

Figure 7.1: The relation of the objectives defined in Section 1.2, the contributions listed in Section 7.2 and their location in this dissertation.

It can be claimed that all the objectives set in Section 1.2 have been accomplished in this dissertation. As a consequence of this, the goal of the dissertation has been achieved. The EAM learning system which is composed by the activity clustering process (Chapter 4) and the activity model learner (Chapter 5) has been implemented and tested. The learning system uses generic but incomplete activity models and unlabelled sensor datasets. As a result of the learning process, personalised activity models have been shown to be learnt, capturing personal ways of performing activities in terms of actions.

Finally, through the accomplishment of the objectives and the goal, the hypothesis of this dissertation has been validated. The results obtained in Chapter 6 prove that it is possible to learn accurately new actions to obtain personalised activity models for every user, combining generic but incomplete activity models and data-driven learning techniques. Section 6.2.3.2 shows the results that prove that all the actions executed by several users to perform different activities are accurately captured and modelled. Several spurious models are also learnt in the process, but the number of such models is assumable as discussed in Section 6.2.3.3.

7.4 Relevant Publications

The contributions of this dissertation have been presented to the scientific community in a series of international forums, such as journals and conferences.

7.4.1 International JCR journals

The current version of the EAM learning system was published in the following journal:

- Gorka Azkune, Aitor Almeida, Diego López-de-Ipiña, Liming Chen. Extending Knowledge-Driven Activity Models through Data-Driven Learning Techniques. Expert Systems with Applications, vol. 42, no. 6, pp. 3115-3128. Pergamon-Elsevier Science LTD. United States. DOI: 10.1016/j.eswa.2014.11.063, ISSN 0957-4174, JCR Impact Factor (2013): 1.965, Q1. April 2015.

The formalisation of the hybrid evaluation methodology and the synthetic dataset generator was published in the following journal:

- Gorka Azkune, Aitor Almeida, Diego López-de-Ipiña, Liming Chen. Combining Users' Activity Survey and Simulators to Evaluate Human Activity Recognition Systems. MDPI Sensors. vol. 14, no. 4, pp. 8192-8213, DOI: 10.3390/s150408192, ISSN 1424-8220, JCR Impact Factor (2013): 2.048, Q1. Basel, Switzerland, April 2015.

An up-to-date survey of the state of the art in activity modelling can be found in the following journal:

- Gorka Azkune, Aitor Almeida, Diego López-de-Ipiña, Liming Chen. Latest Trends in Human Activity Modelling. Dyna. JCR Impact Factor (2013): 0.200, Q4. Accepted, to be published.

The preliminary work that led to the research developed in this dissertation was published in the following journal:

- Gorka Azkune, Pablo Orduña, Xabier Laiseca, Eduardo Castillejo, Diego López-de-Ipiña, Miguel Loitxate and Jon Azpiazu. Semantic Framework for Social Robot Self-Configuration. MDPI Sensors. vol. 13, no. 6, pp. 7004-7020, DOI: 10.3390/s130607004, ISSN 1424-8220, JCR Impact Factor (2013): 2.048, Q1. Basel, Switzerland, May 2013.

7.4.2 International conferences

The Semantic Activity Annotation Algorithm (SA^3) was presented in the following conference:

- Gorka Azkune, Aitor Almeida, Diego López-de-Ipiña, Liming Chen. A Knowledge-Driven Tool for Automatic Activity Dataset Annotation. Proceedings of the 7th International Conference Intelligent Systems IEEE IS'2014, September 24-26, 2014, Warsaw, Poland, Volume 1: Mathematical Foundations, Theory, Analyses. ISBN: 978-3-319-11312-8, pp. 593-604.

The hybrid evaluation methodology was presented in the following conference:

- Gorka Azkune, Aitor Almeida, Diego López-de-Ipiña, Liming Chen. A Hybrid Evaluation Methodology for Human Activity Recognition Systems. In Proceedings of 8th International Conference on Ubiquitous Computing & Ambient Intelligence, UCAmI 2014, December 2-4, 2014, Belfast, United Kingdom. ISBN: 978-2-319-13101-6, pp. 92-99.

Preliminary work which led to the research done in this dissertation was presented in the following conferences:

- Gorka Azkune, Pablo Orduña, Xabier Laiseca, Diego López-de-Ipiña, Miguel Loitxate. Semantic Based Self-configuration Approach for Social Robots in Health Care Environments. Ambient Assisted Living and Home Care. Proceedings of 4th International Workconference on Ambient Assisted Living, IWAAL 2012, Vitoria-Gasteiz, Spain, December 3-5, 2012. ISBN: 978-3-642-35394-9, pp. 354-361.

- Gorka Azkune, Mikel Astiz, Urko Esnaola, Unai Antero, José Vicente Sogorb, Antonio Alonso. A Navigation System for a High-Speed Professional Cleaning Robot. Towards Autonomous Robotic Systems. Proceedings of 12th Annual Conference, TAROS 2011, Sheffield, UK, August 31 - September 2, 2011. ISBN: 978-3-642-23231-2, pp. 24-35.

7.5 Future Work

Inspired by the weaknesses and limitations of the research presented in this dissertation, the following further research lines have been identified:

7.5.1 Integrate the EAM learning system in a complete activity modelling process

As has been said in Section 7.1, the EAM learning system is a solid step towards a dynamic and personalised activity modelling system. However, there are some other pieces to complete the jigsaw puzzle. For example, Chen et al. developed a learning approach to learn descriptive properties of knowledge-based activity models in (Chen et al., 2014). They also presented a mechanism to discover new activities.

Combining action and descriptive properties learning schemes with new activity discovery mechanisms in the same activity modelling system is still a challenge. The building blocks are already there, but their integration is not straightforward and it is beyond a technical issue.

Some of the posed challenges are:

1. Defining forgetting strategies for activity models that are not being executed any more by a concrete user. As a user changes her way to perform activities, models which populate the knowledge base but are not being observed should be given a special treatment.

2. Integrating the approaches to learn action and descriptive properties. Both aspects, actions and descriptive properties, are tightly coupled in an activity model. However, there might be the case where the same activity model in terms of actions, produces different patterns of descriptive properties. A proper modelling design has to be set up in order to attain the complex relations between actions and descriptive properties.

3. Obtaining generic models from discovered activities. As the most natural way to discover new activities is observing personal data, discovered activities will be by definition personalised models. The question of how generic

models can be extracted from those discovered personalised models is very important.

7.5.2 Discard meaningless user-object interactions

The monitoring and modelling approach relies on user-object interactions monitored by simple sensors. However, not all the interactions registered by the sensors are meaningful, in the sense that they are not always directed to the execution of a concrete activity. This phenomenon has been identified in the dissertation as being mainly caused by user erratic behaviour. The consequence of meaningless user-object interactions is the appearance of false positive actions within activity models in the learning process. Thus, research on sensor-action mapping steps to distinguish between meaningful and meaningless user-object interactions is needed.

A possible approach could be to add a sensor-action mapping step, where only those sensor activations that last for a concrete amount of time are mapped to actions. Such a processing step would require monitoring sensor state changes from interaction state to no-interaction state and measuring state changes time lapses. Another criterion which can be used for sensor-action mapping step is to monitor how many times a user interacts with an object in a time interval, even though those interactions are short. Combining both criteria, only meaningful object interactions could be identified. Such an approach would allow applying the same EAM learning algorithm to probably obtain better results, reducing the false positive rate of learnt activity models.

7.5.3 Single user - concurrent activities

Another promising future research direction is to extend the learning approach to the single user - concurrent activities scenario. People do not usually perform activities sequentially, but they tend to interleave activities, such as washing dishes while preparing pasta. This will have an impact in the clustering process, demanding more complex pattern recognition and time management. For instance, SA^3 uses the single user - single activity constraint for its pattern recognition algorithm, and AA defines insider and outsider actions based on the same constraint. If concurrent activities are considered, the clustering process should be changed. However, some key ideas could be maintained.

A first idea to tackle such scenarios is to invert the sequence of the clustering process. In the single user - single activity scenario, activities are sequentially ordered in the time axis, which makes the usage of SA^3 as the initialisation step convenient. But if activities are not sequentially arranged in time, initialisation is more complicated. It seems that defining more generic metrics in the activity space, considering time, location and type information, would generate some initial clusters formed by several activities per cluster. Once those clusters are obtained, initial activity models could be used to try to analyse each cluster and see how many activity combinations in terms of initial activity models can be found. This way, several action sequences per activity could be found, to afterwards apply the Activity Model Learner as described in this dissertation.

7.5.4 **Perception for complex actions**

Dense sensing monitoring cannot capture all the actions performed by a user with an object. For example, it is not the same to grab a bottle or to open a bottle. But to distinguish between those two actions with the same object is very complicated - if not impossible - following the dense sensing monitoring approach.

However, activity models would greatly benefit from the usage of more complex actions, because descriptions would be more accurate. Being able to use actions such as *opensBottle*, *poursBottleContent*, *closesBottle* and so on would be a big step forward. It seems that vision-based monitoring is the best option for this perception level. Looking at the recent expectation around head mounted devices with cameras like Google Glass[1], making research on their benefits for activity monitoring would be very interesting. Head mounted devices allow receiving information about the concrete actions a user is executing, because people tend to look to objects which are being manipulated. Besides, privacy problems are mitigated, because head mounted devices do not record the user externally and they do not have to be necessarily working continuously. Another advantage of such devices is the interaction capabilities they offer. In the case of Google Glass, voice interaction, gestures and a small screen are available, which can help developing many applications around activity recognition systems.

Taking into account the recent progress made in artificial vision and the expected proliferation of head mounted devices, vision-based activity monitoring may become a very good option for activity recognition systems.

7.5.5 **Learn temporal relations between actions**

Activity models, as defined and used in this dissertation, do not provide any dependency information between actions. They are simple sequences of actions. Nevertheless, when performing an activity, actions have usually some temporal dependencies which can offer new information for recognition. For example, when preparing fish, fish should not be introduced in the oven until the oven has been switched on. There are some constraints and dependencies between actions that are not currently modelled.

[1]https://www.google.com/glass/start/

Thus, the first step is to design modelling approaches to properly address the dependencies and constraints between actions. The second step is to learn those dependencies from user generated data. A learning solution might be the following: once action clusters defining activities have been extracted, the order in which actions have been executed can be analysed. Using frequency analysis and some other learning techniques, it may be feasible to discover and learn temporal relations between actions.

A final research question regarding action dependencies is how this information can be used for activity recognition. It has to be analysed whether modelling dependencies offers any advantage in the recognition step.

7.6 Final Remarks

With this dissertation we have tried to make significant contributions in the area of activity modelling and recognition, which is deemed to play an important role in human-centred intelligent systems. We hope that the presented contributions and identified future research lines will serve to other researchers as a guide to deliver further contributions and thus keep advancing in such an important research area.

Bibliography

Aipperspach, R., Cohen, E., and Canny, J. (2006). Modeling human behavior from simple sensors in the home. In *Pervasive Computing*, pages 337–348. Springer Berlin Heidelberg. 20

Akdemir, U., Turaga, P., and Chellappa, R. (2008). An ontology based approach for activity recognition from video. In El-Saddik, Abdulmotaleb; Vuong, Son; Griwodz, Carsten; Del Bimbo, Alberto; Candan, K. Selçuk; Jaimes, A., editor, *Proceedings of the 16th ACM international conference on Multimedia*, pages 709–712, Vancouver, Canada. ACM. 12, 36

Alemdar, H. and Ersoy, C. (2010). Wireless sensor networks for healthcare: A survey. *Computer Networks*, 54(15):2688–2710. 12

Almeida, A. and López-de-Ipiña, D. (2012). Assessing ambiguity of context data in intelligent environments: towards a more reliable context managing system. *MDPI Sensors*, 12(4):4934–51. 38

Ashbrook, D. and Starner, T. (2003). Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286. 18

Bao, L. and Intille, S. (2004). Activity recognition from user-annotated acceleration data. In Science, L. N. i. C., editor, *Pervasive Computing*, pages 1–17, Vienna, Austria. Springer Berlin Heidelberg. 17, 18, 20, 22, 24, 26

Bouchard, B., Giroux, S., and Bouzouane, A. (2006). A Smart Home Agent for Plan Recognition of Cognitively-impaired Patients. *Journal of Computers*, 1(5):53–62. 34

Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden Markov models for complex action recognition. In *Proceedings of Computer Vision and Pattern Recognition*, pages 994–999. IEEE. 14, 23

Brdiczka, O. (2009). Learning situation models in a smart home. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 39(1):56–63. 24

Brdiczka, O., Reignier, P., and Crowley, J. (2007). Detecting individual activities from video in a smart home. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 363–370. Springer Berlin Heidelberg. 22

Bruneau, J., Jouve, W., and Consel, C. (2009). DiaSim: A parameterized simulator for pervasive computing applications. In *Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International*, pages 1–10. IEEE. 101

Buettner, M. and Prasad, R. (2009). Recognizing daily activities with RFID-based sensors. In *Proceedings of the 11th international conference on Ubiquitous computing*, pages 51–60. ACM. 17, 21

Carberry, S. (2001). Techniques for plan recognition. *User Modeling and User-Adapted Interaction*, 11(1-2):31–48. 34

Chan, M., Estève, D., Escriba, C., and Campo, E. (2008). A review of smart homes—Present state and future challenges. *Computer methods and programs in biomedicine*, 91(1):55–81. 17, 20

Chen, D., Yang, J., and Wactlar, H. (2004). Towards automatic analysis of social interaction patterns in a nursing home environment from video. In *Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval*, pages 283–290. ACM. 35

Chen, D., Yang, J., and Wactlar, H. (2005). A study of detecting social interaction with sensors in a nursing home environment. In *Computer Vision in Human-Computer Interaction*, pages 199–210. Springer Berlin Heidelberg. 25

Chen, L., Hoey, J., Nugent, C., Cook, D., and Yu, Z. (2012a). Sensor-based activity recognition. *IEEE Transactions on Systems, Man, and Cybernetics-Part C*, 42(6):790–808. 13, 16, 147

Chen, L. and Nugent, C. (2008). A logical framework for behaviour reasoning and assistance in a smart home. *International Journal of Assistive Robotics and Mechatronics*, 9(4):20–34. 35

Chen, L. and Nugent, C. (2009). Ontology-based activity recognition in intelligent pervasive environments. *International Journal of Web Information Systems*, 5(4):410–430. 37

Chen, L., Nugent, C., and Mulvenna, M. (2009). Semantic smart homes: towards knowledge rich assisted living environments. *Intelligent Patient Management*, pages 279–296. 36, 37

Chen, L., Nugent, C., and Okeyo, G. (2014). An Ontology-based Hybrid Approach to Activity Modeling for Smart Homes. *IEEE Transaction on Human-Machine Systems*, 44(1):92–105. 2, 4, 39, 42, 48, 155

Chen, L., Nugent, C., and Wang, H. (2012b). A knowledge-driven approach to activity recognition in smart homes. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):961–974. xix, 3, 12, 37, 44, 47, 85, 109, 129

Choudhury, T. and Consolvo, S. (2008). The mobile sensing platform: An embedded activity recognition system. *IEEE Pervasive Computing*, 7(2):32–41. 1, 12

Cook, D., Feuz, K., and Krishnan, N. (2013). Transfer learning for activity recognition: a survey. *Knowledge and information systems*, 36(3):537–556. 27, 28

Cook, D. and Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480. 12, 22

Ding, D., Cooper, R., Pasquina, P., and Fici-Pasquina, L. (2011). Sensor technology for smart homes. *Maturitas*, 69(2):131–136. 12

Fernández-Caballero, A. (2012). Human activity monitoring by local and global finite state machines. *Expert Systems with Applications*, 39(8):6982–6993. 1

Finni, T., Hu, M., and Kettunen, P. (2007). Measurement of EMG activity with textile electrodes embedded into clothing. *Physiological measurement*, 28(11):1405. 18

Fishkin, K., Philipose, M., and Rea, A. (2005). Hands-On RFID: Wireless Wearables for Detecting Use of Objects. In *ISWC*, pages 18–43. 21

Foerster, F. and Fahrenberg, J. (2000). Motion pattern and posture: correctly assessed by calibrated accelerometers. *Behavior research methods, instruments, & computers*, 32(3):430–457. 17

Fong, T., Nourbakhsh, I., and Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and autonomous systems*, 42(3):143–166. 1, 12

Francois, A., Nevatia, R., and Hobbs, J. (2005). VERL: an ontology framework for representing and annotating video events. *MultiMedia, IEEE*, 12(4):76–86. 36

Galata, A., Johnson, N., and Hogg, D. (1999). Learning structured behaviour models using variable length Markov models. In *IEEE International Workshop on Modelling People*, pages 95–102. IEEE. 22

Gavrila, D. (1999). The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98. 22

Gellersen, H., Schmidt, A., and Beigl, M. (2002). Multi-sensor context-awareness in mobile devices and smart artifacts. *Mobile Networks and Applications*, 7(5):341–351. 17, 19

Georis, B., Maziere, M., Bremond, F., and Thonnat, M. (2004). A video interpretation platform applied to bank agency monitoring. In *Proc. of the 2nd Workshop Intell. Distributed Surveillance System*, pages 46–50. 35

Golding, A. and Lesh, N. (1999). Indoor navigation using a diverse set of cheap, wearable sensors. In *Wearable Computers, 1999. Digest of Papers. The Third International Symposium*, pages 29–36. IEEE. 16

Gu, T., Wu, Z., Tao, X., Pung, H., and Lu, J. (2009). epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In IEEE, editor, *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. 17, 21, 26

Guralnik, V. and Haigh, K. (2002). Learning models of human behaviour with sequential patterns. In *Proceedings of the AAAI-02 workshop "Automation as Caregiver"*, pages 24–30. 26

Hakeem, A. and Shah, M. (2004). Ontology and taxonomy collaborated framework for meeting classification. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, pages (4)219–222. 35, 36

Hamming, R. (1950). Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160. 86

Harms, H., Amft, O., Tröster, G., and Roggen, D. (2008). Smash: A distributed sensing and processing garment for the classification of upper body postures. In *Proceedings of the ICST 3rd international conference on Body area networks*, page 22. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering). 18

Helal, A., Cho, K., and Lee, W. (2012). 3D modeling and simulation of human activities in smart spaces. In *Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pages 112–119. IEEE. 101, 114

Helal, S., Lee, J., and Hossain, S. (2011). Persim-Simulator for human activities in pervasive spaces. In *Intelligent Environments (IE), 2011 7th International Conference on*, pages 192–199. IEEE. 100, 114

Helal, S., Mann, W., and El-Zabadani, H. (2005). The gator tech smart house: A programmable pervasive space. *Computer*, 38(3):50–60. 20

Helaoui, R., Niepert, M., and Stuckenschmidt, H. (2011). Recognizing interleaved and concurrent activities using qualitative and quantitative temporal relationships. *Pervasive and Mobile Computing*, 7(6):660–670. 39

Helaoui, R., Riboni, D., and Stuckenschmidt, H. (2013). A probabilistic ontological framework for the recognition of multilevel human activities. In *Ambient Intelligence and Future Trends-International Symposium on Ambient Intelligence (ISAmI 2010)*, pages 247–254, Guimaraes, Portugal. Springer Berlin Heidelberg. 38

Hoaglin, D., Mosteller, F., and Tukey, J. (1983). *Understanding robust and exploratory data analysis*. Wiley, New York. 88

Hodges, M. and Pollack, M. (2007). An object-use fingerprint: The use of electronic sensors for human identification. In *Proc. of the 9th International Conference on Ubiquitous Computing*, pages 289–303. 21

Hollosi, D. and Schroder, J. (2010). Voice activity detection driven acoustic event classification for monitoring in smart homes. In *Applied Sciences in Biomedical and Communication Technologies (ISABEL), 2010 3rd International Symposium on*, pages 1–5. 20

Horvitz, E., Breese, J., and Heckerman, D. (1998). The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 256–265. 23

Hu, D. and Yang, Q. (2008). CIGAR: Concurrent and Interleaving Goal and Activity Recognition. In *AAAI*, pages (8)1363–1368. 25

Jain, A. and Dubes, R. (1988). *Algorithms for clustering data*. Prentice Hall. 86

Kan, P., Huq, R., and Hoey, J. (2011). The development of an adaptive upper-limb stroke rehabilitation robotic system. *Journal of neuroengineering and rehabilitation*, 8(1):1–18. 23

Kasteren, T. V. and Noulas, A. (2008). Accurate activity recognition in a home setting. In *Proceedings of the 10th International conference on Ubiquitous computing*, pages 1–9. 12

Kautz, H. (1991). *A Formal Theory of Plan Recognition and its Implementation, Reasoning About Plans*. Morgan Kaufmann, San Mateo. 34

Kautz, H., Arnstein, L., and Borriello, G. (2002). An overview of the assisted cognition project. In *AAAI-2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care*, pages 60–65. 23

Kern, N., Schiele, B., and Junker, H. (2003). Wearable sensing to annotate meeting recordings. *Personal and Ubiquitous Computing*, 7(5):263–274. 17

Klein, M., Schmidt, A., and Lauer, R. (2007). Ontology-centred design of an ambient middleware for assisted living: The case of soprano. In *Proc. of the 30th Annual German Conference on Artificial Intelligence*, Osnabrück. 36, 37

Laerhoven, K. V. (2001). Real-time analysis of data from many sensors with neural networks. In *Wearable Computers, 2001. Proceedings. Fifth International Symposium on*, pages 115–122. 17

Laerhoven, K. V. and Aidoo, K. (2001). Teaching context to applications. *Personal and Ubiquitous Computing*, 5(1):46–49. 1, 12, 17

Latfi, F., Lefebvre, B., and Descheneaux, C. (2007). Ontology-Based Management of the Telehealth Smart Home, Dedicated to Elderly in Loss of Cognitive Autonomy. *CEUR Workshop Proceedings*, 258. 36, 37

Lee, S. and Mase, K. (2002). Activity and location recognition using wearable sensors. *IEEE pervasive computing*, 1(3):24–32. 17, 18

Leonhardt, U. and Magee, J. (1998). Multi-sensor location tracking. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 203–214. ACM. 16

Lester, J., Choudhury, T., and Kern, N. (2005). A Hybrid Discriminative/Generative Approach for Modeling Human Activities. In *IJCAI*, pages (5)766–772. 26

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710. 86

Liao, L., Fox, D., and Kautz, H. (2006). Location-based activity recognition. *Advances in Neural Information Processing Systems*, 18:787. 101

Liao, L., Fox, D., and Kautz, H. (2007a). Hierarchical conditional random fields for GPS-based activity recognition. In *Robotics Research*, pages 487–506. Springer Berlin Heidelberg. 17, 25

Liao, L., Patterson, D., Fox, D., and Kautz, H. (2007b). Learning and inferring transportation routines. *Artificial Intelligence*, 171(5):311–331. 18, 23

Lukowicz, P., Ward, J., Junker, H., and Stäger, M. (2004). Recognizing workshop activity using body worn microphones and accelerometers. In *Pervasive Computing*, pages 18–32. 18

Mahdaviani, M. and Choudhury, T. (2008). Fast and scalable training of semi-supervised crfs with application to activity recognition. In *Advances in Neural Information Processing Systems*, pages 977–984. 26

Mantyjarvi, J. (2001). Recognizing human motion with multiple acceleration sensors. In *Systems, Man, and Cybernetics, 2001 IEEE International Conference on*, pages 747–752. IEEE. 17, 18

Maurer, U. and Rowe, A. (2006). Location and activity recognition using eWatch: A wearable sensor platform. In *Ambient Intelligence in Everyday Life*, pages 86–102. Springer Berlin Heidelberg. 22, 24

Mikic, I., Huang, K., and Trivedi, M. (2000). Activity monitoring and summarization for an intelligent meeting room. In *Human Motion, 2000. Proceedings. Workshop on*, pages 107–112. IEEE. 12

Modayil, J., Levinson, R., and Harman, C. (2008). Integrating Sensing and Cueing for More Effective Activity Reminders. In *AAAI Fall Symposium: AI in Eldercare: New Solutions to Old Problems*, pages 60–66. 26

Moeslund, T., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126. 15, 22

Mozer, M. (1998). The neural network house: An environment hat adapts to its inhabitants. In *Proc. AAAI Spring Symp. Intelligent Environments*, pages 110–114. 16

Nevatia, R., Hobbs, J., and Bolles, B. (2004). An ontology for video event representation. In *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*, page 119. IEEE. 36

Nugent, C. and Mulvenna, M. (2009). Experiences in the development of a smart lab. *International Journal of Biomedical Engineering and Technology*, 2(4):319–331. 17, 20

Okeyo, G. and Chen, L. (2012). A hybrid ontological and temporal approach for composite activity modelling. In *IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. 38

Okeyo, G., Chen, L., Wang, H., and Sterritt, R. (2012). Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. *Pervasive and Mobile Computing*, 10:155–172. 101, 107, 114, 116

Oliver, N., Garg, A., and Horvitz, E. (2004). Layered representations for learning and inferring office activity from multiple sensory channels. *Computer Vision and Image Understanding*, 96(2):163–180. 23

Omar, F., Sinn, M., and Truszkowski, J. (2010). Comparative analysis of probabilistic models for activity recognition with an instrumented walker. In *Proc. of the 26th Conference on Uncertainty in Artificial Intelligence*, pages 392–400. 26

Palmes, P., Pung, H., Gu, T., Xue, W., and Chen, S. (2010). Object relevance weight pattern mining for activity recognition and segmentation. *Pervasive and Mobile Computing*, 6(1):43–57. 32

Pantelopoulos, A. and Bourbakis, N. (2010). A survey on wearable sensor-based systems for health monitoring and prognosis. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 40(1):1–12. 12

Patterson, D. and Fox, D. (2005). Fine-grained activity recognition by aggregating abstract object usage. In *Wearable Computers, 2005. Proceedings. Ninth IEEE International Symposium on*, pages 44–51. IEEE. 17, 21

Patterson, D., Liao, L., Fox, D., and Kautz, H. (2003). Inferring high-level behavior from low-level sensors. In *UbiComp 2003: Ubiquitous Computing*, pages 73–89. Springer Berlin Heidelberg. 17, 18, 20

Pentney, W., Philipose, M., and Bilmes, J. (2008). Structure Learning on Large Scale Common Sense Statistical Models of Human State. In *AAAI*, pages 1389–1395. 26

Pentney, W., Philipose, M., Bilmes, J., and Kautz, H. (2007). Learning large scale common sense models of everyday life. In *Proceedings of the National Conference on Artificial Intelligence*, pages 465–470. 26

Perkowitz, M., Philipose, M., Patterson, D., and Fishkin, K. (2004). Mining models of human activities from the web. In *Proc. of the 13th International World Wide Web Conference*, pages 573–582. 31, 32

Philipose, M. and Fishkin, K. (2004). Inferring activities from interactions with objects. *Pervasive Computing*, 3(4):50–57. 1, 12, 17, 21

Poppe, R. (2010). A survey on vision-based human action recognition. *Image and vision computing*, 28(2):976–990. 12, 15

Quinn, J. (2009). Factorial switching linear dynamical systems applied to physiological condition monitoring. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(9):1537–1551. 22

Randell, C. and Muller, H. (2000). Context awareness by analysing accelerometer data. In *Wearable Computers, The Fourth International Symposium on*, pages 175–176. IEEE. 17

Rashidi, P. and Cook, D. (2011). Discovering activities to recognize and track in a smart environment. *Knowledge and Data Engineering, IEEE Transactions on*, 23(4):527–539. 2, 26, 80, 98, 115

Rashidi, P. and Cook, D. (2013). COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):64. 26

Ravi, N., Dandekar, N., Mysore, P., and Littman, M. (2005). Activity recognition from accelerometer data. In *AAAI*, pages (5) 1541–1546. 25, 26

Riboni, D. and Bettini, C. (2011a). COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289. 39, 98

Riboni, D. and Bettini, C. (2011b). OWL 2 modeling and reasoning with complex human activities. *Pervasive and Mobile Computing*, 7(3):379–395. 37

Roberts, D. (1986). Ordination on the basis of fuzzy set theory. *Vegetatio*, 66(3):123–131. 86

Sánchez, D., Tentori, M., and Favela, J. (2008). Activity recognition for the smart hospital. *Intelligent Systems, IEEE*, 23(2):50–57. 12

Schmidt, A., Beigl, M., and Gellersen, H. (1999). There is more to context than location. *Computers & Graphics*, 23(6):893–901. 17

Schmidt, A. and Laerhoven, K. V. (2001). How to build smart appliances? *Personal Communications, IEEE*, 8(4):66–71. 17, 19

Seker, S., Altun, O., Ayan, U., and Mert, C. (2014). A Novel String Distance Function based on Most Frequent K Characters. *International Journal of Machine Learning and Computing (IJMLC)*, 4(2):177–183. 86

Shanahan, M. (1997). *Solving the frame problem: a mathematical investigation of the common sense law of inertia.* MIT Press. 34

Sørensen, T. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons. *Kongelige Danske Videnskabernes Selskab*, 5(4):1–34. 86

Srivastava, M., Muntz, R., and Potkonjak, M. (2001). Smart kindergarten: sensor-based wireless networks for smart developmental problem-solving environments. In *Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 132–138. 20

Starner, T. (1995). Visual Recognition of American Sign Language Using Hidden Markov Models. 22

Steinhauer, H. and Chua, S. (2010). Utilising Temporal Information in Behaviour Recognition. In *AAAI Spring Symposium*, pages 54–59. 39

Stikic, M. and Schiele, B. (2009). Activity recognition from sparsely labeled data using multi-instance learning. In *Location and Context Awareness*, pages 156–173. Springer Berlin Heidelberg. 24

Sung, M., DeVaul, R., Jimenez, S., Gips, J., Pentland, A. (2004). Shiver motion and core body temperature classification for wearable soldier health monitoring systems. In *Wearable Computers, 2004. ISWC 2004. Eighth International Symposium on*, pages 192–193. IEEE. 18

Sutton, C., McCallum, A., and Rohanimanesh, K. (2007). Dynamic conditional random fields: Factorized probabilistic models for labeling and segmenting sequence data. *The Journal of Machine Learning Research*, 8:693–723. 25

Tapia, E., Intille, S., and Larson, K. (2004). Activity Recognition in the Home Using Simple and Ubiquitous Sensors. *Pervasive computing*, 3001:158–175. 20, 22, 98

Tapia, E. M., Choudhury, T., and Philipose, M. (2006). Building Reliable Activity Models Using Hierarchical Shrinkage and Mined Ontology. In *Proc. of PERVASIVE*, pages 17–32. 32, 36

Tran, S. and Davis, L. (2008). Event modeling and recognition using markov logic networks. In Forsyth, David A.; Torr, Philip H. S.; Zisserman, A., editor, *Computer Vision–ECCV 2008*, pages 610–623, Marseille. Springer Berlin Heidelberg. 39

Turaga, P. (2008). Machine recognition of human activities: A survey. *Circuits and Systems for Video Technology, IEEE Transactions on*, 18(11):1473–1488. 15

Vail, D., Veloso, M., and Lafferty, J. (2007). Conditional random fields for activity recognition. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 235. ACM. 25

van Kasteren, T. and Krose, B. (2007). Bayesian activity recognition in residence for elders. In *Proc. of the International Conference on Intelligent Environments*, pages 209–213. 22

Ward, A., Jones, A., and Hopper, A. (1997). A new location technique for the active office. *Personal Communications, IEEE*, 4(5):42–47. 16

Weinland, D., Ronfard, R., and Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241. 12, 15

Weiser, M. (1991). The computer for the twenty-first century. *Scientific American*, 265(3):94–104. 12

Wilson, D. and Atkeson, C. (2005). Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. In *Pervasive computing*, pages 62–79. Springer Berlin Heidelberg. 20, 23

Wobcke, W. (2002). Two logical theories of plan recognition. *Journal of Logic and Computation*, 12(3):371–412. 34

Wren, C. and Tapia, E. (2006). Toward scalable activity recognition for sensor networks. In *Location-and context-awareness*, pages 168–185. Springer Berlin Heidelberg. 12, 17, 20

Wu, J. and Osuntogun, A. (2007). A scalable approach to activity recognition based on object use. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE. 26, 31

Wyatt, D., Philipose, M., and Choudhury, T. (2005). Unsupervised activity recognition using automatically mined common sense. In *Proc. of AAAI*, pages 21–27. 31, 32

Yamada, N., Sakamoto, K., and Kunito, G. (2007). Applying ontology and probabilistic model to human activity recognition from surrounding things. *Information and Media Technologies*, 2(4):1286–1297. 36, 37

Ye, J., Stevenson, G., and Dobson, S. (2011). A top-level ontology for smart environments. *Pervasive and mobile computing*, 7(3):359–378. 37

Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13. 15, 16

This dissertation was finished writing in Bilbao on January 7^{th}, 2015