Intent Recognition of Autonomous Agents

Ali Azzam Naeem
M.Sc.

Thesis submitted for the degree of Doctor of Philosophy

NATIONAL UNIVERSITY OF IRELAND, CORK

School of Computer Science & IT

Insight Centre for Data Analytics

April 2024

Head of School: Professor Utz Roedig

Supervisors: Professor Kenneth N. Brown
Dr. Nic Wilson

Contents

List of Figures

List of Tables

I, Ali Azzam Naeem, certify that this thesis is my own work and has not been submitted for another degree at University College Cork or elsewhere.

Ali Azzam Naeem

Dedicated to my beloved parents
Arifa Mohamed & Adam Naeem
who have sacrificed so much
for my education.

Acknowledgements

I would like to thank my supervisor *Professor Kenneth Brown* for his support and guidance during my research career to date. His advice was invaluable not only in the preparation of this thesis but also in the course of my professional development. On several occasions, he went over and above the call of duty to help me succeed in my research and achieve my next career steps. I would also like to thank my secondary supervisor *Dr. Nic Wilson* for his guidance and support.

My heartfelt appreciation goes to *Dr. Patrick Tuite* for his comradery over our decade-long friendship. I met Pat when he taught me as a lecturer in my Masters. He has since become a trusted friend and confidante. I appreciate his genuine and sincere concern for my success while juggling the Ph.D. and full time work.

My wholehearted thanks to *Dr. Justin McGuinness* who has been a steadfast friend for over 10 years, always available to advise and help me complete the program. I consider myself lucky to call him my friend during the often turbulent journey this thesis represents.

A special mention and thanks to *Padraigh Jarvis*. Though we first met as colleagues, we have now become good friends. He has been an ever supportive voice, caring always for my well being and making the time to offer his support.

I would also like to thank *Martina Caruso* for her encouragement and friendship. She is a genuine person who has wholeheartedly wished the best for me in all the while I've known her. Her advice and her trust are deeply valued.

I am extremely blessed to have my sisters *Dr. Shanaz Naeem* and *Haifa Naeem* who always wish the best for me and help me reach my goals in any way they can. I am immensely grateful for my brothers-in-law *Dr. Patrick Ovando-Roche* and *Mohamed Misthoh*. They always drew from their own experience to give me the best support and advice moving forward.

This list would be incomplete without my nephew *Lihyan Mohamed Misthoh* and my niece *Ella Mohamed Misthoh*. Lihyan, though you are 8 years old, you bring us all so much joy and happiness. My memories with you are irreplaceable and treasured. And Ella, the newest member of our family. As I complete this thesis you have only been born a few months. I can already see the bright spark you have and I look forward to seeing you grow. I love you both.

A particularly special thank you goes to my uncle *Ismail Rasheed*. He is someone who invariably spoke with only my best interests at heart. His advice, encouragement and steadfast support play an integral role in this thesis becoming the finished product before you today.

Last, but not least, I can never thank my parents enough. They are incredibly supportive having the utmost faith in all of their children's abilities. They are a constant source of strength and mere words on a page will never suffice to give them the recognition and thanks they truly deserve.

Funding acknowledgement

The research presented in this thesis was financially supported by *Science Foundation Ireland* under grant number **12/RC/2289-P2** which is co-funded under the *European Regional Development Fund*.

The theory of probabilities is
nothing but common sense
reduced to calculus.

Pierre-Simon Laplace
(1749-1827)

CHAPTER 1

Introduction

1.1 Motivation

In recent years, increasingly powerful and capable artificial intelligence (AI) systems have become available. These systems now permeate every aspect of our lives and this trend is only expected to accelerate in the future.

A common trend observed in these new developments is an increasing level of autonomy exhibited by such intelligent systems. In fact, an inherent sense of agency is becoming more and more prevalent within them. These autonomous intelligent agents can perform a variety of ever more complex tasks and activities such as carrying out intelligent conversations, recommending courses of action or assisting the human in the goals they are trying to achieve.

In order for an autonomous support system to be truly intelligent, these systems must also be able to anticipate the activities that the human is about to perform. In this thesis, these activities are defined to be the intent of the person. Intent recognition, therefore, is the process of inferring the activities an autonomous agent or a person is about to perform in the near future from observations of their behaviour.

However, intent recognition can be a challenging task; one needs a clear definition of what intent is and even then an agent's intent is inherently latent, rarely known

explicitly (unless the agent self reports intent).

1.2 Thesis statement

In this thesis, an *activity* is a collection of atomic actions observed.

Intent is defined as a future activity the agent will undertake within a limited time window in order to achieve a goal it desires. Further discussion is provided in section 3.2.

The statement this thesis defends is:

It is possible to infer the intent of an autonomous agent using a hierarchical framework which combines machine learning models (for example, Bayesian inference models or gradient-boosted decision trees) with manifold learning techniques.

1.3 What was achieved?

This thesis contributes to the field of artificial intelligence/machine learning (AI/ML) and intent recognition by introducing a novel framework that infers the intent of an autonomous agent. The framework pushes the boundaries of the current knowledge in this area by combining machine learning models with manifold learning techniques in such a way that the framework is robust to the curse of dimensionality as well as incorporating mechanisms to account for varying quality of data sets across multiple domains. The proposed framework also narrows the design space for a practitioner who wishes to carry out intent recognition on a new data set by providing a structured approach to tackle the problem. The implementation examples demonstrate the framework's effectiveness across multiple domains.

The importance of intent recognition in autonomous agents has been recognized in previous research (e.g. [167, 112]). However, current methods have limitations in terms of scalability, robustness, and interpretability.

The proposed hierarchical framework addresses these limitations by combining machine learning models with manifold learning techniques, which enable the

representation and analysis of complex, high-dimensional data.

A lot of work published in the literature focuses on *activity* recognition. *Activities* correspond most closely to what are called *actions* in this thesis. In section 3.2 a clear distinction is proposed between *actions*, *activities* and *intent*. The latter two represent a higher level of abstraction; multiple actions performed as a group indicate an activity.

The focus of this thesis will be on *intent* prediction. As laid out in the thesis statement (section 1.2), intent is defined as an activity the agent will undertake within a limited time window in the immediate future.

This definition also differentiates intent recognition from plan recognition which is taking as input a sequence of actions performed by an agent and then inferring the intended sequence of actions which will be pursued by that agent to achieve their goal. Furthermore, based on the definition of the terms action, activity and intent specified in this thesis (section 3.2), intent recognition is also distinct from next activity prediction which is predicting the next observed actions without explicitly modelling the underlying activities or intents. More discussion of the differences between these terms follow in section 2.1.

1.4 The Implementation Examples

This section introduces three implementation examples through which this framework will be validated and made concrete. The three examples span different domains and leverage different machine learning models to achieve their goal.

Common between them, however, are few key features, namely:

- A specific definition of the three fundamental terms action, activity and intent within each context
- The use of clustering to establish an abstraction
- A hierarchical approach built on the clustering
- Two machine learning models operating on the two levels of the hierarchy to yield the final output: a prediction for the agents' intent

The machine learning models employed may be the same or different across the levels in the hierarchy. Chapters 4, 5 and 6 dive into each example in detail. However, a summary is offered here for the reader.

Activities of Daily Living Intent prediction of residents in a smart home carrying out activities of daily living.

Access controlled buildings Intent prediction of employees moving around an office building.

Moving vehicles Intent prediction of taxis driving around Porto, Portugal.

1.5 How is the claim validated?

To validate the proposed framework and prove the claim, this framework was applied and evaluated on the three implementation examples described in section 1.4.

In the first implementation example (activities of daily living, chapter 4), a common baseline was established with a simpler approach. Consequent experiments evaluating intent recognition are benchmarked against this common baseline.

For the second implementation example (access control, chapter 5), a proprietary data set was used from one of our industry partners. As such, there is no publicly available benchmark. Nevertheless, an ablation study was setup similar to the approach in the first implementation example where results from the approach proposed in the thesis was compared to a simpler approach as a baseline.

In the third implementation example (moving vehicles, chapter 6), the results obtained are benchmarked against results achieved by other users on Kaggle for the same data set (the data set used for this implementation example was originally made available as part of a Kaggle contest).

1.6 Who is going to use this?

This thesis presents very practical research which is applicable across a wide variety of domains. For example, the implementation example presented in chapter 4 (activities of daily living) has potential for great impact in the assisted living space. The framework is used on sensor data from a smart home to infer what the resident is doing at that moment and what the resident most likely intends to do in a short time horizon. Imagine, for instance, a patient with early onset Alzheimer's. A system employing this framework could preserve their independence for longer by learning their daily routines and giving them a prompt when the system detects they are getting confused. Such a system would also provide

peace of mind to their loved ones by quickly detecting and highlighting a change in the residents' behaviour.

Another practical example comes from the implementation example in chapter 6 where the framework is applied to predict the destination of moving vehicles. In this case the practical application would be predicting where the driver is going even if they do not explicitly set the GPS. This is particularly useful in cases of routine trips such as going to work where the driver may not set the navigation. However, if the intent recognition can still infer their destination, relevant information may be provided such as an accident on their regular route to work and proposal for a different route.

The activities of daily living and moving vehicles examples may also be combined together with this framework. An example of this would be where the smart home has detected the resident having breakfast and it is a weekday. The inference would then be they are now going to leave for work soon which may then trigger a message in their car for any incidents along their regular route to work.

The implementation example with access control has several applications in smart buildings and energy efficiency. By learning the routines of the employees and also their expected intent over the course of the day, heating for their personal offices and meeting rooms could be intelligently controlled (personal offices warmed up in advance and turned off when the employee leaves).

At a macro level, intent recognition across multiple employees could generate useful insight for the company on how their employees are utilising the facilities. The findings could then be used to inform more intelligent design of the workspaces to maximise productivity and collaboration.

The implementation example also carries security applications. For instance, if one considers a building with sensitive areas; anomalous behaviour by employees (even those with pre-approved access) could be intelligently detected. For example, a technician may have pre-approved access to the server room as part of their job. However, even in this case, the intent recognition framework could detect an anomalous pattern of access by the technician (at odd times etc.) which indicates an intention beyond completing their work duties. The key benefit is moving beyond preset access rules and providing a more precise detection of anomalous intent (should it exist) from those with access to such sensitive sections of the building.

1.7 Thesis contributions

This thesis makes the following contributions:

- Presents an intent recognition framework applicable to multiple domains.
- Formalizes the notion of *intent*, *activity* and *action* within the framework.
- Describes a method for applying manifold learning to reduce dimensionality and discover hidden structure in the data for clustering and developing a hierarchical abstraction.
- Describes how to build on the output embeddings calculated by manifold learning to train machine learning models which take advantage of the abstractions learnt from the data.
- Describes three distinct implementation examples which are candidates for the application of the framework proposed in this thesis.
- Demonstrates the implementation of the described framework on the three different examples from different domains.
- Evaluates the proposed framework on the different examples demonstrated. The evaluation is carried out by experiment and benchmarking is carried out where such benchmarks are available.
- Elements of this thesis were published in the following conference proceedings [120]:

  Ali Azzam Naeem and Kenneth N. Brown. *Inferring Destination from Mobility Data*, Proceedings of the 25th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2017), Dublin Institute of Technology, Dublin, Ireland, 7-8 December. CEUR Workshop Proceedings, Vol. 2086, pp. 153-165, 2017.

1.8 Thesis layout

The remainder of this thesis is laid out as follows; Chapter 2 provides an overview of the related work in intent recognition as well as a background in the machine learning techniques utilised in the framework. A review of intent recognition is carried out in general as well as through the specific lens of each implementation example highlighting the gap in the literature this work fills. Chapter 3 introduces the components of the intent recognition framework and the steps one needs to follow when applying the framework to a specific problem.

Chapter 4 is the first of three implementation examples. This chapter covers intent recognition framework as applied to the activities of daily living use case. Chapter 5 is the second implementation example; access control. Chapter 6 describes the third implementation example, moving vehicles. Finally, potential future directions for this work are described in chapter 7.

CHAPTER 2

Background & related work

The intent recognition framework presented in this thesis draws from several distinct areas. Therefore, in this chapter, the background and state of the art for each of these areas is reviewed in addition to the problem of intent recognition specifically. The thesis presents three implementation examples for the proposed framework in three different domains. Thus, intent recognition in the context of each domain is also discussed.

2.1 General concepts

The general scope of the term *"intent recognition"* is extremely broad and there is variation in the literature on how the term *intent* is interpreted. Intent recognition and other related concepts follow in section 2.1.1. Following that, intent recognition is reviewed specifically in the context of the implementation examples.

2.1.1 Plan, Activity and Intent recognition

Plan recognition, *activity* recognition and *intent* recognition all involve making inferences about other actors from observations of their behaviour. The term *behaviour* generally means the actor's interaction with their environment. In the following paragraphs, the definitions of these terms as found in the literature are

reviewed. These definitions are compared to how the same terms are interpreted in this thesis (a more detailed definition of the terms follow in section 3.2).

The plan recognition problem is defined as taking an input of a sequence of actions performed by an actor and then inferring the intended sequence of actions which will be pursued by that actor to achieve their goal [142]. Plan recognition is usually carried out with the help of a set of predefined recipes (called a *plan library*), which comprises the knowledge and action steps that can be performed by the agent being observed [106]. Several methods for plan recognition appear in the literature. The authors of [47] classify the most notable plan recognition methods into 4 categories; deductive [79], abductive [8], probabilistic [33] and case-based [84]. Plan recognition approaches may also be classified by whether the recognition process was *intended* [79] or *keyhole* [36]. If the agent being observed cooperates with the observer and conveys their intent willingly then the plan recognition process is said to be intended. In contrast, if there is no interaction between the observer agent and observed agent, then the process is known as keyhole plan recognition [7, 6]. Traditionally, the plan library has been handcrafted as is the case in [79]. However, this approach is feasible only in the cases where domain complexity is low. In a real world application of practical interest, constructing complete plan libraries by hand is often not possible. Machine Learning techniques have been employed in more recent research to automate the process of building the plan library [99, 84].

Plans are made up of states and actions that enable state transitions. In plan recognition architectures such as *PRODIGY* [168], a state of the world is represented by a conjunction of first order predicates. An action may only be executed if the pre-conditions associated with that action are met by the states of the world. Crucially, plan recognition involves organising the action sequence of the actor agent into a structure (known as the plan). The work presented in this thesis, therefore, is not related to plan recognition, as there is no inference on the action sequence for the autonomous agent whose behaviour is being observed in this thesis.

The activity recognition problem, in the literature, takes as input a sequence of noisy sensor inputs over time (such as inertial data) [94, 30]. These inputs directly capture some effect the actor or agent has on their environment. The targets of prediction for these studies include walking, crawling, standing, sitting etc. (example applications include first responders and emergency personnel) [190, 179, 141]. However, in this thesis, a more abstract definition of the term

activity is adopted. The chosen definition here follows the vein of [162] where an example of an activity would be cooking dinner or making breakfast.

In the taxonomy of this thesis, the more granular activities predicted in the previously referenced studies [190, 179, 141] such as walking, standing and sitting are called *actions*. A collection of actions form an activity in this thesis and finally intent is an activity with a forward time component (the reader is directed to figure 3.1 for a visual representation and invited to read section 3.2 for more details).

The taxonomy adopted in this thesis is inspired by the *Belief-Desire-Intention (BDI)* framework proposed by Michael E. Bratman [166]. His model proposes that human action is driven by a hierarchy of mental states that include beliefs, desires and intentions. The model starts with *desires* which are an individual's goals and objectives. These are influenced by their *beliefs*. In the BDI framework, beliefs form the agent's understanding and representation of the world, serving as the basis upon which desires and intentions are formed.

Intentions, then, are the most specific mental states in the hierarchy and refer to an individual's plan of action to achieve their desires based on their beliefs. Bratman argues that intentions are different from mere plans, as they involve commitment to carrying out the plan and a recognition of the potential obstacles that may arise.

Therefore according to the BDI model, intentions play a crucial role in coordinating an individual's behaviour towards achieving their desired goal. Bratman further argues that the formation of intentions involve a process of practical reasoning whereby an individual evaluates various possible courses of action and selects the most appropriate one based on their beliefs and desires.

Overall, the BDI model provides a framework for understanding how an individual's beliefs, desires and intentions interact to drive their actions and behaviours. The model has been used to propose a model for childlike reasoning known as *Children's Reasoning about Intentions, Beliefs and Behaviour (CRIBB)* [172]. It has also been used to develop a software model for intelligent programming agents [57].

In the taxonomy of this thesis, an activity is a collection of actions. Intent is an activity taking place in the near future as described in section 3.2. There is room for a higher level of abstraction (to be known as *goals*) as an extra level above actions, activities and intents defined in this thesis. An example of goals could

be spending time with family, making a living, etc..

In activity recognition, a hierarchical approach has been employed in the literature as well [161, 16], however the approach in this thesis differs from these related studies in a few key aspects. In [16], the abstractions at the higher level of the hierarchy are the beginning/end point of the activity and the type of activity. In [161], the authors utilise hierarchical modelling for the prediction of activities happening in the present moment rather than future intents and similar to [16], they also employ a hierarchical Hidden Markov Model. This is in contrast to the work presented in this thesis, which combines two distinct models linked by manifold learning to predict intent (activities taking place in the near future). A three-level hierarchical formulation of activity recognition have also been proposed in the literature [191] which begins with actions and then activities, however, the third level is modelled as complex interactions between humans that involve more than one person. A more detailed comparison of [161] and other existing work most closely related to this thesis follows in section 2.13.

There is precedent for using probabilistic and Bayesian methodologies in the problem of intent prediction. Joshua B. Tenenbaum's *Bayesian Theory of Mind* (BToM) [151] proposes that humans use probabilistic reasoning to infer the mental states of others, such as their beliefs, desires, and intentions. The human mind is described as a probabilistic inference engine that uses prior knowledge and observed input from the environment to make predictions about the world and infer mental states of other agents.

According to Tenenbaum, the Bayesian framework allows humans to make efficient and accurate inferences about others' mental states by combining prior beliefs and the likelihood of observed behaviors. This process of probabilistic inference can be used to explain a wide range of human behavior, including social cognition, language learning, and problem-solving.

Tenenbaum's theory also emphasizes the importance of hierarchical modeling in human cognition (which supports the decision made in this thesis to employ a hierarchical approach). The research further suggests that the mind actually operates at multiple levels of abstraction, with higher-level models providing context and constraints for lower-level models [82]. This hierarchical approach allows humans to reason about complex, abstract concepts like causality and agency, as well as learn and generalize from limited data; a similar benefit the present framework aims to derive and leverage.

Overall, Tenenbaum's Bayesian Theory of Mind provides a powerful framework for understanding human cognition, and has implications for a wide range of fields, including psychology, neuroscience, linguistics, and artificial intelligence. The theory of mind has also been used with Bayesian techniques for plan recognition expressing generative models of belief and desire dependent planning in terms of partially observable Markov Decision Processes (POMDPs) [149].

While the BDI framework for intent described above comes from the field of psychology, theories for intent have been put forward in other fields as well. For instance, within the field of cognitive science, a dynamic hierarchical model for intent has been proposed in [119]. Like others before them, the authors also argue that intentions consist of multiple levels of organization, ranging from higher-order intentions (e.g., long-term goals) to lower-level intentions (e.g., action plans).

Unique to their study is how the authors formulate and elaborate on the hierarchical structure of intentions; emphasizing the distinction between *distal* intentions (e.g., goals) and *proximal* intentions (e.g., action plans). They propose that distal intentions provide the overall purpose or aim of an action, while proximal intentions determine the specific means and strategies to achieve the distal intentions. This hierarchical organization enables individuals to flexibly adjust their actions based on contextual changes.

The dynamic nature of intentions is also emphasised, highlighting that intentions can be revised, abandoned, or created during the course of action. They propose a recursive mechanism by which intentions at different levels interact and influence each other, facilitating real-time adjustments to the ongoing action.

Finally, the authors integrate cognitive processes such as planning, decision-making, and self-monitoring into their hierarchical framework of intentions. They argue that these processes play a crucial role in the generation and execution of intentions. By incorporating cognitive processes, the model provides a more comprehensive account of how intentions are formed, updated, and executed.

Turning next to the field of philosophy, a seminal work by philosopher G. E. M. Anscombe published in 1957 explores the question "*What distinguishes actions which are intentional from those which are not?*" [13]. According to Anscombe, intentions are not mere mental states or thoughts about future actions but rather practical judgments or decisions that commit an agent to act in a certain way.

Intentions are characterized by their *direction of fit* meaning they align actions

with desires or goals, shaping one's behavior to bring about desired outcomes. As an example for direction of fit, Anscombe introduced two ways of interpreting a shopping list; one as a set of instructions for what to put in the basket, the other, as a detective spying on a shopper's movements, as a record of what has gone into the basket. In the first case the goal is to have the basket fit the list, however in the second, the list has to fit the basket. The concept is used to differentiate mental states like desires or intentions, which conform to the first model, from representations and beliefs, which conform to the second [153].

Intentional action for Anscombe are distinct from mere *bodily movements*. She emphasizes that intentional actions involve a thoughtful consideration of *why* such an action is being performed, a conscious decision made to act in a certain manner. This distinguishes intentional actions from reflexive or involuntary movements.

Another related concept in the literature is *next action prediction*. Next activity prediction involves the anticipation of future actions or behaviors based on observed historical data. This field is an intersection of time series analysis, pattern recognition, and predictive modeling, where algorithms are trained to forecast the next event (or events) in a sequence. The primary challenge in next activity prediction is dealing with the temporal dependencies and the context in which previous activities occurred.

To tackle this, various approaches have been proposed, including sequence modelling approaches, contextual analysis and feature engineering. Sequence modelling involve machine learning models specially designed to capture the temporal dynamics in the data. An example of a traditional sequence model is a Hidden Markov Model (HMM) and more recent deep learning based approaches to modelling sequences involve Recurrent Neural Networks (an idea introduced in a simpler form in [44]), Long Short-Term Memory (LSTM) networks [67], and Transformer models [165].

Incorporating contextual information (e.g., time of day, user location, or environmental conditions) enhances the prediction accuracy by providing additional insights into the circumstances surrounding each observed action. Feature Engineering also play a part in next action prediction. Meaningful features are extracted from raw data that encapsulate the essence of the actions and their progression. This might involve statistical features, embeddings, or manually crafted features that describe the historical patterns observed.

Some key differences between next action prediction and intent recognition (as

defined in this thesis) will now be described. In this thesis, intent recognition is viewed as the process of inferring the underlying intention indicated by an observed sequence of actions by autonomous agents. The thesis proposes a hierarchical framework that integrate various machine learning models with manifold learning and clustering to infer intent. The framework in this thesis presents a multi-layered approach factoring in activities in addition to the observed actions and their context to predict the intent.

In contrast, next action prediction, while it can employ hierarchical models, concentrate on directly predicting the next or future actions without explicitly modelling the underlying activities or intents. This distinction is based on the definitions laid out in the thesis for the terms action, activity and intent. As noted, these terms are used somewhat interchangeably in the literature with a distinction often not being made between some of these terms.

Another key difference between next action prediction and intent recognition arising from the nomenclature specified in the thesis is that intent is always a latent variable predicted from observed actions. Even for the ground truth, as the intent will never be directly observed, the agent must self-report its intent. Actions, however, will always be observed as they occur according to the framework presented in this thesis.

In terms of the relevant application domains also, a distinction may be made between intent recognition and action recognition. Intent recognition is particularly relevant in scenarios where understanding the underlying dynamics of the observed actions are crucial, such as in security, personalized assistance, or adaptive user interfaces. Next activity prediction is more suitable for demand forecasting, user behavior prediction in web services, anticipatory computing, etc..

Given the added complexity of intent recognition, however, it may also require richer datasets that include not just collections of observed actions but also contextual or explanatory data that can hint at the intentions behind the observed actions. Next activity prediction, as a simpler problem, often works with simpler sequence data, although incorporating context can enhance performance.

In summary, then, while both next action prediction and intent recognition aim to anticipate future states, intent recognition seeks to understand the deeper motivations behind actions, requiring a nuanced analysis of behavior and context. Next activity prediction, conversely, is more directly focused on accurately fore-

casting the immediate next event based on past sequences or observations of those events.

An example of LSTM models applied in the context of next action prediction is to predict human trajectories [4]. This research introduces a specialized model for human trajectory prediction in crowded spaces, emphasizing social interactions. The work introduces "*Social LSTM*", an extension of the traditional LSTM model to capture social interactions among individuals. This model uses a social pooling layer to allow LSTMs corresponding to different individuals to share information, thus learning social behaviors. The work presented in this thesis, however, provides a broader and more structured approach to intent recognition in general across multiple domains using a hierarchical framework.

The intent recognition framework presented in this thesis differs from the work presented in [4] in how the problem is modelled. The approach in the research paper always predicts the future trajectories of pedestrians in crowded environments which is at the same level of abstraction as the observed actions. Their approach involves predicting the future positions of individuals based on their past movements while accounting for the social and spatial dynamics that naturally occur in crowded settings, such as a busy street, shopping malls or airports. The Social LSTM model aims to foresee where a person will move in the near future by considering the person's past positions and the influence of surrounding people's movements. This task is viewed as a sequence generation problem, where the sequence of a person's past positions is used to generate the sequence of their future positions.

In the framework presented in this thesis, however, there is an explicit definition of what an action, activity and intent are for each use case. Activities and intents are often defined at a higher level of abstraction than the observed actions and may also be latent depending on the use case the framework presented in this thesis is being applied to. The framework then predicts intent using the observed actions. Overall, this is a more flexible formulation for intent recognition in general compared to the research paper focusing specifically on next action prediction.

Moving forward with the thesis, the background of intent recognition will now be reviewed specifically in the context of the three implementation examples presented in the upcoming chapters of this thesis.

2.1.2 Activities of daily living

An application of the intent recognition framework (presented in chapter 4) is prediction of intent in *activities of daily living* (ADLs). ADLs are routine activities that most people would be able to perform without assistance such as eating, bathing and toileting [78]. An important indicator of a person's ability to live and function independently is how well they can perform ADLs. An inability to perform ADLs result in the dependence of individuals on other care givers or assistive mechanical devices. Loss of ability to perform ADLs is a significant health problem for the elderly with major ramifications on their quality of life [192].

With advances in modern medicine and people living longer, there is an increased need for innovations which allow people to preserve their independence as much as possible. In the context of this thesis, intent is modelled as an activity the resident intends to carry out in the near future (this may include continuing to perform the current activity).

Smart home systems which monitor how well residents are performing ADLs increase the safety of people living on their own and allow elderly residents to be self reliant longer. This application is considered part of an emerging discipline known as *ambient intelligence* (AmI). AmI brings intelligence to our everyday environments and makes those environments sensitive to our actions [37]. This is achieved by advances in sensors and sensor networks, pervasive computing, and artificial intelligence. Ambient intelligence has emerged as a promising new paradigm particularly in the field of healthcare [140, 5, 2]. To realise the full potential of ambient intelligence systems, the predictions of such a system must be available in near real-time. Studies have proposed evaluating prediction of activities within 60 seconds of that activity commencing [173]. This time window is the benchmark utilised in this thesis.

In the literature, several methods are proposed for recognising the activities being performed by a person. One approach utilizes body-worn inertial sensors [190, 183, 75, 141, 96] or smartphone sensors [90, 5, 146, 11, 18, 137, 64, 91]. That approach often infers atomic actions such as falling, walking and running rather than the higher level activities such as cooking breakfast and showering which are the target for prediction in this thesis (chapter 4). Inferring atomic actions are useful for safety applications such as automatic fall detection. Several forms of wearable devices (smart watches, etc.) with alarm features have been deployed for this purpose [31, 170, 107]. These approaches suffer from a drawback in that they

rely on the sensor or smart device being actively maintained, kept fully charged and worn correctly by the user on their person to function effectively. This is not a reasonable assumption in every scenario of assisted living, particularly in an elderly resident use case.

An alternative approach is sensing the effects a user has on their environment. This can be achieved by a network of non obtrusive sensors placed in the user's environment. For example, sensors embedded in a smart home will generate readings while residents perform their daily routines [37]. These readings are collected and stored in a database. Knowledge extraction and reasoning from this data is achieved with machine learning models to learn a user's activity patterns [164, 101]. Such a system can detect changes in behaviour that indicate a deteriorating condition by recognizing when the user's activity deviates markedly from expected patterns. Approaches that recognise ADLs from sensors placed in the environment has been shown to perform well. One drawback of sensing the residents environment to infer their intent is the inherent ambiguity where the same sensor firings may correlate with multiple intentions on the part of the resident. For instance, have they opened the fridge because they are about to leave the house and want to check whether they have milk? Or are they grabbing some food? In the first case, the residents intent is to leave the house whereas in the second case their intent is to have a snack.

To address and handle this uncertainty, Bayesian models [163] are utilized in the literature (and in this thesis) for reasoning about the intent of the resident. It is noted that Support Vector Machines [37], Hidden Markov Models, Conditional Random Fields [162], deep convolutional neural networks [42, 75, 187, 63, 136], long short term memory (LSTM) recurrent neural networks [9, 1, 124] and other deep learning approaches [34] have been employed for activity recognition from time dependent sensor data. A newer trend incorporates *Attention* based mechanisms reflecting their growing popularity in the field of Machine Learning [86, 188].

In addition to sensor based recognition of activities, with the recent advances in computer vision, video based approaches are also becoming more popular with a greater focus on actions level predictions (although some studies use the term action and activity interchangeably). In the prediction of activities from video problem, the representation of space-time shapes received attention by the research community. In the earlier stages, global representations for images or silhouettes were popular [23]. The emergence of *space time interest points* (STIPs) [93] led to the adoption of more local representations that focused on informative interest

points. Another local representation method is *dense trajectories* [174, 175]. The method proposes densely sampling points and avoids extracting points frame by frame and concatenating them. Early spatiotemporal methods, in general, adopt a methodology of regarding the video as (x, y, t) 3D volumes [87, 177, 143].

More recently, video based activity recognition has focused on depth based representation with data captured from depth cameras. Previously research on human activity recognition focused on sequences captured by RGB cameras. Depth cameras were seen of limited utility at the time due to their high cost and complexity of operation [125]. However, that changed with the advent of low cost depth sensors such as Microsoft Kinect [145]. Methods developed with depth based representations revolved around the available depth maps and technology which enabled the recording of skeletal joint positions in real-time (such as the Kinect SDK adopting algorithms in [145] for example). Studies have now been published which attempts to fully exploit the potential of the Kinect sensor for the activity recognition problem including in combination with RGB video feeds [50]

In comparison to sensor-based approaches, video comes with greater privacy concerns. Therefore privacy preservation techniques are also included in video based approaches such as extracting silhouettes from the video frames [85], purposefully degrading the video feed and carrying out the inference task with extremely low temporal or spatial camera resolutions [39] and coded apertures [176].

Another non-invasive technique for human activity recognition is using WiFi signals [29]. The idea behind activity recognition using WiFi signals is that the human body will affect the signal by reflection and different activities will show different characteristics. The main activities in the classification task of this study were sitting, standing, walking, lying down, falling and human absence. As with [190, 141], these labels would be considered as *actions* rather than full-fledged *activities* in the terminology used in this thesis.

Another deep learning based approach as applied to activities of daily living is presented in [3]. The authors of this study present a unified deep learning model which integrates three stages; activity recognition, anomaly detection, and next activity prediction, to provide support for caregivers and medical teams. This research employs a sequence where each stage feeds into the next. Initially, a Deep Neural Network (DNN) is utilized for activity recognition based on various extracted features. Subsequently, an overcomplete-deep autoencoder (OCD-AE) identifies anomalies in the recognized activities. Finally, a Long Short-Term Memory (LSTM) algorithm predicts the next activity based on a sequence of recog-

nized and normal activities. The model's effectiveness is demonstrated using real smart home datasets. The elements of the approach focusing on activity recognition and anomaly detection are beyond the scope of this thesis. The methodology for next activity prediction, specifically the employment of LSTM algorithms is different to the hierarchical modelling approach proposed in this thesis. In addition, all three stages described in this paper rely on neural networks which require high volumes of clean data [59, 97].

All the approaches proposed rely on the availability of high quality labelled training data. However, such data is costly to procure and curate both in terms of time and effort. People also have an understandable apprehension to recording their daily activities due to privacy concerns. In addition, the target population which would most benefit from assisted living systems often may not be able to burden themselves with recording and logging their activities daily. An impact of not having large volumes of data is seen in the literature where applying techniques such as recurrent neural networks (generally shown to perform well given abundance of data) performs comparably to more classical machine learning methods [14]. Therefore, the sacrifices made in using deep learning networks (such as interpretability of the model and explainability of the results) are less justifiable in this scenario as there is no major gain in predictive accuracy.

To mitigate the lack of significant training data, simulators for human activity have been proposed [169]. The study in [14] takes a different approach and augments pre-existing data with anomalous activities to then investigate the detection of abnormal behaviours which deviate from previously observed patterns. In the literature, a particular interest is also given to the idea of transfer learning. In this scenario, data for a set of houses in which high quality data is available is used to hypothesise reasonable starting values in a new house for which there is little or no data. One study which addressed this problem used historical data to produce a prior distribution for a potential new house [160]. They also addressed the challenge of physical differences where no two houses had the same physical layout and sensor locations. The proposed solution was to learn "meta-features" between the two houses which were at a higher level of abstraction and a common frame of reference for both houses. For example, one house may have a sensor on the fridge and another may have a sensor above the microwave. Both sensors provide information on activities in the kitchen which is a common frame of reference between the two houses.

2.1.3 Access control

The second application of intent recognition (presented in chapter 5) is access control. Specifically, the intent of employees using their access credentials to gain entry to an office building is discussed.

In the literature, the problem of secure access in a building may be considered under the umbrella domain known as *frictionless access control*. This refers to a method of granting or denying access to resources without imposing unnecessary barriers or hindrances.

In the context of secure access, friction and security often go hand in hand. Consider a wide open doorway; the user experiences the least amount of friction with this setup, however it is also the least secure in that anyone can pass through. On the other hand, a multi-factor authentication system, while being extremely secure, also imposes greater friction on the user during authentication.

A lot of research has been done with the goal of maximising security while minimising friction. A relevant avenue for the application of the framework in this thesis is a type of continuous authentication known as *Behaviometrics* [62]. DARPA has run an *Active Authentication Program* [41] where several types of behavioural biometrics are investigated in the context of recognising and verifying the identity of a person. The framework presented in this thesis may be used to verify that the user's intent is consistent with expected behaviour patterns and continue authenticating their access to a secured system.

In addition, within the context of *energy control*, intent recognition in smart buildings is a critical research area that focuses on understanding and interpreting human intentions and behavior. It plays a significant role in enabling efficient energy management and optimizing resource allocation in smart building systems.

Smart buildings are equipped with various sensors and actuators that collect data about occupants, environmental conditions, and energy consumption patterns. Intent recognition techniques analyse this data to infer users' intentions and preferences related to energy usage, occupancy, and comfort. By accurately recognising and predicting user intents, smart buildings can dynamically adjust settings and automate energy control processes to optimize energy efficiency and user comfort [100]. It should be noted that [100] is not carrying out intent recognition of autonomous agents as in the work presented in this thesis. Rather they focus on inferring which purpose a room or some other physical space in a building is being used for (either presentation where someone is giving a talk, "discussion"

where there is a meeting happening or "none" where the room is not occupied at all). The difference, then with the work presented in this thesis and [100] is that [100] makes inferences about a particular physical space whereas this thesis infers intent of an autonomous agent across multiple physical locations.

Other studies found in the literature does note that predicting and reasoning about occupant presence and movement over time is extremely pertinent to the accurate performance of smart building control systems [123], a finding which supports the decision made in this thesis in implementation example II (access control) to model the behaviour of the employees rather than the fixed readers (for more details, the reader is directed to section 5.4).

In summary, intent recognition in smart buildings for efficient energy control is an active area of research. The integration of intent recognition techniques enables smart buildings to adapt and optimize energy-related processes based on occupants' intentions and preferences. Several studies have demonstrated the effectiveness of intent recognition in achieving significant energy savings, improved occupant comfort, and enhanced overall building performance.

2.1.4 Vehicle mobility

The third application of intent recognition (presented in chapter 6) is the intent of drivers in a vehicle.The vehicles in the data set are taxis and the intent of the driver is viewed as the destination of a given trip. Looking at similar work in the literature, the *Predestination* algorithm [89] uses a Bayesian Inference model combined with geographic data such as types of land cover. This allows the model to pre-emptively rule out, for example, areas of water as possible destinations by using the Bayesian models prior belief probabilities; areas of water are assigned zero probability in the model's priors which means they are never considered as potential destinations in the posterior distribution.

The algorithm implements an open world model which attempts to capture the possibility of a vehicle going to destinations which have not yet been observed in the training data. It does this by assuming that there is a certain likelihood of a driver going to a destination which is near some other destination that they have visited before (for instance, they may stop off at the supermarket or restaurant close to their home or place of work. The algorithm also incorporates the idea of driving efficiency. It supposes that the driver will take a "reasonable" route to their destination and thus assigns higher preference to those destinations which can be reached within reasonable travel.

The *PROCAB* algorithm [193] improves on Predestination by using Markov Decision Processes to learn preferences on the part of drivers (for instance some road types are more preferable and hence have higher utility). This allows the algorithm to learn contextual information which increases performance. PROCAB is shown to perform similarly to Predestination when a large portion of the trip is observed and do better than Predestination earlier in the trip.

It is often the case that in approaches for destination prediction where the destination probability is calculated conditioned on certain observed features of the trip, there exists a data sparsity problem. Specifically the method is only feasible if there exists in the data a historical trajectory with the exact same characteristics as the current observed trip. One solution to this is to include extra information such as road maps, route planning, etc. as seen in the Predestination algorithm with land cover [89].

An alternative method which also addresses the problem of destination prediction is *Sub-Trajectory Synthesis* (SubSyn) [181]. The SubSyn algorithm addresses the data sparsity problem by decomposing historical trajectories into sub-trajectories, which are then synthesized into new trajectories for prediction purposes. This method significantly expands the number of query trajectories that can have predicted destinations.

For this implementation example of the proposed intent recognition framework, the method of handling data sparsity differs from both studies [89, 181] cited above. In the work presented in this thesis, a geometric model is designed and employed to address data sparsity. This approach does not require any external pieces of information which is an advantage the proposed method has over [89] which leveraged extra information such as road maps and types of land cover.

Furthermore, the geometric prior model presented in the thesis estimates transition probabilities for specific portions of the trips even where the same transition is not observed in any example in the training data, which is the advantage the method proposed in the thesis has over [181] which is still limited by the training data in terms of new trips that can be synthesised from (in this comparison, the sub-trips extracted from longer whole trips in [181] are considered similar to the transitions between discrete locations which is how the method proposed in the thesis represents a trip). The specific details of the geometric prior used in this implementation example are described in chapter 6.

The *T-Drive* algorithm [184, 185] builds a landmark graph in which a node repre-

sents a road travelled often by taxis. The route prediction is made in a hierarchical fashion similar to the concept presented in this paper. Initially, a rough route is predicted using the landmark graph. In the second step, this rough route is refined to give a final prediction. The key difference to the work presented in this thesis however is that the T-drive algorithm's goal is to provide the most efficient driving directions i.e., finding the fastest route between two points using historical taxi trajectory data. The work presented in this thesis predicts the taxi driver's intent, modelled as the final transition into a drop off location for the trip. This study and the work presented in this thesis, then, differ in the target variable for a which prediction is made and the problem that each aims to solve.

The problem of destination prediction with taxis has also been attempted with neural networks [40] and regression based approaches [92]. A common thread in these approaches is some discretisation of the locations whether by applying a uniform grid or clustering.

2.1.5 Summary

The section discussed the concept of intent recognition in various fields, including psychology, cognitive science, philosophy, and applied contexts relevant to this thesis namely, activities of daily living (ADLs) in smart home systems, access control and moving vehicles. It explores different definitions and interpretations of terms such as plan recognition, activity recognition, and intent recognition.

Relevant frameworks and models for intent recognition like the Belief-Desire-Intention (BDI) framework and the Bayesian Theory of Mind were also explored. It highlights the hierarchical nature of intentions and the role of cognitive processes in forming and executing intentions.

A comparison of the work presented in this thesis to closest pieces of work published in the literature and how the thesis work fits in the context of the reviewed frameworks follow in section 2.13.

2.2 Manifold learning

2.2.1 Overview

In recent years, manifold learning techniques have gained significant attention in the field of machine learning. Traditional machine learning algorithms assume that the data lies on a flat Euclidean space, which may not accurately represent

the underlying structure of high-dimensional data. Manifold learning techniques aim to uncover and represent the intrinsic nonlinear structure or manifold on which the data resides. This chapter provides an overview of manifold learning techniques and highlights their benefits in improving the performance of machine learning models.

Manifold learning refers to a set of techniques that map high-dimensional data into a lower-dimensional space while preserving the underlying structure of the data. By reducing the dimensionality of the data, manifold learning can facilitate visualization, clustering, classification, and other machine learning tasks. The core idea is to model the data as if it lies on a smooth manifold embedded in the high-dimensional space.

2.2.2 Benefits of Manifold Learning Techniques

Improved Data Visualisation Manifold learning techniques enable the visualisation of high-dimensional data in lower-dimensional spaces. By projecting the data onto a lower-dimensional manifold, complex structures and patterns can be better understood and visually interpreted.

Nonlinear Dimensionality Reduction Traditional linear dimensionality reduction techniques, such as Principal Component Analysis (PCA), assume a linear relationship between the variables. Manifold learning methods, on the other hand, can capture nonlinear relationships, making them more suitable for datasets with intricate structures. Manifold learning algorithms, in general, aim to discover the underlying structure of high-dimensional data and represent it in a lower-dimensional space.

Noise Reduction and Outlier Detection Manifold learning techniques can help identify and remove noisy or outlier data points. By focusing on the intrinsic structure of the data, these methods can distinguish between the true underlying patterns and noise, leading to improved model performance.

Improved Classification and Clustering Manifold learning methods can enhance the performance of classification and clustering algorithms. By mapping the data onto a lower-dimensional manifold, these techniques can reveal clusters or decision boundaries that are not apparent in the original high-dimensional space, resulting in better separation and classification accuracy.

2.2.3 Popular Manifold Learning Methods

t-Distributed Stochastic Neighbour Embedding (t-SNE) is a powerful manifold learning technique that maps high-dimensional data into a low dimensional embedding space, emphasizing the preservation of local similarities. It is particularly effective in revealing clusters and structure in the data [156, 157, 158, 159].

t-SNE converts the distances between data points in the high dimensional space into conditional probabilities that represent similarities. The key steps involved in this process are:

- *Similarity calculation in higher dimensional space* - For each pair of points x_i and x_j, t-SNE calculates the conditional probability $P(j|i)$ that x_i would pick x_j as its neighbour if those neighbours were picked in proportion to their probability density around a Gaussian centered at x_i.
- *Similarity calculation in lower dimensional space* - t-SNE also defines a similar probability distribution $Q(j|i)$ between each pair of points x_i and x_j in the low dimensional embedding space, however, it now replaces the Gaussian with the Student's t-distribution giving the technique its name t-SNE. The t-distribution has fatter tails, allowing some distances to be less faithfully preserved in the embedding.
- *Optimisation* - The goal now is to make the two distributions defined above as similar as possible. To achieve this goal, the Kullback-Leibler (KL) divergence between the two distributions is minimised using gradient descent. This process ensures that similar objects are modelled by nearby points and dissimilar objects are modelled by distant points in the lower dimensional space.

Locally Linear Embedding (LLE) is a non-linear dimensionality reduction technique that approximates the local geometry of the data. It seeks to preserve the local relationships between neighbouring points by reconstructing each point as a linear combination of its neighbours [139]. The approach can be summarised in three key steps:

- *Neighbourhood preservation* - For every data point x_i in the high dimensional input space, LLE finds k nearest neighbours based on Euclidean distance or some other metric.

- *Linear reconstruction weights* - LLE then computes weights w_{ij} that best linearly reconstruct each data point in the high dimensional input space from its neighbours minimising the reconstruction error.
- *Embedding into lower dimensional space* - The weights w_{ij} are fixed and LLE now seeks a set of points in the low dimensional embedding space that best preserves these local linear relationships. This is achieved by minimising a cost function that measures the discrepancy between the linear reconstructions in the high dimensional input space and the low dimensional embeddings.

Isomap is a widely used manifold learning algorithm that preserves geodesic distances between points. It constructs a neighbourhood graph based on pairwise distances and then computes a low-dimensional embedding that preserves the geodesic distances as much as possible [150].

Laplacian Eigenmaps is a method that constructs a graph representation of the data and then computes a low-dimensional embedding by solving an eigenvalue problem. It emphasizes the preservation of local distances between neighbouring points [19].

Uniform Manifold Approximation and Projection (UMAP) is a competing technique to t-SNE which is constructed from a theoretical framework based on Riemannian geometry and algebraic topology. The authors contend that the technique is on par with t-SNE on visualisation quality while preserving more of the global structure present in the original data [110].

These are just a few examples of popular manifold learning methods, and there are many other techniques available in the literature. The choice of a specific method depends on the characteristics of the data and the requirements of the task at hand.

2.2.4 Example Application of Manifold Learning

Consider an example from the *natural language processing* (NLP) domain where manifold learning is used to find words that are similar to "*petrichor*" based on their semantic and contextual similarities. Petrichor refers to the pleasant earthy smell that often accompanies the first rain after a dry spell. Some examples of words that might be found near petrichor in the learned manifold include "earthy", "rain", "aroma" and "refreshing". All of these words share similar connotation

to petrichor referring to qualities linked to soil and smells. Manifold learning techniques can be used to embed sentences containing these words and ideas into a numeric representation. The exact placement of words in the manifold will depend on the specific algorithm used, the data and the context of the word relationships.

To apply manifold learning to find similar words to "petrichor", a dataset that contains a wide range of words and their associated features, such as word embeddings or contextual information are utilised. Word embeddings are dense vector representations that capture semantic relationships between words. The words in the dataset can be transformed into embeddings using popular methods for this purpose include *Word2Vec* [115], *GloVe* [128], or *fastText* [114]. These embeddings capture semantic relationships between words based on the distributional hypothesis, which suggests that words that appear in similar contexts share similar meanings.

Once word embeddings are known, pairwise similarities between them are computed to create a similarity matrix. Similarity metrics like cosine similarity can be used for this purpose. The similarity matrix captures the relationship between each word pair based on their embeddings.

At this point manifold learning algorithms, such as t-Distributed Stochastic Neighbour Embedding (t-SNE), Uniform Manifold Approximation and Projection (UMAP), or Locally Linear Embedding (LLE) are employed to project the high-dimensional similarity matrix into a lower-dimensional latent space. These algorithms aim to preserve the local and global structures of the data. The resulting latent space is a lower-dimensional representation of the original data, where each point corresponds to a word. The distances between points in this space reflect their similarities based on the initial word embeddings. In this case, words with similar connotations to "petrichor", such as "earthy", "rain", "aroma", and "refreshing," would be located close to each other in the latent space.

As illustrated, manifold learning techniques provide a robust and powerful method to analyse and interpret high dimensional data with minimal loss of context. For human interpretation, the latent space can also be visualised using scatter plots and other relevant visualisation techniques. Each word is represented as a point in the plot, and their proximity reflects their semantic similarity. By analyzing the plot, it is possible to identify clusters or groups of words that share similar characteristics, helping to uncover the semantic relationships between words with natural language processing and manifold learning techniques.

The ability of manifold learning to find latent structure in the data and isolate subgroups is utilised in the thesis framework to establish levels of the hierarchical model (more details in chapter 3).

2.3 Clustering

2.3.1 Overview

Clustering is a fundamental technique in machine learning that aims to partition data into groups or clusters based on their similarities. It is an unsupervised learning technique that enables the discovery of inherent structures in data without the need for labeled examples. By grouping similar data points together, clustering algorithms help uncover hidden patterns and relationships.

Clustering plays a crucial role in various domains, including data analysis, pattern recognition, image processing, and recommendation systems. This section provides an overview of clustering techniques, their benefits, popular clustering methods and evaluation methods to assess the quality of clustering.

2.3.2 Benefits of Clustering

One key area of benefit is in *data exploration* where clustering aids in identifying subgroups (or clusters) within a data set enabling researchers and practitioners to explore the data in a more granular manner. These benefits extend into *pattern recognition* where clustering algorithms reveal latent patterns and structures within the data.

Clustering is also of benefit in the *anomaly detection* domain. By separating normal data points from outliers, clustering techniques can effectively identify anomalies or novel patterns in the data set. Another area of application for clustering is *data compression* where clustering can help reduce the dimensionality of large data sets. This is achieved by representing them with a smaller number of representative clusters. Clustering is also applicable in *recommendation systems* where clustering can be utilised to group similar users or items which leads to more personalised recommendations.

2.3.3 Popular Clustering Methods

This section provides an overview of some popular clustering algorithms with references to their original works. The choice of clustering method depends on the specific requirements of the problem at hand. Some popular clustering methods will now be described.

2.3.3.1 KMeans Clustering

KMeans [105] is a centroid-based clustering algorithm that aims to partition a set of observations into a predefined number of clusters k, where each observation belongs to the cluster with the nearest mean. The algorithm operates through an iterative process that minimizes the *within-cluster sum of squares* (WCSS), which is equivalent to maximizing the between-cluster sum of squares.

Lloyd's algorithm [102] is the most widely recognized and commonly used version of the KMeans clustering algorithm. The algorithm details the iterative refinement technique for optimising the centroids and cluster assignments. The steps to apply the algorithm may be summarised as follows:

- *Initialisation* - Choose k initial centroids, typically as random observations from the data set.
- *Assignment step* - Assign each observation to the cluster with the nearest centroid, usually using Euclidean distance as the measure.
- *Update step* - Recalculate the centroids of each cluster as the mean of all points assigned to that cluster.
- *Iteration* - Repeat the assignment and update steps until convergence, which occurs when the assignments no longer change or the change in the value of the cost function (such as the total within-cluster variance) falls below a threshold.

2.3.3.2 DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [46] is an algorithm which groups data points based on their density and is particularly effective at discovering clusters of arbitrary shapes. The algorithm's strategy of identifying clusters based on the density of the data points differentiates it from centroid based clustering techniques.

The core concepts of DBSCAN include:

- *Core Points* - A point is considered a core point if it has more than a specified number of points ($MinPts$) within a given radius (ϵ).
- *Border Points* - Points that are not core points but are in the neighbourhood of a core point.
- *Noise* - Points that are neither core nor border points.

The DBSCAN algorithm proceeds by:

- *Identifying Core Points* - For each point in the dataset, if it has at least $MinPts$ within ϵ distance, it is marked as a core point.
- *Forming Clusters* - Starting from a core point, the algorithm recursively includes all directly reachable points within ϵ distance, forming a cluster. This process is repeated for each unvisited core point, generating distinct clusters.
- *Assigning Border and Noise Points* - Border points are assigned to one of the clusters of their associated core points, whereas noise points are not included in any cluster.

2.3.3.3 Mean Shift

Mean Shift [54] is a non-parametric clustering algorithm that seeks to find dense areas of data points by updating candidates for centroids to be the mean of the points within a given region. Unlike KMeans, Mean Shift does not require specifying the number of clusters in advance.

The key parameters of the Mean Shift algorithm are:

- *Bandwidth* - The radius of the circular (in two dimensional cases) or spherical (in higher dimensions) window that is used to compute the mean shift. It determines the size of the region to search for points to include in the mean calculation.
- *Kernel* - The kernel function determines the weight of nearby points for re-estimation of the mean. Common choices include the Gaussian kernel and the flat kernel. The kernel function influences how the distance from the centroid to each point within the window affects the calculation of the new centroid.

- *Stopping Criteria* - The stopping criteria for the Mean Shift algorithm determine when the iterative process of updating centroids will stop. This can be based on a maximum number of iterations or a convergence threshold, where the centroids do not move significantly between iterations.

Mean Shift is adept at handling clusters of various shapes and sizes. Its main challenge lies in selecting the appropriate bandwidth size, as a bandwidth that is too small can lead to over-segmentation, whereas a bandwidth too large may merge distinct clusters.

The Mean Shift algorithm involves:

- *Initialization* - Place a circle (in two dimensions) or a sphere (in higher dimensions) at each data point's location. The radius of this circle/sphere is a bandwidth parameter that dictates the size of the region to search for data points.
- *Updating Centroids* - Compute the mean of all points within each circle/sphere and move the circle/sphere center to that mean.
- *Convergence* - Repeat the updating step until the centers no longer move significantly, indicating that the clusters have been found.

2.3.4 Other Clustering Methods

Some other clustering algorithms are summarised below:

Gaussian Mixture Models (GMMs): GMMs [133, 111] assume that the data points are generated from a mixture of Gaussian distributions and use the Expectation-Maximization (EM) algorithm [116] to estimate the parameters.

Hierarchical Clustering: Agglomerative and divisive hierarchical clustering methods [71] construct a tree-like hierarchy of clusters to represent the data's structure.

Spectral Clustering: This method [122] leverages the eigenvectors of the similarity matrix to perform clustering, allowing the discovery of non-linearly separable clusters.

2.3.5 Clustering Evaluation

To evaluate the quality of clustering results, various metrics and indices have been developed. These measures assess the coherence and separation of clusters, enabling quantitative assessment. Silhouette scores will be reviewed first as this measure is used to evaluate the quality of clustering for the implementation examples in this thesis. For completeness, an overview of other popular clustering evaluation metrics follow.

2.3.5.1 Silhouette Scores

Silhouette analysis [138] is used to analyse the separation distance between the clusters produced by a clustering algorithm. The analysis involves calculating a number known as the mean Silhouette coefficient of all the samples. This coefficient has a range between -1 and +1 and indicates how close each point in one cluster is to points in the neighbouring clusters. Silhouette coefficients near +1 indicate that the sample is far away from the neighbouring clusters and this is desirable. A value of 0 indicates that the sample is on or very close to the decision boundary between 2 neighbouring clusters. Negative values are an indication that the sample has been assigned to the wrong cluster. Therefore, higher values for the silhouette score indicate better-defined and well-separated clusters.

The silhouette coefficient is calculated using the mean intra-cluster distance a and the mean nearest cluster distance b (the distance between the sample and the nearest cluster that the sample is not a part of). The formula for silhouette coefficient S then is:

$$S = \frac{b - a}{max(a, b)}$$

To determine the best value for k at each level, a range of cluster labels which made sense for each specific implementation was specified. The clustering algorithm was then run for each value of k which falls into the range. The mean silhouette coefficient was then calculated for each resultant clustering. The value for k which yielded the highest mean silhouette coefficient at each level was chosen as the final choice for the number of clusters.

2.3.6 Summary

This section provided an overview of clustering techniques, their benefits, listed some popular clustering methods, and evaluation methods to assess the quality of clustering results. Understanding these concepts is essential for the subsequent chapters of this Ph.D. thesis, where clustering algorithms will be applied and evaluated in specific implementation examples.

2.4 Evaluating a machine learning classifier

This section provides a summary of popular methods used for evaluating the performance of machine learning classifiers. Machine learning classifiers play a pivotal role in various domains, ranging from image recognition to natural language processing. Evaluating the performance of these classifiers is vital for understanding their efficacy and enabling comparisons between different models. The evaluation methods discussed in this section encompass both traditional evaluation measures as well as further techniques that have gained prominence in recent years.

2.4.1 Standard evaluation metrics

The performance of the framework in Implementation Example I (activities of daily living, chapter 4) and Implementation Example II (access control, chapter 5) are evaluated using the metrics of *precision*, *recall* and *f-score* [104]. These were chosen as the definition of intent in these examples yield a classification problem. The chosen metrics are described here [1].

Precision and *recall* are two numbers which together are used to evaluate the performance of classification models. Precision is the percentage of actual answers which were correct whereas recall is the percentage of possible answers provided by a classification algorithm which were correct [152]. The precision metric attempts to answer the question *"what proportion of positive identifications are actually correct?"*. Recall, on the other hand, answers the question *"what proportion of actual positives are identified correctly?"*. A perfect precision score of 1 is achieved by a model that produces no false positives whereas a perfect recall score of 1 is achieved by a model which produces no false negatives.

[1] Implementation Example III (moving vehicles) involve a regression model and is therefore evaluated with distance based metrics which are described in chapter 6.

Both precision and recall measures may be calculated with the aid of a confusion matrix, an example of which is shown in table 2.1. A confusion matrix provides a comprehensive breakdown of classification results, displaying true positive, true negative, false positive, and false negative values. It serves as a basis for deriving evaluation metrics such as accuracy, precision, recall, and F1-score.

Before discussing the interpretation of the confusion matrix, however, some nomenclature is required and will now be introduced:

True positives (TP_i) The number of times class or label i was the true label and inferred correctly.

Total true (TT_i) The number of times class or label i was the true label occurring in the test data.

Total inferred (TI_i) The number of times class or label i was inferred by the model.

The diagonal of the confusion matrix in table 2.1 are the true positives (TP_i). The sum of each row gives the total true labels (TT_i). The sum of the columns are the total inferred labels (TI_i). The off-diagonal entries ϵ_{ij} represent mis-classifications where true label i has been mis-classified as incorrect label j by the model.

Table 2.1: Confusion matrix

		inferred		
true	**1**	**2**	**3**	
1	TP_1	ϵ_{12}	ϵ_{13}	TT_1
2	ϵ_{21}	TP_2	ϵ_{23}	TT_2
3	ϵ_{31}	ϵ_{32}	TP_3	TT_3
	TI_1	TI_2	TI_3	*total*

In all nomenclatures, the subscripts i and j represents a label or class in the data. During the evaluation procedure, precision and recall are calculated separately for each class using equations 2.1 and 2.2 respectively.

To fully evaluate the performance of a classification model, however, one must examine both precision and recall as these metrics are often in tension; improving one deteriorates the other. Therefore, a further metric which takes into account

both precision and recall is required. One such metric is the F-Measure or F-Score [134] which is defined as the harmonic mean between precision and recall. Normally as recall goes up, precision tends to go down and vice versa [49]. The F-Score provides a way of combining recall and precision to get a single measure which falls somewhere between recall and precision.

The calculated values for precision and recall are used to calculate a separate F-Measure for each class using equation 2.3 [73]. The use of these metrics are indicated by the fact that this data suffers from the problem of unbalanced classes, the ramifications of which will be described.

In the following equations, N represents the total number of labels or classes to be predicted.

$$precision_i = \frac{TP_i}{TI_i} \qquad i \in \{1 \dots N\} \tag{2.1}$$

$$recall_i = \frac{TP_i}{TT_i} \qquad i \in \{1 \dots N\} \tag{2.2}$$

$$f - measure_i = \frac{2 \cdot precision_i \cdot recall_i}{precision_i + recall_i} \qquad i \in \{1 \dots N\} \tag{2.3}$$

At this stage, the metrics are calculated for each class separately. Given that in this example, we are dealing with imbalanced classes, there are two separate strategies presented for combining these class-wise metrics. The first strategy is standard averaging which is referred to as the *macro* average. The macro average treats each class label as equally important. These macro averages for each metric are calculated using equations 2.4, 2.5 and 2.6.

$$precision_{macro} = \frac{1}{N} \sum_{i=1}^{N} precision_i \tag{2.4}$$

$$recall_{macro} = \frac{1}{N} \sum_{i=1}^{N} recall_i \tag{2.5}$$

$$f - measure_{macro} = \frac{1}{N} \sum_{i=1}^{N} f - measure_i \tag{2.6}$$

A more representative metric in most cases for calculating the overall classification performance is the *weighted* averaging scheme. Under this strategy, the class wise metrics are averaged by weighting each class metric with the support of that class

(the number of true instances of that class in the data). The weighted average is a more accurate reflection of user experience as the better the model is at predicting activities that happen more frequently, the greater impact this has on the experience of the users.

The weighted average may also be used in cases where certain outcomes, though infrequent, are still important and a false negative has greater consequence. For example, consider a patient with cancer being incorrectly reported as cancer-free. This outcome has major ramifications including losing the opportunity to treat and mitigate the disease in its early stages.

A final metric included in the results is *accuracy*. The accuracy is interpreted as the proportion of correctly classified data points over all the data points. Under this definition of accuracy, with imbalanced classes in the training data, the more frequent classes can artificially inflate the classifier's performance when those classes are predicted accurately. For example if class label A makes up 90% of the training data, then a one rule "model" that only ever predicts class A will still achieve 90% accuracy. Therefore, while simple and intuitive, accuracy may not be suitable for imbalanced data sets.

The accuracy metric is still included in the results of this thesis for completeness. However it is, by itself, not a truly representative metric when one has imbalanced classes as in this data. The formula used for calculating accuracy is presented in equation 2.7.

$$accuracy = \frac{\sum_{i=1}^{N} TP_i}{total} \tag{2.7}$$

2.4.2 Other evaluation techniques

Some other evaluation techniques for machine learning classifiers will now be described.

Cross-validation is a resampling technique that estimates the model's performance by dividing the data into training and validation subsets. Common variants include k-fold cross-validation and stratified cross-validation, ensuring representative distribution of classes.

Bootstrap resampling involves repeatedly drawing random samples with replacement from the original data set. It allows for estimating various performance

measures, such as confidence intervals and bias, without making assumptions about the underlying data distribution [43].

2.4.3 Summary

Evaluating the performance of machine learning classifiers is an essential aspect of research and model development. This section highlighted popular evaluation methods, including traditional measures like accuracy, precision, recall, and F1-score, as well as other techniques such as ROC curves, precision-recall curves, cross-validation, and bootstrap resampling.

2.5 XGBoost

2.5.1 Overview

XGBoost (eXtreme Gradient Boosting) is a powerful machine learning algorithm that belongs to the family of gradient boosting methods. It was introduced by Chen and Guestrin in [35] and has gained significant popularity in both academia and industry due to its exceptional performance and versatility. XGBoost is particularly well-suited for solving a wide range of machine learning problems, including classification, regression, and ranking tasks.

2.5.2 Benefits of XGBoost

Gradient boosting: XGBoost employs an ensemble learning technique called gradient boosting, which builds a strong predictive model by combining weak individual models, known as decision trees. It iteratively improves the model's performance by minimizing the loss function using gradient descent optimization. The term *gradient boosting* was originally coined by Friedman in [53].

Regularisation: XGBoost integrates regularisation techniques to prevent overfitting. It offers *L1* and *L2* regularisation, also known as the *Lasso* and *Ridge* penalties, respectively [72]. Regularization helps control the complexity of the model, improves generalization, and reduces the risk of overfitting.

Efficiency: XGBoost is highly efficient and can handle large-scale datasets with millions of instances and features. It leverages parallel processing, col-

umn block loading, and compressed sparse column techniques to accelerate training and prediction speed. Additionally, it provides an early stopping mechanism to terminate the training process if no improvement is observed, further enhancing efficiency.

Feature importance: XGBoost enables measurement of feature importance, which helps identify the most relevant features contributing to the predictive performance. This information can aid in feature selection, feature engineering, and understanding the underlying patterns in the data.

Flexibility: XGBoost offers various hyperparameters that can be tuned to optimize model performance for specific problems. It supports custom loss functions, allowing users to incorporate domain-specific objectives. Furthermore, XGBoost can be seamlessly integrated with other machine learning libraries and frameworks, such as *Scikit-learn* [127] and *Apache Spark* [186].

2.5.3 Other Popular Ensemble Learning Methods

Following are just a few examples of popular ensemble learning methods:

Random Forest is an ensemble method that constructs multiple decision trees and combines their predictions through voting or averaging. It is known for its robustness against overfitting and ability to handle high-dimensional data [28].

AdaBoost (Adaptive Boosting) is a boosting algorithm that assigns weights to the training instances and trains weak learners iteratively. It focuses on misclassified instances to improve their performance in subsequent iterations, thereby constructing a strong ensemble model [52].

Gradient Boosting Machines (GBM) is a family of boosting algorithms that iteratively trains weak learners to minimize a loss function. It gradually builds an ensemble by adding models that correct the mistakes of the previous ones. In fact, XGBoost is an enhanced implementation of GBM [53] which is parallelised and optimised for training speed compared to the standard GBM algorithm. Another distributed high-performance framework similar to XGBoost is known as *LightGBM* developed by Microsoft [80]. In contrast to the horizontal or level-wise growth employed by XGBoost, LightGBM carries out leaf-wise or vertical growth. While this approach can result in more loss reduction (and therefore higher accuracy), it

can also result in greater overfitting on the training data which would need to be handled by parameter running. Empirically, the XGBoost algorithm is capable of building more robust models than LightGBM.

Bagging (Bootstrap Aggregating) is an ensemble method that creates multiple bootstrap samples from the training data and trains individual models on each sample. The final prediction is obtained by averaging or voting across the predictions of these models [27].

Voting Classifiers combine predictions from multiple base models and make the final prediction based on majority voting or weighted voting. It can be implemented with different approaches like hard voting, soft voting, and stacking [180].

2.6 Bayesian inference

Bayesian Inference is named after Thomas Bayes, an 18th-century Presbyterian minister who devised a mathematical rule for explaining how you should change your existing beliefs in the light of new evidence. It is a method for statistical inference which models one's degree of belief in a given hypothesis. As new evidence is observed that either supports or contradicts this hypothesis the degree of belief is iteratively updated to reflect the new evidence.

Mathematically, Bayes rule [88] is written as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2.8}$$

More specifically, we may apply this rule to learn the parameters θ of a model with observed data $\mathcal{D}$:

$$P(\theta|\mathcal{D}) = \frac{P(\mathcal{D}|\theta)P(\theta)}{P(\mathcal{D})} \tag{2.9}$$

The term $P(\mathcal{D}|\theta)$ is known as the *likelihood* as it is the probability of observing the current data under the proposed parameters. $P(\theta)$ is the *prior* which is some previous knowledge we may already have about those parameters. This includes any assumptions made as to the distributions those parameters may take. $P(\mathcal{D})$ is often computationally expensive to calculate as it involves summing over the parameters of the model.

$$P(\mathcal{D}) = \sum_{\theta} P(\mathcal{D}|\theta)P(\theta) \tag{2.10}$$

The counting or frequency based approach for exact inference suffers from the curse of dimensionality. When one conditions on several different features as is the case in most models of practical interest, calculating exact likelihoods becomes a multidimensional sum with the latent variables associated with the extra features all being additional dimensions that need to be marginalized out. Given observed data $\mathcal{D}$, hidden or latent variables x and parameters θ:

$$P(\mathcal{D}|\theta) = \sum_{x} P(\mathcal{D}|x,\theta)P(x|\theta) \tag{2.11}$$

In such models with latent variables included $P(\mathcal{D})$ becomes:

$$P(\mathcal{D}) = \sum_{x,\theta} P(\mathcal{D}|x,\theta)P(x|\theta)P(\theta) \tag{2.12}$$

However, $P(\mathcal{D})$ is also a normalization term and constant regardless of the model parameters hence when comparing different parameters (for instance in Maximum Likelihood Estimation), it is possible to do so based on proportionality using a likelihood and prior known only up to the normalization constant. This idea is used in probabilistic inference both in the approximate case based on Markov Chain Monte Carlo sampling and the exact case based on probability tables.

The exact inference approach involves calculating the likelihood and prior probabilities by counting the number of times a given case occurs in the data set and the scaling it by the length of the data set. For instance to calculate the prior for one particular case:

$$P(A = a) = \frac{N_a}{N} \tag{2.13}$$

N_a is the number of times a occurs in the data set. N is the number of instances in the data set.

For exact calculation of the likelihood, an example of the counting approach is as follows:

$$P(B = b|A = a) = \frac{N_{ab}}{N_a} \tag{2.14}$$

N_{ab} is the number of times a and b occurs in the data set. As before, N_a is the number of times a occurs in the data set.

2.6.1 Bayesian Inference Techniques

This section provides an overview of exact and approximate Bayesian inference techniques, highlighting their advantages, disadvantages, and popular applications. Bayesian inference is a powerful framework for statistical analysis that allows for the incorporation of prior knowledge and updating beliefs based on observed data.

Exact inference techniques, such as *variable elimination* [189], *conjugate priors* and *exact posterior sampling*, provide closed-form solutions when the posterior distribution is analytically tractable. However, in complex models or when analytical solutions are intractable, approximate inference methods, including *Markov Chain Monte Carlo* (MCMC) [135] and *variational inference* (VI) [25], offer efficient and scalable solutions.

Bayesian inference is a cornerstone of modern statistical analysis, enabling researchers to make probabilistic statements about unknown quantities of interest based on observed data. It involves updating prior beliefs using Bayes' theorem and obtaining the posterior distribution of the parameters. Exact inference methods aim to find the posterior analytically, while approximate methods seek to approximate it using computational techniques.

2.6.1.1 Exact Bayesian Inference Techniques

Variable Elimination The *variable elimination* algorithm is a powerful method to efficiently compute marginal probabilities or conditional probabilities of variables in a Bayesian network. The algorithm aims to eliminate variables from the graphical model by iteratively "summing out" or marginalizing them. As an illustrated example, consider a chain Bayesian Network (i.e.: a probability) of the form:

$$p(x_1, \ldots, x_n) = p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}).$$

The goal is to compute a marginal probability $p(x_n)$. A naive approach would be to sum all the probabilities across k^{n-1} assignments to $x_1, ..., x_{n-1}$:

$$p(x_n) = \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1, \ldots, x_n).$$

However, a more efficient approach would be leveraging the factorisation of the probability distribution; rewrite the sum in a way that pushes in certain variables deeper into the product:

$$\begin{aligned} p(x_n) &= \sum_{x_1} \cdots \sum_{x_{n-1}} p(x_1) \prod_{i=2}^{n} p(x_i \mid x_{i-1}) \\ &= \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_1} p(x_2 \mid x_1) p(x_1) \end{aligned}$$

Sum the inner terms first, starting from x_1 and ending with x_{n-1}. More concretely, start by computing an intermediary *factor* by summing out x_1:

$$\tau(x_2) = \sum_{x_1} p(x_2 \mid x_1) p(x_1)$$

The resulting factor $\tau(x_2)$ may be interpreted as a table of k values with one entry for each assignment to x_2. Next, rewrite the marginal probability τ as:

$$p(x_n) = \sum_{x_{n-1}} p(x_n \mid x_{n-1}) \sum_{x_{n-2}} p(x_{n-1} \mid x_{n-2}) \cdots \sum_{x_2} p(x_3 \mid x_2) \tau(x_2).$$

This has the same form as the initial expression except that the summation is over one fewer variable. Another factor may now be computed:

$$\tau(x_3) = \sum_{x_2} p(x_3 \mid x_2) \tau(x_2)$$

The process may thus be repeated only x_n is left which was the goal. In the process above, a variable is eliminated in each iteration which gives the algorithm its name. The initial naive solution takes $O(k^n)$ time whereas the inference procedure above takes $O(nk^2)$ time.

Conjugate Priors Conjugate priors are chosen such that the posterior distribution belongs to the same parametric family as the prior [55]. This allows for closed-form solutions, simplifying the inference process. For example, the *beta* distribution is a conjugate prior for the *binomial* distribution.

It should be noted, however, that conjugate priors are limited in their flexibility and may not be available for complex models.

2.6.1.2 Approximate Bayesian Inference Techniques

For most probabilistic models of practical interest, exact inference is intractable, and some form of approximation is required [21]. Some popular sampling methods are now described.

Markov Chain Monte Carlo (MCMC) methods, such as the popular *Metropolis-Hastings* algorithm [113] and its variants like the *Hamiltonian Monte Carlo* [121], provide a way to sample from the posterior distribution. MCMC techniques are widely used due to their flexibility and ability to handle complex models. However, they can be computationally expensive for large datasets or high-dimensional parameter spaces. Modern variations of these algorithms have emerged which aim to improve convergence rates and increase computational efficiency (such as *Gibbs sampling* [56], *adaptive rejection Metropolis* sampling [58] and the *No-U-Turn Sampler* (NUTS) [68]).

Variational Inference (VI) approximates the posterior distribution by framing inference as an optimization problem. It involves finding a distribution from a predefined family that minimizes the Kullback-Leibler divergence to the true posterior. VI is computationally efficient and scalable to large datasets but may introduce biases in the approximation.

2.6.1.3 Exact Inference Techniques pros and cons

The key benefit of exact inference techniques are that they provide analytically tractable solutions (for simple models). Where conjugate priors exist, they are also very efficient computationally in their calculation. An exact technique guarantees calculation of the true posterior distribution.

On the other hand, an exact inference technique is only feasible on a limited set of models which are not very complicated in their state space. In high-dimensional settings, they are extremely demanding computationally. Furthermore, a conjugate prior may not exist for all models and analytic solutions may not be available for more complex models.

2.6.1.4 Approximate Inference Techniques pros and cons

In contrast to exact techniques, approximate inference techniques have tremendous flexibility to handle complex models and data structures. They can scale to

large data sets and high dimensional parameter spaces. They provide reasonable approximations even when the posterior is analytically intractable.

The caveat, however, is that approximations may introduce biases or errors. Furthermore, while approximate inference techniques are computationally feasible for most models of practical interest, given a large enough dataset and complex enough model, the computational cost can still be high. These methods also often require judicious tuning of parameters to work effectively.

2.6.2 Summary

Exact and approximate Bayesian inference techniques offer distinct advantages and disadvantages. Exact methods provide closed-form solutions but are limited to specific model classes, while approximate methods offer scalability and flexibility but introduce approximation errors. The choice of inference technique depends on the complexity of the model, available computational resources, and the desired trade-off between accuracy and efficiency.

2.7 Probabilistic graphical models

This section provides an overview of probabilistic graphical machine learning models, with a particular focus on Bayesian networks. It discusses the fundamental concepts, advantages, and applications of Bayesian networks in various domains. Additionally, it highlights some other popular applications of probabilistic graphical models.

Probabilistic graphical models (PGMs) offer a powerful framework for representing and reasoning about uncertainty in machine learning. PGMs employ graph structures to model complex dependencies among variables, while incorporating probabilistic reasoning to make predictions or inferences. This section focuses on one type of PGM, namely Bayesian networks, which provide an intuitive and efficient way to model uncertainty and causality.

2.7.1 Bayesian Networks

Bayesian networks, also known as belief networks or causal probabilistic networks, are graphical models that encode probabilistic relationships between variables using *directed acyclic graphs* (DAGs) [88]. The nodes in the graph represent random variables, and the edges indicate conditional dependencies between them. The

graph structure is guided by domain knowledge or learned from data, enabling a compact representation of complex probabilistic relationships.

Two different approaches have been used to construct Bayesian networks; data-based approach and knowledge-based approach. The data-based approaches use conditional independence semantics of Bayes nets to induce models from data [65, 66]. The knowledge-based approach uses causal knowledge of domain experts in constructing Bayesian networks [95]. This approach is especially useful in situations where availability of data is scarce and therefore domain knowledge becomes crucial if not the only viable option. Elicitation of qualitative knowledge from domain experts is critical in constructing Bayesian networks because humans find it easier to handle qualitative over quantitative data [126].

2.7.2 Bayesian Network Inference

One of the primary benefits of Bayesian networks is their ability to perform efficient probabilistic inference. By exploiting conditional independence assumptions encoded in the graph structure, Bayesian networks can compute the posterior probabilities of unobserved variables given observed evidence. This inference process, known as probabilistic reasoning, allows for predictions, explanations, and decision-making in uncertain environments.

2.7.3 Learning Bayesian Networks

Bayesian networks can be learned from data using various approaches, including parameter estimation and structure learning. Parameter estimation involves estimating the conditional probability distributions of variables given their parents, either from data or using prior knowledge. Structure learning aims to infer the graph structure itself, capturing the causal relationships among variables. Learning methods range from simple score-based algorithms to more advanced techniques such as constraint-based and hybrid methods [74].

2.7.4 Benefits of Bayesian Networks

Bayesian networks offer several advantages that contribute to their widespread use in various domains:

Uncertainty Modeling: Bayesian networks provide a natural way to model and reason about uncertainty. By explicitly representing and propagating

uncertainty through the graph structure, these models can handle incomplete or noisy data, as well as make robust predictions and decisions in the face of uncertainty.

Causality and Explanation: Bayesian networks excel in capturing causal relationships between variables. The directed edges in the graph represent causal dependencies, enabling the understanding and explanation of how changes in one variable affect others. This causal modeling capability is particularly valuable in decision support systems and domains where understanding causal relationships is crucial.

Integration of Prior Knowledge: Bayesian networks allow the incorporation of prior knowledge and domain expertise into the model. By specifying prior probabilities and conditional dependencies, experts can encode their knowledge, improving the model's performance and interpretability.

2.7.5 Applications of Bayesian Networks

Bayesian networks find applications in diverse fields. For example, in the field of healthcare and medicine, they are used for disease diagnosis, treatment planning, personalised medicine and in clinical decision support systems [147]. Within bio-informatics, Bayesian Networks are used to power gene expression analysis, protein structure prediction, and biological network modeling [148].

They are also used in the field of finance and risk management for tasks such as credit scoring, risk assessment and fraud detection [103, 117]. Within environmental science, they are used for climate modelling, environmental monitoring and ecological risk assessments [77].

In the field of robotics and autonomous systems, Bayesian networks find applications in sensor fusion, perception, planning and decision-making under uncertainty [130, 45]. The rapidly growing field of natural language processing also utilises Bayesian networks for information extraction, sentiment analysis, text classification and machine translation [61, 154].

2.7.6 Summary

Probabilistic graphical machine learning models, particularly Bayesian networks, provide a flexible framework for modeling uncertainty, causality, and decision-making. Their ability to represent complex dependencies and perform efficient inference makes them valuable tools across various domains.

2.8 Latent Variable models

Graphical models can be used to define high-dimensional joint probability distributions. A dependence between two variables is modelled by adding an edge between them in the graph. An alternative approach is to assume that the observed variables are correlated because they arise from a hidden common "cause". Model with hidden variables are also known as *latent variable models* or LVMs [118].

Latent variable models and generative probabilistic processes have been successfully employed in the context of intent recognition, providing a powerful framework for capturing the hidden structure and uncertainty inherent in this task.

Latent variable models allow for the representation of complex relationships between observed input data (such as user queries or actions) and unobserved variables (such as underlying intent categories). By positing the existence of latent variables, these models aim to uncover the hidden intentions driving the observed behavior. This enables a more nuanced understanding of user interactions and facilitates accurate intent recognition.

Latent variable models offer a powerful framework for intent recognition. By leveraging the ability to model hidden variables and uncertainty, these approaches enable a deeper understanding of user intentions.

2.9 Generative probabilistic processes

Generative probabilistic processes play a central role in latent variable models for intent recognition. They define a joint probability distribution over observed input data and latent variables, capturing the underlying data generation process. By sampling from this generative process, new instances of input data can be synthesized, providing a way to generate realistic examples of user queries or actions associated with specific intents. Some examples of popular generative models include Generative Adversarial Networks (GANs) [60] and Hidden Markov Models (HMMs) [131],

In the context of intent recognition, the iterative process of *"Build, Compute, Critique, Repeat"* is particularly relevant [24]. In the *"Build"* step, researchers construct a generative model that captures the dependencies between input data and latent intents. This model encodes assumptions about the underlying intent

recognition problem and provides a probabilistic representation of the relationships between observed and unobserved variables.

The *"Compute"* step involves performing inference in the latent variable model to estimate the posterior distribution of the intents given the observed input data. Bayesian inference techniques, such as variational inference or Markov chain Monte Carlo methods, can be used to approximate this posterior distribution. By computing the posterior probabilities of intents, the model can make probabilistic predictions about the underlying intentions behind user actions or queries.

In the *"Critique"* step, researchers evaluate the model's performance. This can involve comparing the model's predictions to ground truth labels or human-labeled data, calculating evaluation metrics such as accuracy or precision-recall, or conducting user studies to assess the system's effectiveness. Based on these evaluations, researchers can identify areas for improvement and refine the model accordingly.

The iterative nature of the *"Repeat"* step allows for continuous refinement and enhancement of the latent variable model for intent recognition. Researchers can modify the model's architecture, incorporate additional features or contextual information, or explore alternative parameterisations to improve the accuracy and robustness of the intent recognition system.

Generative probabilistic processes as built using the iterative "*Build, Compute, Critique, Repeat*" process allows for continuous improvement and refinement of the model, enhancing the accuracy and effectiveness of intent recognition systems in practical applications.

2.10 Discrete Time Markov Chains

This section provides a comprehensive overview of *Discrete Time Markov Chains* (DTMCs), *Hidden Markov Models* (HMMs), and *Markov Chains* in general. The fundamental concepts, properties, and applications of these probabilistic models are explained.

Markov Chains are mathematical models used to describe the evolution of systems with probabilistic transitions between states [131]. They have found extensive applications in diverse fields such as physics, economics, computer science, biology, and more. Markov Chains provide a framework to analyze and predict the behavior of systems based on their current state and the probabilities of transitioning

to other states.

2.10.1 Markov Chains

A Markov Chain is a stochastic model [81] that consists of a set of states and a transition matrix describing the probabilities of transitioning from one state to another [69]. These models possess the *Markov* property, which states that the future behavior of the system depends solely on its current state and is independent of its past states.

2.10.2 Discrete Time Markov Chains

Discrete Time Markov Chains (DTMCs) are Markov Chains in which the time variable progresses in discrete steps. At each time step, the system transitions from one state to another according to the transition probabilities specified by the transition matrix. DTMCs are widely used for modeling and analyzing a wide range of systems, including biological processes, queuing systems, finance, and more. They provide a powerful tool to study steady-state behavior, absorption probabilities, hitting times, and various other properties of the modeled systems.

2.10.3 Hidden Markov Models

Hidden Markov Models (HMMs) are extensions of Markov Chains that incorporate hidden or unobserved states. In HMMs, the system's state is not directly observable, but it emits a sequence of observable symbols or outputs. HMMs find extensive applications in speech recognition, natural language processing, bioinformatics, and many other fields. The state sequence of a discrete time finite state Markov process may be estimated by the Viterbi algorithm [171, 48].

2.10.4 Structure of HMMs

An HMM consists of a set of hidden states, a set of observable symbols or outputs, and transition and emission probabilities. The transition probabilities define the probabilities of transitioning between hidden states, while the emission probabilities determine the probabilities of emitting observable symbols from each hidden state.

2.10.5 Capabilities of HMMs

HMMs offer several beneficial capabilities in modeling and analysis:

Sequence Prediction: HMMs excel at predicting or decoding the most likely sequence of hidden states given a sequence of observed symbols [70]. This makes them particularly useful in speech recognition, where the goal is to convert an audio signal into a corresponding text sequence.

Clustering and Classification: HMMs can be used for clustering or classifying sequences based on their hidden structure [182]. This property is advantageous in applications such as genome analysis, where DNA sequences need to be grouped into different categories.

Fault Diagnosis/Anomaly Detection: HMMs can be applied to fault diagnosis and anomaly detection in complex systems [15]. By modeling the normal behavior of a system using an HMM, deviations from the expected behavior can be identified and flagged as potential faults.

2.10.6 Limitations of Markov Chains

While Markov Chains, DTMCs, and HMMs offer numerous advantages, they also have certain limitations. One key limitation is the assumption of memorylessness, where the future state depends only on the current state. This assumption may not hold in all real-world scenarios. Additionally, the size and complexity of the state space can pose challenges in terms of computational efficiency and model training.

2.10.7 Applications of Markov Chains

Markov Chains, including DTMCs and HMMs, have extensive applications in various domains. In economics and finance, Markov Chains are used to model economic systems, stock markets, and financial processes [32]. They help analyze the behavior of asset prices, forecast market trends, and optimize investment strategies.

In biology and genetics, Markov Chains are employed to model biological processes [51]. Applications include gene regulatory networks, protein folding, and epidemiology. HMMs are widely used for DNA sequence analysis, protein structure prediction, and identifying functional regions in genomes.

In the field of Natural Language Processing, HMMs and other probabilistic models based on Markov Chains are utilized for speech recognition, machine translation, sentiment analysis, and part-of-speech tagging [76]. Within the Operations

Research (OR) domain, Markov Chains play a crucial role in modeling and optimizing various manufacturing processes, supply chains, and queuing systems [26].

2.10.8 Summary

Markov Chains, DTMCs, and HMMs provide valuable tools for modeling, analyzing, and predicting systems with probabilistic behavior. Their wide-ranging applications across diverse fields highlight their significance and effectiveness. By leveraging the properties and benefits of these models, researchers can gain insights, make predictions, and optimize complex systems.

2.11 Hierarchical models

Hierarchical models in the context of machine learning represent a class of statistical models that incorporate a nested structure, allowing for modelling of relationships at different levels of granularity within a data set. This hierarchical approach is particularly valuable in scenarios where data exhibit a natural hierarchy or grouping. One prevalent concept within hierarchical models is the idea of *pooling*, a technique that leverages information across different levels of the hierarchy to improve model robustness and predictive accuracy [109].

Pooling, in the context of hierarchical models, involves combining information from various levels of the hierarchy to make more informed and generalised predictions. In the literature, hierarchical models are often discussed in the context of Bayesian statistics, where the incorporation of prior knowledge plays a crucial role [55].

Fully pooled models treat all data points equally and assume a uniform distribution across all levels of the hierarchy. These models essentially ignore any potential variations within groups, treating the entire data set as a homogeneous entity. While computationally efficient, fully pooled models may oversimplify the underlying structure of the data, leading to sub-optimal performance when dealing with diverse and heterogeneous data sets.

Partially pooled models, on the other hand, allow for variability within groups by incorporating group-specific parameters alongside shared parameters across the entire data set. This flexibility permits the model to capture both common patterns and individual idiosyncrasies within different groups, resulting in improved

predictive accuracy. Partial pooling strikes a balance between the two extremes of complete pooling and no-pooling models, thus offering a more nuanced representation of the underlying data structure.

The incorporation of hierarchical models and pooling techniques has found applications in various domains, such as social sciences, epidemiology, and ecology. For example, in educational research, hierarchical modelling has been applied to the problem of accurately assessing student growth [144]. More recently, hierarchical modelling has also been applied in *graph neural network* (GNN) research [129].

In conclusion, hierarchical models with pooling mechanisms provide a powerful framework for capturing complex patterns within data sets. While fully pooled models may oversimplify relationships, partially pooled models strike a balance between capturing shared patterns and accommodating individual variations. The choice between these models depends on the nature of the data and the research question at hand. Furthermore, the Bayesian perspective inherent in many hierarchical models adds an additional layer of interpretability and uncertainty quantification.

2.12 Research gap analysis

A few papers particularly relevant to this thesis will now be highlighted and compared to the framework presented in this thesis. Firstly the paper "*Hierarchical Activity Recognition Using Automatically Clustered Actions*" by Van Kasteren et al. [161] employs the same data set as the activities of daily living implementation example in this thesis (chapter 4). The paper proposes a method for activity recognition based on hierarchical modeling and automatically clustered actions. The goal is to improve the accuracy and efficiency of recognizing complex activities from sensor data. The authors begin by discussing the limitations of traditional activity recognition approaches, which often rely on predefined actions and lack the ability to handle complex, hierarchical activities. To address this, they propose a hierarchical approach that models activities at different levels of granularity. The key idea is to automatically cluster low-level actions based on their temporal patterns. This clustering process helps to discover common sub-activities and allows for the recognition of higher-level activities composed of these sub-activities. The key differences between this paper and the work presented in this thesis is in the clustering scheme and target for prediction. The

work in [161] clusters actions whereas in this thesis, specifically chapter 4, the clustering is carried out on the activities to establish the hierarchy using the same data set. The authors in [161] also predict the current activity taking place in the present moment whereas in this thesis intent is predicted (activity taking place within a limited future time window).

Another paper similar to this work is "*Bayesian intention inference for trajectory prediction with an unknown goal destination*" by Graeme Best and Robert Fitch [20]. This work addresses the problem of predicting the future trajectory of an intelligent agent. The authors consider the case where the agent has a higher-level intention to move to a goal region of their environment, which is only known to the agent itself. The agent is assumed to take the shortest path to the goal region with some uncertainty while avoiding static obstacles. The proposed method is limited to moving agents which overlaps with the third implementation example in this thesis (chapter 6). Crucially, however, the work presented in this thesis differs in that the target is intent defined as the final destination of the agent as opposed to the trajectory or path taken by the agent.

A third similar paper is "*Probabilistic programs for inferring the goals of autonomous agents*" by Marco Cusumano-Towner et al. [38]. The work explores a novel approach to inferring the goals of autonomous agents using probabilistic programming. The paper notes the challenges associated with inferring the goals of autonomous agents, such as limited observability and uncertainty in their decision-making processes. To address these limitations, the authors introduce a probabilistic programming framework for inferring goals. Probabilistic programming allows for flexible modeling of uncertainty and complex dependencies, making it suitable for capturing the intricacies of autonomous agent behavior. The proposed framework employs a generative model that describes the relationships between the agent's goals, actions, and observed behavior. By leveraging probabilistic programming, the framework can infer the goals of the agent by conditioning on observed data. The approach in [38] differs from the hierarchical framework presented in this thesis in a few ways; the work presented in this thesis utilise hierarchical modelling and also utilize manifold learning techniques providing more flexibility and expressive power in modelling complex data. The authors of [38] do not demonstrate a hierarchical model specifically rather they focus more on highlighting the advantages of their class of probabilistic programs in capturing the nuances of agent behavior and demonstrate the effectiveness of the proposed framework through experiments on multiple problems of which agent goal prediction is one.

In reviewing the literature, it was noted that a general purpose framework which aims to standardise the problem of intent recognition was missing. A lot of the studies which looked at similar problems used terms such action, activity and intent interchangeably, often with definitions very specific to the domain in which the study was being conducted. A widely accepted and generalisable definition for the terms action, activity and intent were not available. This is the research gap which the work presented in this thesis aims to fill.

A standardised definition is provided for actions, activity and intent in this thesis. Furthermore, a clear set of steps is laid out for a machine learning practitioner to apply this framework to carry out intent recognition on any data set with discrete observations (or a discretised set of continuous observations) from an autonomous agent. Presently, a practitioner who wishes to conduct intent recognition on a new dataset is faced with a huge decision space in terms of data pre-processing, model selection, training and evaluation. The proposed framework narrows the scope for the practitioner and provides a recipe to tackle the problem in a structured manner. The framework accounts for a commonly encountered problem in data sets arising from most practical applications which is the curse of dimensionality by proposing the use of manifold learning techniques. Finally, the framework is evaluated by experiment on three implementation examples across very different domains to establish its versatility and performance. The proposed combination of manifold learning techniques and hierarchical machine learning models applied to the problem of intent recognition is a unique contribution of the work presented in this thesis.

2.13 Conclusion

In summary, this chapter discussed the concept of intent recognition and provided an overview of different interpretations of the term *intent* in the literature. It covered plan recognition, activity recognition, and intent recognition, highlighting their distinctions. The section also introduced the taxonomy and hierarchical framework adopted in this thesis, inspired by the *Belief-Desire-Intention* (BDI) model. The BDI model proposes that human action is driven by beliefs, desires, and intentions, with intentions playing a crucial role in coordinating behavior towards desired goals.

The section further explored the *Bayesian Theory of Mind*, which suggests that humans use probabilistic reasoning to infer the mental states of others, including

their beliefs, desires, and intentions. The role of hierarchical modeling is emphasized in Tenenbaum's framework with higher-level models providing context and constraints for lower-level models of human cognition. In this sense, the hierarchical nature of the framework presented in this thesis aligns with Tenenbaum's framework. In contrast to the Bayesian Theory of Mind however, the framework presented in this thesis predicts intent specifically which is an action with a forward time component. Examples appear in the literature where a single type of machine learning model (such as a partially observable Markov decision process) was employed in an application of the Bayesian Theory of Mind to infer an agent's belief and desire distribution [17], however not intent as defined in this thesis. Furthermore, the work in this thesis encompasses and combines different types of machine learning models and techniques, a fact which is demonstrated across the implementation examples (with both boosting and Bayesian models being utilised in conjunction with manifold learning).

In addition to the Bayesian Theory of Mind, other theories and models of intent from fields such as cognitive science and philosophy are also discussed. A dynamic hierarchical model of intent was discussed in [119] which proposes a recursive mechanism where intentions at different levels interact and influence each other. The dynamic hierarchical model proposed by the authors provides a theoretical foundation for understanding the hierarchical structure and dynamics of intentions, which are essential for intent recognition. The dynamic aspect of the model is crucial in intent recognition. By incorporating the dynamic nature of intentions, the hierarchical framework for intent recognition can capture real-time adjustments in intentions and enable more accurate recognition of ongoing actions. Moreover, the integration of cognitive processes in the dynamic hierarchical model aligns with the cognitive aspects of intent recognition. Recognizing intentions require understanding the decision-making, planning, and self-monitoring processes underlying human actions. By considering these cognitive processes within the hierarchical framework, the intent recognition system can better model and interpret human behavior. The work presented in this thesis differs from [119] in some key aspects. Firstly, as a theoretical framework, there is no element of learning from data in [119]. Furthermore, while the hierarchical formulation is similar, the interpretation of the hierarchies in [119] as proximal and distal differs from the thesis framework. In this thesis, the levels of the hierarchy are interpreted as differing levels of abstraction. The time element comes in when intent is defined in the thesis as activities undertaken within a limited time window. This is on top of the abstraction hierarchy where activities are a

higher level of abstraction to atomic actions (more details in chapter 3).

G. E. M. Anscombe's framework on intention was also noted in this chapter. Anscombe's work emphasizes the distinction between intentional actions and bodily movements, highlighting that intentions involve thoughtful consideration and conscious decision-making. Anscombe's work relates to the framework presented in this thesis in differentiating actions and intents with intents being described as movements carried out in service of a particular goal. The work in this thesis also recognises intents as an abstraction above atomic intents. Similar to [119], Anscombe's work does not present a learning from data aspect while the work presented in this thesis describes and demonstrates how to instantiate the framework on three different practical examples and train the models within the framework.

The latter sections of this chapter provided an overview of relevant machine learning concepts and techniques to understand and interpret the work presented in the following chapters of this thesis.

In conclusion, it was noted in the research gap section of the literature review (section 2.12) that no clear framework exists which aims to generalise the problem of intent recognition. The machine learning practitioner is confronted with a vast decision space of several choices when addressing the problem of intent recognition. This gap is addressed by the work in this thesis with the intent recognition framework combining hierarchical models with machine learning and manifold learning. The framework narrows the decision space and streamlines the process of solving the intent recognition problem on a given data set. The reader is directed to chapters 4, 5 and 6, where the process is demonstrated on three different examples from distinct application domains.

CHAPTER 3

Intent Recognition Framework

3.1 Overview

This thesis presents a general framework which can be applied to predict the intent of an autonomous agent. A key assumption of the framework is that the agent is a black box; the intent of the agent is not directly observable at the time of making the prediction. The framework allows the application of probabilistic reasoning to manage the uncertainty that arises with this assumption.

When a machine learning practitioner confronts an intent recognition problem, there are myriad of decisions required of them and several steps they may explore to tackle the problem. The key contribution of the framework proposed in this thesis is to narrow the scope for the practitioner and prescribe a plan of action by which they may tackle their problem, thereby simplifying their task. The implementation examples described in section 1.4 and the associated experimental results presented in chapters 4, 5 and 6 establish that this framework achieve results comparable to external benchmarks for intent recognition across different use cases and different domains. Where a directly relevant external benchmark is not available for intent recognition, baseline models are established and evaluated for comparison.

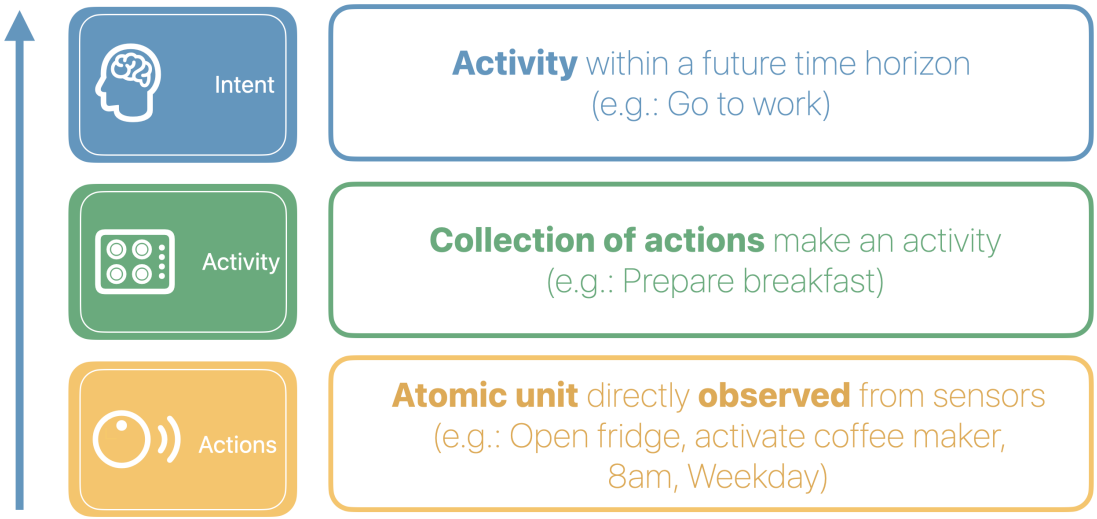


Figure 3.1: Example relationship between actions, activity & intent

It is expected that the data available will provide a sequence of observations relating to or from the agent. This could be in the form of events, time series or even text. Depending on the use case for which the framework is being instantiated, the data could represent localisations in space over time, utterances, physical movements, physiological changes, etc..

The framework is capable of inferring intent with data which capture the effects an agent has on their environment; i.e.: the framework is capable of handling indirect observational data (consider the implementation example for activities of daily living in chapter 4 where the sensors are installed around the house detecting a door or cupboard being opened by the resident rather than an accelerometer or inertial measurement unit (IMU) carried on their person).

3.2 Actions, Activity & Intent

As a matter of first order, the fundamental terms on which this framework is built need to be defined. These terms are "*actions*", "*activity*" and finally "*intent*". Figure 3.1 illustrates the relationship between these terms.

A description of these terms are below:

Actions These are the atomic units observed directly in the data set. For instance, in smart homes, actions would be reports from sensors around the home which indicate the resident has opened a cupboard or activated the coffee maker, etc. (something that denotes an effect the agent has had on their environment). In a moving vehicle application, the actions would be

individual GPS locations recorded and in an access control application, the actions are the card swipes by a given employee on the readers in their building.

Activities An activity is a collection of actions typically performed to achieve some goal. However, the precise definition depends on the application the framework is being applied to. The collection, therefore, maybe a sequence, a set or some other structure depending on the application. An activity may also be repeated actions or a set of just one action. Sticking with a similar example as above, in a smart home, the following collection of actions might indicate the *prepare breakfast* activity:

{*previous sensor fired*: open fridge, *current sensor fired*: coffee maker}

The framework also allows for capturing relevant context alongside observed actions i.e.:

⟨{*hour of day*: 8am, *minutes since last sensor report*: 3}, {*previous sensor fired*: open fridge, *current sensor fired*: coffee maker}⟩.

In a moving vehicle application, the activities would be the transitions made by the driver from one GPS location to the next and relevant context is the time of the day.

In an access control application where the actions are all the card swipes observed by a set of employees in a building, the activity could be a single card swipe at a particular reader by a given employee (an example where the activity is a set of one action). Examples of relevant context in this use case are, for instance, the day of the week and whether it is a public holiday or not.

Intent While activities are defined as a collection of actions, *intent*, however, is defined as an activity taking place in the near **future**. Concretely, then, intent is an activity the agent will perform within a predefined window of time ahead (known as the *intent horizon*).

Depending on the specific use case the framework is being applied to, both actions and activities may be directly observable (for example, in moving vehicles where observed actions are being in a particular discrete location and activities are

transitions between discrete locations or in access control where the swipe of their card by an employee at a given reader are directly observed).

In other use cases, however, the activities may not be observable. Rather they are latent and represented by a distinct set of labels at a higher level of abstraction (such as the case in activities of daily living where the activities are "making breakfast", "doing the laundry" and so on).

In the activities of daily living example, as noted, the "activity" is specified by a set of labels which correspond to commonly accepted activities of daily living (ADLs) such as "cooking a meal", "going to work", etc.. Therefore, in this particular example, there is a close correspondence to what one would consider as activities in the standard English language sense. However, the definition of an activity as described by the framework is also respected since each activity is represented in the model as a collection of observed sensor reports (most recent and previous) combined with contextual features (time of day and seconds since last sensor report at time of inference).

In the other two implementation examples, the definition of an activity is also a collection of observed actions; for the vehicle destination prediction example, the action is being in a discrete location and the activity is a transition between those discrete locations represented in the model as:

$$\{\textit{previous location ID}\text{: } 56,\ \textit{current location ID}\text{: } 64\}$$

In the access request example, the activity is defined as the most recent card swipe by the employee modelled as a collection of one item; the most recent reader ID swiped by the employee:

$$\{\textit{last reader ID swiped}\text{: } 13\}$$

In both of the cases described above, while the example provided is the definition of activity, the model also leverages contextual variables when making the prediction. For instance, in both cases, moving vehicles and access control, the model would consider the time of the day in making its prediction. A precise example of how these contextual variables are captured alongside the activity are described above.

Once the definition of an activity is established, the intent is then defined as an activity occurring in the near future (parameterized by the *intent horizon*). Figure 3.2 summarises an overview of how actions, activities and intents are defined in all three implementation examples.

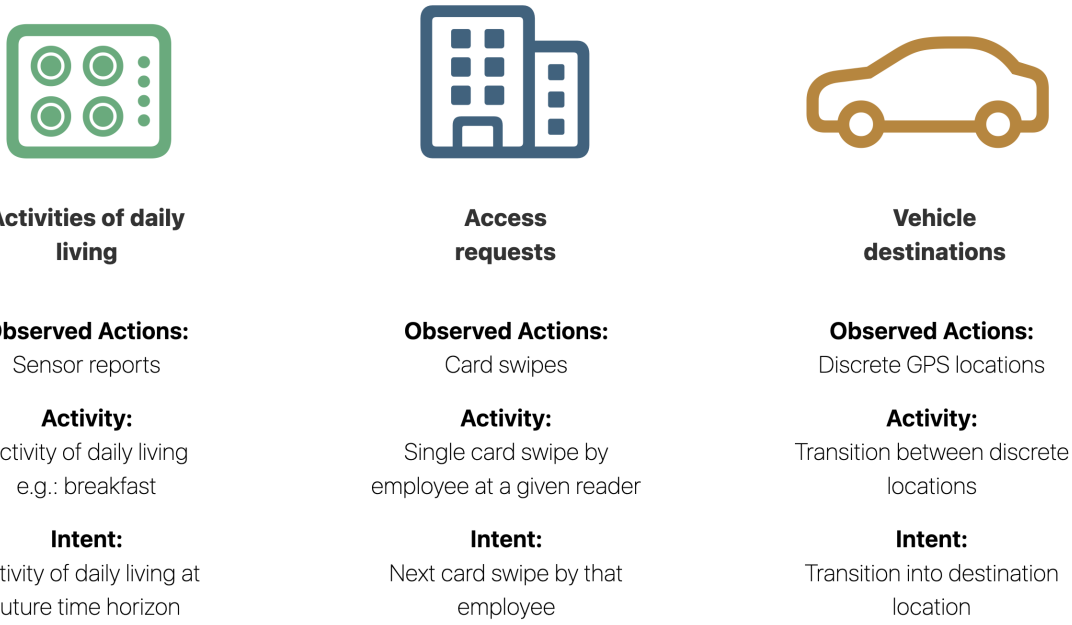


Figure 3.2: The three implementation examples of the framework

3.3 Stages of applying the framework

The procedure to apply this framework is summarised in figure 3.3. As indicated in the figure, there are 4 key stages to applying the framework. The first row in the figure summarise the key steps within each stage of applying the framework. The second row lists some techniques the practitioner may apply within each key step in the framework and the third row in green are the expected outputs from each step when applying the framework to a given problem.

Each of the stages in applying the framework are elaborated below drawing specific details from the implementation examples for clearer understanding.

Stage 1 - Data preparation This stage involves loading the data in and carrying out a series of pre-processing steps such as cleaning, discretising (if necessary) and splitting into training and test data sets. The method utilised for each of these tasks should make sense for the particular use case. For

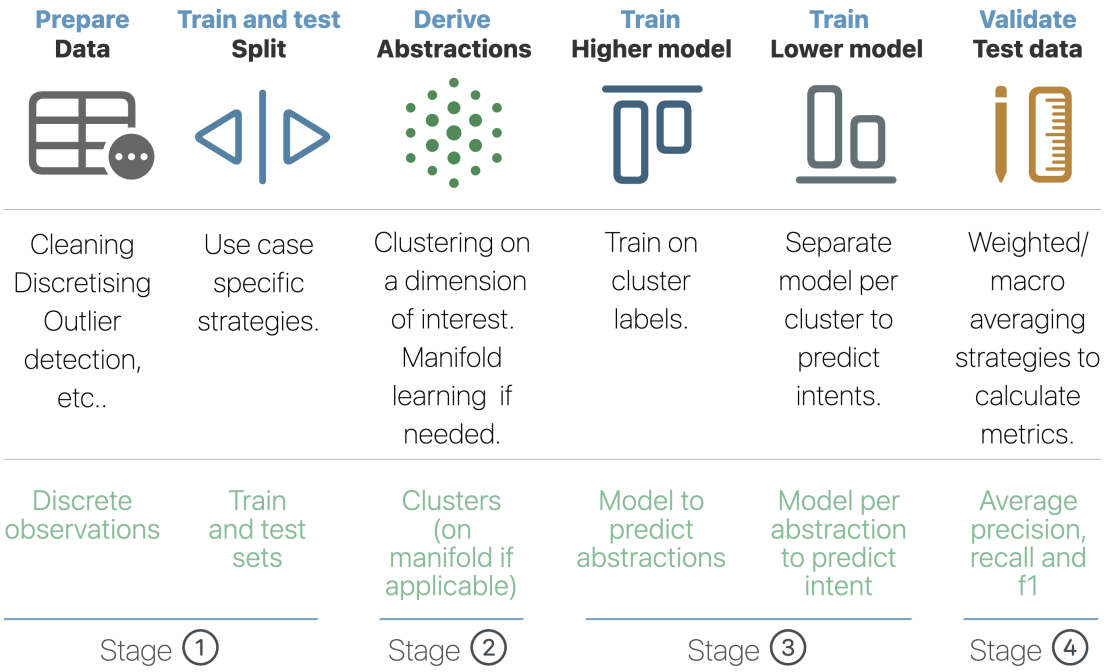


Figure 3.3: Framework stages

instance, in the activities of daily living, train and test sets were split on full days recorded. It would not be a clean separation of the data to have half of a day in the training set and other half of the same day in the test set. Similarly, for the moving vehicles; care was taken during the train and test data set split to ensure only complete trips with a pick up and drop off location inclusive were assigned into each subset.

Data cleaning must be carried out at this stage. The specific steps required to clean the data are unique to every data set of course, however, some general points to consider are detecting outliers in the data and applying a strategy for managing them, dropping duplicates in the data, detecting and imputing missing values if the chosen model cannot handle missing data, etc.. For example, specific data cleaning steps carried out in the moving vehicles implementation example were figuring out which trips had missing data and which trips strayed outside the region of interest (the reader is referred to chapter 6 for more detail on how each of these were achieved).

Some data sets may require specific pre-processing steps due to their nature (such as being continuous). The goal of pre-processing should be that a series of discrete observations emerge that are split into training and test sets which then enter the next stage in this pipeline which is clustering and embedding. An example of a specific pre-processing step carried out in the

activities of daily living implementation example is discretisation. In this example a portion of the data set was available as continuous activity logs where an activity label was associated with a start and end time. Before applying this framework, the data set had to be discretised into time "slices" (windows of fixed duration, 60 seconds in this case).

Stage 2 - Clustering/embedding The goal of clustering stage is to learn an abstraction of the data based on its inherent structure. This abstraction forms the higher level of the hierarchical approach. The advantage of establishing the hierarchy is that the system can make a 2-step guess; first predict at the abstraction level and then "refine" the prediction to a more precise output.

In this framework, clustering should be carried out along a dimension of interest that is expected to yield informative sub-groups. For instance, with the moving vehicles example, we could cluster GPS coordinates of drop off locations to establish a drop off "region" that is predicted first at a coarse level. This prediction may then be refined to a more precise prediction within the higher level region.

As another example, with the activities of daily living in a smart home, the clustering is calculated on the frequency of counts for each sensor associated with each activity. This yields common groups of activities that are associated with each sensor. For a more general clustering, clusters can be learnt by mapping the top 2 or top 3 most frequent sensors for each activity.

The data set is often high dimensional, as is the case with the activities of daily living data set depending on how many sensors are considered in the data (some houses have up to 23 sensors installed). As a general intent recognition framework, an approach is provided where required for such cases; advocating the use of manifold learning techniques to reduce the dimensionality of the observation space with minimal information loss before clustering.

A detailed explanation of how the clustering is calculated for each implementation example is provided in their respective chapters 4, 5 and 6.

Stage 3 - Model training In this stage, two models are trained. The first requirement is a *higher* model predicting cluster label (activity/intent cluster for smart homes or destination cluster for moving vehicles). Secondly, this initial prediction is refined by the *lower* model which may actually be sev-

eral "sub" models (i.e.: a separate model per cluster each trained on only the data within that cluster).

The specific machine learning model chosen at each level can be of any type. However, in this thesis, *Bayesian* and *XGBoost* models are mainly employed across the three implementation examples. Specifically, in activities of daily living (chapter 4), an XGBoost model is used at the upper and lower level. A Bayesian model was also evaluated for this example however the quality of the data was not sufficient to train a Bayesian model. For access control (chapter 5), however, the data was much better quality and a Bayesian model is used to demonstrate how a model capable of reasoning under uncertainty is compatible with the framework given data quality is sufficient. The third implementation example (moving vehicles, chapter 6) employs a Bayesian model at the lower level however the top level is a Discrete Time Markov Chain (DTMC). This example demonstrates how different types of machine learning model may be combined in this framework.

Conceptually the framework is also flexible in that there is no requirement for a separate model to be trained for each cluster. This scheme implies no information sharing between clusters which represents one extreme; every cluster is treated as its own data set ("no pooling"). Conversely, employing a flat non-hierarchical model represents the other extreme where the whole data set is treated as homogeneous with all latent groups and structure ignored. However, the framework also provides a partial pooling option where the cluster label is a feature in the data set. In this case, information is shared between clusters which can enhance predictive performance when some clusters do not contain enough data.

Stage 4 - Model evaluation The final stage is testing the predictive performance of the model on the hold out data set. The appropriate evaluation metrics must be chosen depending on the specific use case. This thesis includes two examples; classification problems are evaluated using f1-score whereas the moving vehicles example involve predicting continuous GPS coordinates, therefore the error in predictive performance is utilised with the mean haversine distance measure.

3.4 Deployment in production

An example flow for the proposed intent recognition framework deployed in production is laid out in figure 3.4. The raw data from the sensors pass through a preprocessing pipeline in a streaming fashion.

The processed data are then the inputs to the higher level model which predicts an initial abstraction of the intent. The prediction from the higher model indicates which lower model to use to infer the intent. This inference will then trigger a response action as required by the use case. For instance, in the activities of daily living example, an intent prediction of leaving the house when the most recent activity inferred was preparing breakfast might trigger a reminder to the users phone with respect to any medications they might need to take.

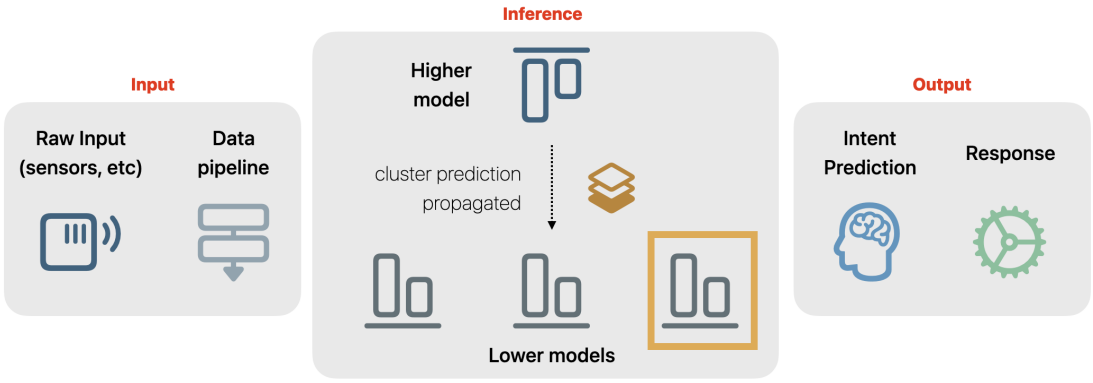


Figure 3.4: Example flow for using this intent recognition framework in production - higher model prediction indicates which lower model to use.

3.5 The clustering approach

To establish the levels (lower and upper) of the hierarchical model, a clustering approach may be utilised such as KMeans [105]. The target on which the clustering algorithm is applied depends on whether manifold learning techniques are utilised in the Stage 2 as described in section 3.1. If manifold learning techniques are applied then the clustering algorithm is applied on the embedding calculated from those techniques. This was the case in implementation examples I (activities of daily living, chapter 4) and II (access control, chapter 5) in this thesis. Otherwise if the data is relatively low dimensional to begin with, the clustering algorithm may be applied directly on the raw data, a method employed in the third implementation example (moving vehicles, chapter 6). Across all these implementation examples, the KMeans algorithm was applied. The ideal values for

k for was evaluated using silhouette scoring as described in section 2.3.5.1. This tuning method may also be used for future applications of this framework where KMeans algorithm is the chosen clustering approach.

In general, the simplest possible clustering algorithm which yields the desired results is the favoured option. In two out of the three implementation examples presented in this thesis, the KMeans algorithm is employed for the fine-grained control the algorithm allows in the number of clusters. KMeans is an ideal starting point as it is easy to interpret, simple to implement and guarantees convergence.

However, it should be noted that the framework does not prescribe a particular clustering algorithm. In fact, if based on domain knowledge, a reasonable set of groups is known in advance or if the quality of data available precludes the use of automated clustering approaches, manual grouping of the data is also possible as was the case in Implementation Example I (activities of daily living, chapter 4). The comparative analysis of multiple clustering approaches in this implementation example which guide the final choice is shown in table 4.4 in section 4.5.

In terms of the overall framework, however, an independent decision for clustering algorithm must be made for each application based on the specifics of the particular use case the framework is being applied to. In this thesis the choice of KMeans algorithm serves as an example of one approach to the clustering step prescribed in the framework.

3.6 Implementation details

The framework was implemented in *Python 3* using the *Scikit-learn* package [127] for the manifold learning algorithms. The Bayesian models were implemented using *pgmpy* package [12]. The XGBoost package [35] was used to implement the tree based models.

CHAPTER 4

Implementation Example I: Predicting activities of daily living

4.1 Introduction

In this chapter, the intent recognition framework is applied to activities of daily living (ADL) prediction.

Automatically recognizing the intent of humans to carry out certain tasks on a daily basis such as cooking, sleeping and showering have many useful applications especially in assisted living scenarios. A system that can detect in advance that a person intends to have breakfast and then leave for work can deliver key insights at the most relevant time such as reminding them to take their medication before leaving.

The problem of predicting the intent of someone to carry out certain activities in the near future relies heavily on the problem of detecting those activities in the first place. Recent developments in sensing technologies have led to solutions which address a number of drawbacks historically associated with activity recognition. These include considerations such as intrusiveness, privacy preservation and ease of installation into existing homes.

Nevertheless, recognizing human activities from data generated by sensors come with a number of challenges:

- The start and end time of activities are not known and will need to be inferred. Indeed, the ambiguity between when one activity ends and another begins is exacerbated by the fact that there is often overlap between successive activities. Possible mitigation strategies for this issue include making assumptions on each discretised time slice. For instance, given a 1 minute window where more than one activity label is recorded, the activity that takes up the majority of that time slice is assumed to be the overall label for that time slice. The secondary activity will continue and become the main label from the next time slice.
- There is ambiguity which activity is indicated by the same sensor report. For instance, cooking and getting a drink both involve opening the fridge which would trigger the corresponding sensor. In our approach, we mitigate this by building more contextual awareness into our models.
- Activities can often be performed in a variety of ways and some actions which comprise an overall activity are often interchangeable in terms of order and still achieve the desired goal. Our models account for this by by learning the habits of the resident over time, in effect, the model starts with a prior but over time learns the mapping of particular sensor report sequences to a given activity label for that resident.
- There is a reliance on human recording to collect ground truth for the activity labels (sensor data collection may be automated). This can sometimes result in mismatch between sensor data and activity labels.
- In addition, to label mismatching, observed sensor data itself is inherently noisy. The noise can be technical, for example, the sensor report is not transmitted to a network glitch. Or the noise can arise from human mistake, for example, opening the wrong cupboard door by accident.

4.2 Data description

The data used in this implementation example is a publicly available data set of activity and sensor logs from residents living in 3 houses. The data sets have sensor events from the smart environment and activity labels for each house [162].

Table 4.1: Summary of data for Implementation Example I

	house A	**house B**	**house C**
resident age	26	28	57
setting	apartment	apartment	house
rooms	3	2	6
duration (days)	25	14	19
sensors	14	27	22
activities	16	26	28
annotation	bluetooth	diary	bluetooth

These activities include watching TV, making food, sleeping, showering etc., which are known in the literature as *activities of daily living* (ADLs). The layout of the 3 houses are described in the following sections [1]. A summary of the data set is listed in table 4.1. The summary and description of the data in the coming sections is adapted from [162].

4.2.1 Sensors used

A commercially available wireless network kit (the RFM DM 1810) was employed by the authors of [162] when collecting this dataset. They chose this option based on the quality of available technical documentation and an energy efficient network protocol being available for data transfer which maximises the battery life of the sensors. The network has both analog and digital inputs. An event is generated when a digital input changes state (1 to 0 and vice versa) or when an analog input crosses a threshold. The authors equipped the nodes with several different kinds of sensors; reed switches to detect whether doors of cupboards and rooms are open or closed; pressure mats to sense a resident sitting on a couch or lying on a bed; mercury contacts to detect movement of objects such as drawers; passive infrared (PIR) sensors detect motion in a given area such as hallway or corridor and float sensors detect when the toilet is flushed. All sensors generated binary input in this use case.

4.2.2 Annotation method

The annotation method refers to how the activity labels were recorded by the residents of the houses. Activity annotation was performed using either a hand-written activity diary or a Bluetooth headset working in concert with speech

[1] Layout figures reprinted by permission from *Springer Nature*. Details in appendix B

recognition software. The beginning and end time of the activities were captured in both instances.

4.2.2.1 Handwritten diary

In House B, the activities were annotated using a handwritten diary. Sheets of paper were placed at locations where activities are performed. The resident would note the beginning and end time of an activity by checking their watch and making a note on the closest sheet.

The advantage of this method is its ease of installation. However, the disadvantage is the amount of effort involved in entering the data into a computer system and the time stamps as measured by the residents watch may differ slightly from the time stamps of the computer recording the sensor data. Overall, this annotation method is less automated and therefore more prone to human error.

4.2.2.2 Bluetooth method

The Bluetooth annotation method relies on a set of prescribed keywords delivered via a Bluetooth headset to record the beginning and end time of the activities. The authors of the study [162] utilized a Jabra BT250v Bluetooth headset. The chosen model had sufficient battery power to last a full day of activity annotation without interruption. The range was 10 meters which was deemed enough for the houses in question. A physical button was available on the headset which was used to trigger the software and initiate a new activity label entry.

The software for processing the annotations was written in C and utilizes both the Bluetooth API and the Microsoft Speech API [2]. The Speech API was a solution for the speech recognition and text-to-speech elements of the task. The authors of [162] created their own speech grammar which contained a limited set of commands such as "begin take shower" and "end make breakfast" which the recognition could recognize. The authors noted that by using a small number of distinctive commands, the algorithm had multiple words by which it could differentiate between different commands which resulted in a near perfect recognition performance of the voice commands during annotation via the Bluetooth method. The Bluetooth API was required to catch the event of the headset button being pressed to activate the speech recognition. The authors utilized the Widcomm Bluetooth stack in their solution [3].

[2]Microsoft Speech API: *http://azure.microsoft.com/en-gb/services/cognitive-services/*
[3]Widcomm Bluetooth stack: *http://www.broadcom.com/support/bluetooth*

4.2.3 House A

This is an apartment with 3 rooms. 14 Sensors were installed at the locations indicated on figure 4.1. The red boxes indicate the locations of the wireless sensor nodes.

25 days of activity are recorded for this house covering 10 distinct activities. Annotation of activity labels were carried out by the resident using the Bluetooth method.

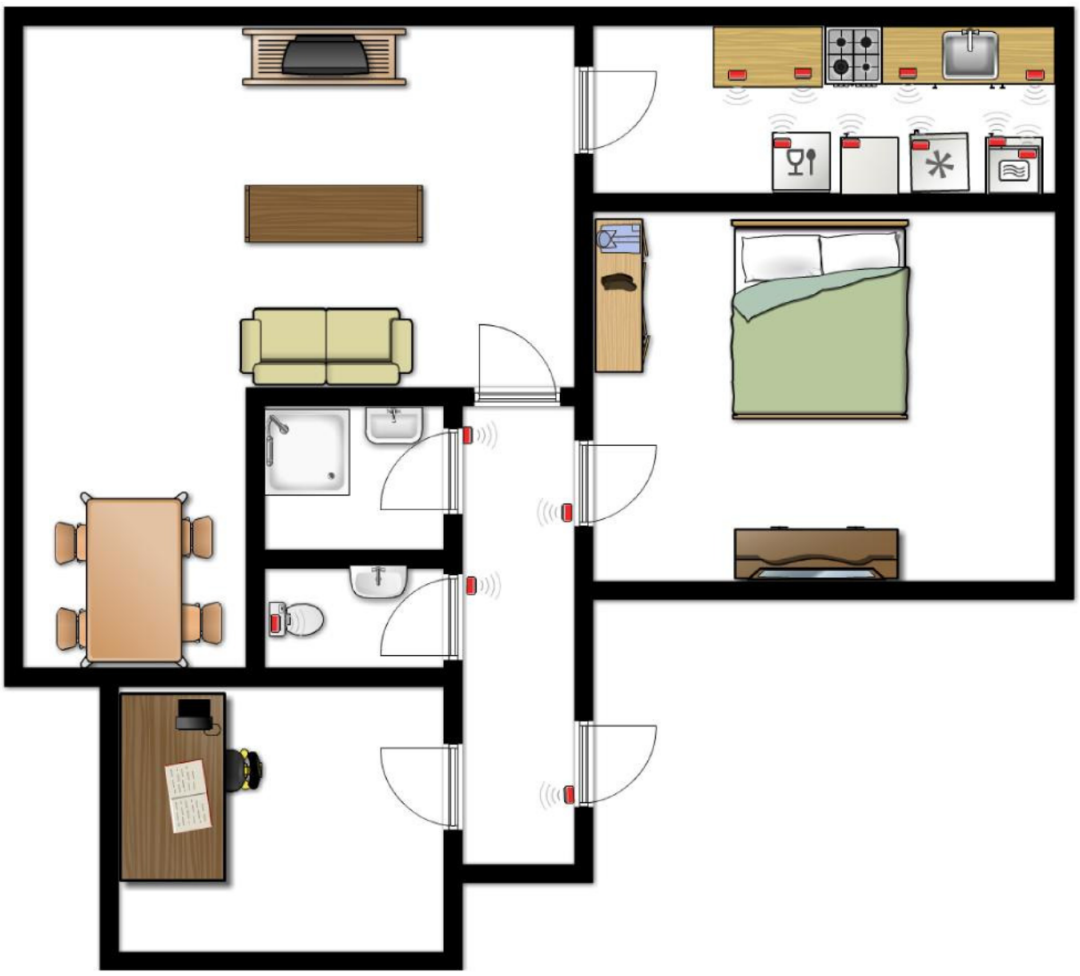


Figure 4.1: Layout of House A (figure from [162])

4.2.4 House B

This is an apartment with 2 rooms. 23 Sensors were installed at the locations indicated on figure 4.2. The red boxes indicate the locations of the wireless sensor nodes.

14 days of activity are recorded for this house covering 13 distinct activities. Annotation of activity labels were carried out by the resident using a hand written diary.

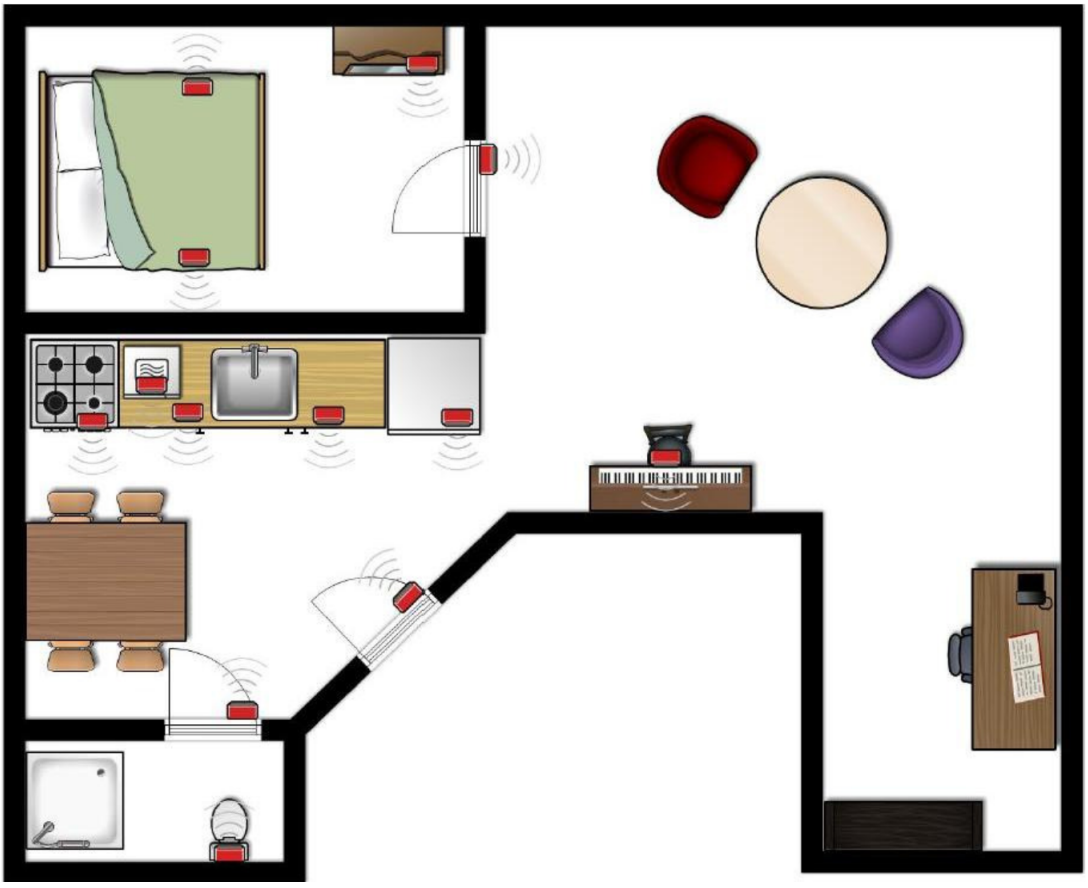


Figure 4.2: Layout of House B (figure from [162])

4.2.5 House C

This is a house with 6 rooms. 21 Sensors were installed at the locations indicated on figure 4.3 and figure 4.4. The red boxes indicate the locations of the wireless sensor nodes. 19 days of activity are recorded for this house covering 16 distinct activities. Annotation of activity labels were carried out by the resident using the Bluetooth method.

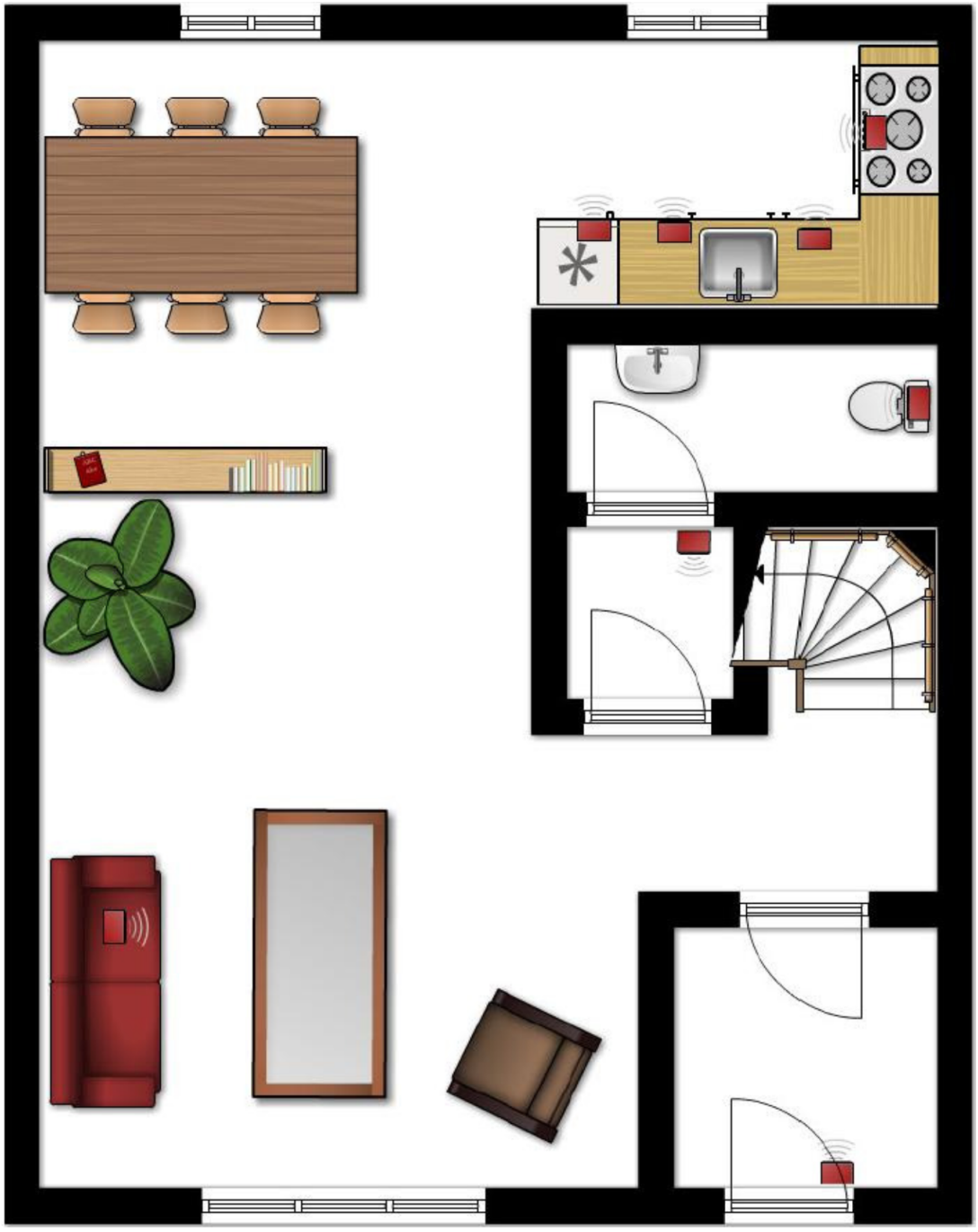


Figure 4.3: Layout of House C (first floor, figure from [162])

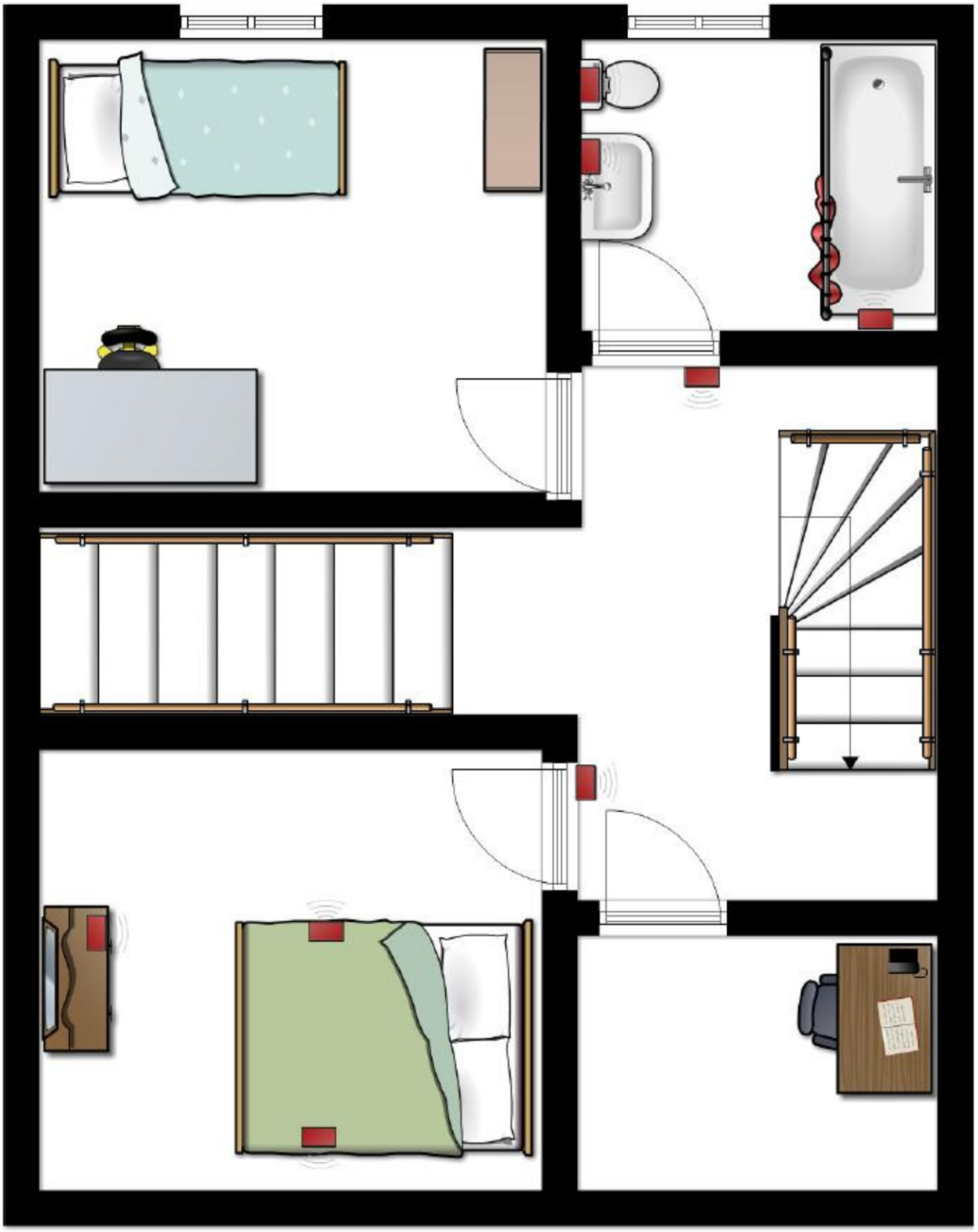


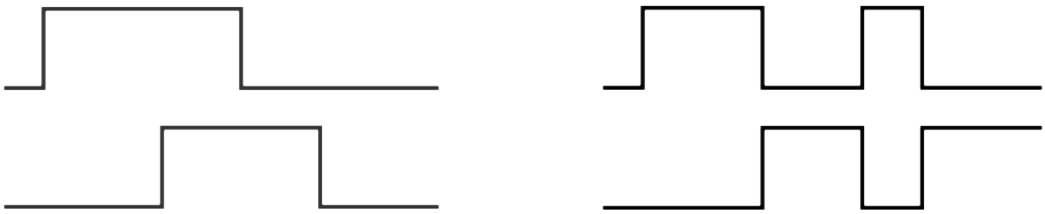
Figure 4.4: Layout of House C (second floor, figure from [162])

4.3 Feature representation

The raw data obtained from the sensors directly were first transformed into a different representation form. The original data were separate event logs of sensor firings and recorded activities. When processing the sensor events while modelling the problem, different representation may be utilised. Two such representations are described below:

Raw The raw representation uses the sensor data directly as received by the sensor. In this representation, a 1 is recorded when the sensor is firing and a 0 otherwise (figure 4.5a).

Last-fired The last-fired sensor representation reports which sensor fired last. The sensor that changed state continues to give 1 and changes to 0 only when another sensor changes state (figure 4.5b).



(a) Raw data representation (b) Last fired sensor representation

Figure 4.5: Sensor representations

The authors of [162] conclude that the *Last Fired* representation performs better than the *Raw* representation. This finding was confirmed in the present analysis as well. The main reason is that the raw sensor data can be noisy. For instance, if the toilet door is left open, the sensor will continue to fire and record 1 even though the resident may now have moved on to other activities. Therefore, it is better to focus on the most recent sensor firing as an indication of the most recent activity or indicator of the residents intent.

4.3.1 Data discretisation

The models investigated in this thesis require discrete time series data. Therefore, the raw data logs must first be discretised into time slices of constant length. Under this scheme, the data obtained from the sensors will be discretised into T timeslices of length ΔT. For each time slice t, the state of each sensor is denoted $x_i^t \in 0, 1$. A 1 indicates the sensor was firing at some point during that particular timeslice and 0 indicates there was no activity for that sensor during the time

slice. It is possible that the sensor started firing in the middle of a time slice and finished firing before the completion of a time slice. In both of these cases, a 1 will be recorded for that whole time slice. These discretised ground truth labels are used to learn the model parameters. The analysis carried out by the authors of [162] found that 30 second and 60 second discretisation granularities performed most consistently across the three houses in this data set. They went on to report their detailed experimental results for activity classification with the 60 second discretisation. As this implementation example leverages the data set provided by the authors of [162], the results of their analysis also inform the present approach and therefore, the 60 second discretisation granularity is adopted in this implementation example.

Regardless of the granularity, the discretisation process introduces an error. In the experimental results, the discretisation error is accounted for by evaluating the performance of the framework at the same level of discretisation in the training and test sets. Therefore, the level of discretisation error is consistent across training and evaluation, thus allowing the performance of the methods and approaches proposed to be evaluated without factoring the discretisation error explicitly. However, were this intent recognition system to be deployed in production, the discretisation error would also have to be quantified and included in the performance evaluation to reflect the actual end user experience as closely as possible.

4.4 Data embedding

For the experiments where a clustering strategy is evaluated, an embedding of the raw data is first generated. The purpose of the embedding is to reduce the dimensionality of the original data (often high given the number of features observed in the environment of the agent). The embedding procedure therefore reduces spurious correlations being detected due to curse of dimensionality [22]. The process to generate the embedding matrix in this implementation example will now be described.

A matrix of counts is calculated where the rows represent activities and columns represent sensors in the house. An entry (i, j) in this matrix represents the number of times activity i is associated with sensor j. The logic underlying this approach is that activities that associate with common sensor firings must be "similar" and therefore candidates for clustering.

More concretely, consider house A, where the data records 17 activities (including the synthetic "GAP" activity representing periods in time where no other activity is recorded) and 14 sensors, a matrix is first established of size 17×14. Entry (i, j) in this matrix is a frequency count of how many times activity i was recorded while sensor j was the last sensor that fired. This matrix is then passed to the embedding algorithm which produces a new matrix of size 17×2. This is now a 2 dimensional representation of the data in the original larger matrix.

The clustering algorithm is then applied to this 2 dimensional representation which yields clusters or groups of activities. The approach is driven by the sensor activity in the residence. These sensor reports are generated by the resident in the house while carrying out their activities of daily living which makes this clustering scheme of grouping activities based on sensor reports a suitable approach given the goal is to model the intent of a singular resident in this implementation example.

4.5 Clustering of activities

In this implementation example, clustering of activities was manually carried out. Three clusters were created at the higher level. The activities "go to bed" (which also captures time spent sleeping in bed) and "leave house" (which includes all the time spent outside the house) were assigned into their own cluster each. All other activities were then assigned into a separate cluster. This manual scheme was inspired by comparing the clustering output across several subsets of the training data in a process described later in this section.

The decision to carry out manual clustering was made based on experimental evaluation where it was found that automated clustering approaches did not produce superior results to the manual clustering approach given the data set for this particular implementation example (these results are detailed and discussed later in this section).

The automated clustering procedure was to first calculate the embedding which results from the procedure described in section 4.4. This embedding is then clustered using the chosen clustering algorithm. In this evaluation, KMeans, DBSCAN and Mean Shift were evaluated as clustering algorithms while t-SNE and LLE were evaluated as candidate embedding algorithms (please refer to section 2.3.3 and 2.2.3 for the background on these clustering and embedding methods). For completeness, a direct clustering approach was also evaluated where the raw

counts data was clustered as is with no dimensionality reduction.

In the case of KMeans, silhouette analysis [138] was used to determine the best value for the number of clusters k as described in section 3.5. In this implementation example the range of possible k is selected to be between two and the number of unique activities recorded for each house. Therefore, the number of clusters in practice can vary (with k selected to maximise the silhouette score).

While the automated clustering approach achieved the desired goals in the other implementation examples, it became evident that the data quality of the present example was not sufficient for applying automated clustering techniques. Depending on the way that data is split into training and testing sets, the exact assignment of activities to clusters can vary. Nevertheless, certain patterns were observed when comparing embeddings calculated on different training subsets of the data. The key pattern was that activities 1 and 10 (representing the resident being out of the house and sleeping in bed respectively) always get assigned their own cluster. The issue is the shorter length of the other activities and the fact that the human annotated labels for the activities often don't align exactly with the sensor reports which are automatically captured. These issues mean that the number of data points recorded reliably for the other activities are often insufficient and where data exists, the mislabelling creates too much noise for these activities to be consistently clustered.

To address this, a manual clustering approach was considered. To motivate the manual clustering approach, consider table 4.2. In this table, the horizontal dividers are the results of the first clustering carried out on training subset 1. These dividers provide a frame of reference to compare the subsequent clusterings on different subsets of the data. They allow the reader to see, at a glance, how consistent the clustering is across different subsets of the data. Indeed, across 5 different subsets, it is observed that there are certain activities which almost always cluster together. It should be noted that the numbers in the table are not inherently meaningful, they are only an auto-generated label for each cluster in a particular run. Therefore cluster label 6 for training subset 1 has no relation to cluster label 6 for training subset 4. The numbers only serve to indicate to the reader which activities fall under the same cluster in a given run.

In table 4.2, it is observed that the activities "*go to bed*" and "*leave house*" always fall into their own cluster whereas other activities move around when different sub-sets of the training data are sampled. This suggested a reasonable scheme for a manual clustering would be to assign the activities "go to bed" and "leave

house" into their own cluster while all other activities are assigned into a separate cluster.

To carry out a more formal analysis for which embedding and clustering method to employ and also to determine whether to use the automated clustering method or the manual clustering method, two experiments were run and results analysed. For both of these experiments, the House A data set was used with a 60 minute intent horizon and a 10 minute evaluation window using a hierarchical formulation with XGBoost models (please refer to section 4.7.1 for details on these experimental parameters and configurations).

These evaluations were carried out end-to-end i.e.: for each clustering and embedding algorithm evaluated, the full experimental pipeline was run including training/test split averaged over cross validation folds, the specified embedding and clustering step followed by training the higher and lower models and finally the evaluation of the classification models trained. The values for precision, recall and f1-score reported in tables 4.3 and 4.4, therefore, represent the performance of the final model trained under the specified embedding and clustering configurations.

In the first experiment, the clustering algorithm was fixed and the embedding algorithm varied. The results for the first experiment is shown in table 4.3. This experiment indicated that t-SNE is the best performing embedding algorithm for the specific data set in this implementation example.

For the second experiment, then, the embedding algorithm was fixed as t-SNE and various clustering algorithms were evaluated. The results from this second experiment was also compared against the manual method proposed above to have a complete picture for comparison. The results from this second experiment are tabulated in table 4.4.

It is observed in the clustering analysis that among the automated methods, DBSCAN performs better the rest. In the absence of any appreciable performance advantages, KMeans has certain inherent advantages over other methods for clustering such as being more finely controllable and interpretable. In the case of the results observed in tables 4.3 and 4.4, though, DBSCAN is the ideal choice from all the automated methods that were investigated.

However, the analysis also shows that in the case of this particular data set, the best performance is seen experimentally with the manual clustering scheme. Based on these trends observed, the decision was made to cluster the activities

Table 4.2: Clustering examples on different training subsets

training subset	**1**	**2**	**3**	**4**	**5**
activity					
get drink	0	0	7	6	0
load dishwasher	0	0	6	6	5
use toilet	1	2	3	1	6
go to bed	2	4	2	2	1
get snack	3	3	7	7	5
unload dishwasher	3	0	5	6	0
prepare breakfast	4	3	1	4	3
take shower	5	1	4	3	2
eating	6	0	1	9	0
brush teeth	6	0	1	0	0
prepare Dinner	6	6	6	8	7
store groceries	6	0	1	9	0
load washing machine	6	0	1	9	0
unload washing machine	6	0	1	9	0
receive guest	6	0	1	0	5
leave house	7	5	0	5	4
number of clusters	8	7	8	10	8

Table 4.3: Comparing embedding algorithms

embedding	**clustering**	**precision (%)**	**recall (%)**	**f1-score (%)**
LLE	KMeans	75.6 ± 16.0	72.3 ± 10.9	71.2 ± 13.0
t-SNE	KMeans	74.6 ± 14.9	73.6 ± 10.9	71.7 ± 13.1
None	KMeans	75.6 ± 15.0	72.5 ± 11.3	71.3 ± 13.7

Table 4.4: Comparing clustering schemes

clustering	**embedding**	**precision (%)**	**recall (%)**	**f1-score (%)**
KMeans	t-SNE	74.6 ± 14.9	73.6 ± 10.9	71.7 ± 13.1
DBSCAN	t-SNE	77.4 ± 11.6	75.2 ± 10.9	73.2 ± 12.8
Mean Shift	t-SNE	76.4 ± 13.9	73.9 ± 10.9	72.5 ± 13.2
Manual	N/A	79.5 ± 11.3	76.3 ± 11.5	75.3 ± 13.2

manually in this implementation example.

It should be noted that a manual clustering scheme was only feasible because the present implementation example is a domain where humans intuitively understand how different activities of daily living are similar or distinct from each other. In some other applications of the intent recognition framework the similarities may not be immediately obvious or the relevant contextual data may not be available for privacy reasons. In these situations the automated clustering approach provided by the framework may be utilised.

4.6 Framework instantiation

In this section, the procedure to instantiate the general framework described in chapter 3 for this specific example is elaborated.

4.6.1 Definition of *action*, *activity* and *intent*

In this implementation example, the observed *actions* are the sensor firings detected as residents go about their daily routine. An *activity* is a higher level of abstraction indicated by a sequence of sensor firings and temporal context. For example, a sequence of sensor firings for coffee maker and stove at 8am in the morning would indicate a breakfast activity. These terms, as described, are not strictly actions and activity from the resident's point of view. An action from the resident's point of view could be turning on the coffee maker rather than a sensor report implying the same. However, in a deployed setting where this framework is being applied in practice, the system will not have direct observability of the resident's actions only the sensor reports generated by those actions.

To be precise, the sensor readings which are detected by the system are indirect observations of the actions the resident is performing and it is these which are the observed "actions" from a framework point of view. The same reasoning holds for activities where a sequence of actions by the resident (take out a cup and turn on the coffee maker) indicates a breakfast activity rather than a sequence of sensor firings. There is an inherent uncertainty in the activity being performed by the resident since the same sensors may associate with multiple activities and some activities may not generate any sensor firings at all. The framework infers activities that can be discerned from sensor reports and temporal context trained on labels for activities that are being performed at the time. It should be noted that while these labels are available for activities in the training data, such a label

will not be accessible at prediction time. Given these constraints and nuances, the observed sensor reports and associated inferences are used as described here when applying the framework to this problem.

Intent is defined as an **activity with a forward time component**. So if 30 minutes after the breakfast activity one would expect to see a leave house activity on a weekday then leaving house is the intention in the above example.

4.6.2 Hierarchical structure

Once the notions of actions, activities and intent are established, the hierarchical structure of the problem needs to be considered and established.

There are two levels in the hierarchical model. As is the case in all implementation examples of the framework, clusters are utilized as abstractions which capture hidden structure in the data. For a description of how clusters are defined in this particular example, please see section 4.5.

One benefit of a hierarchical approach in this example lies in the fact that less common activities or shorter activities are predicted with greater accuracy. The data set is highly imbalanced in terms of the class distribution. Without the hierarchical structure, the intentions of the resident that occur for long periods of time such as sleeping and being out of the house can overpower the less frequent intents and activities. The hierarchical model with clusters or groups of activities at higher level factorise the data and separate the less frequent intents and activities into their own group which in turn allows the framework to predict them more accurately. The key reason is shorter and less frequent activities fall into separate clusters from very long activities such as sleeping.

Once the levels are established via clustering, the upper level model is trained on the cluster labels such that it predicts the activity or intent cluster to which the present set of observed actions belong. These predicted cluster labels (at higher level) are propagated to the lower model which predicts the intent of the resident (or the current activity they are performing in the benchmark case). The target variable for prediction in the lower model is *intent* given a particular activity or intent cluster.

In the hierarchical formulation, every cluster gets *its own lower level model* trained only on data corresponding to that group. This results in improvements in predictive performance for these rarer or shorter activities as observed when comparing the experimental results for the hierarchical and flat formulations of the model.

To establish this empirically, ablation experiments were carried out where the flat model was compared against a hierarchical model. The results show improved predictive performance overall for the hierarchical approach (more details in section 4.7).

4.6.3 The models

Two types of models are explored in this implementation example; tree-based *XGBoost* models and *probabilistic* models.

XGBoost models were selected for this implementation example as the raw data collected for the sensors are very noisy. There are also inaccuracies in the alignment between sensor reports and activity reports (sensor data is collected automatically by the system whereas activity data is self reported by humans). Sometimes the activity is reported to begin sooner than the relevant sensor report or vice versa. Gradient boosted models have shown great versatility overcoming such challenges in structured data [83].

The choice also serves to demonstrate the versatility of this framework by showing how different types of machine learning models can be leveraged effectively within the proposed intent recognition framework. For more detail on XGBoost models, please see section 2.5.

The second type, probabilistic models, provide a quantification of uncertainty and a mechanism for injecting prior knowledge out of the box. Despite their inherent advantages, however, these probabilistic models struggled to perform well given the shortcomings of the present data set. For completeness, results from the probabilistic model experiments are also included in section 4.7) for comparison.

4.7 Experiments & results

4.7.1 Experiment description

There were 4 experiments run for intent recognition; a flat probabilistic model, a flat XGBoost model, a hierarchical XGBoost model and finally, a baseline model to compare the methods evaluated.

The baseline model uses the current activity prediction as a prediction for future intent. Intuitively, the baseline is to assume the intent of the resident is to continue with their current activity until the intent horizon (the time point in

the near future at which a prediction is made).

The flat model is essentially a single clustering layer and serves as an ablation study to demonstrate the benefits of the hierarchical aspect in the proposed intent recognition framework. Two flat models (probabilistic and XGBoost) were evaluated and the better performing of the two (XGBoost) was used in the hierarchical model experiment.

The performance of the proposed framework in this example is evaluated by first splitting the full data set into training and test sets.

There are 58 days of data recorded in the full data set across 3 properties (2 apartments and 1 house). The split is made using a *leave one out* approach on the days recorded in the data. In this case, one day of sensor readings is used as the testing set and the remaining days make up the training set. This is cycled over all the days and average performance measures are calculated and reported.

A summary of the experiments run for this implementation example are described below. In all of the experiments listed below, the target for prediction is the intent that the user will perform within a defined time horizon in the near future (intent horizon). The definition of intent and how it differs from activity is described in section 4.6.

Experiment 1 - Baseline A sanity check; this first experiment serves to benchmark the performance of the framework by establishing a common number to compare against. The baseline generates a prediction assuming the resident continues their current activity at the time horizon.

Experiment 2 - Probabilistic flat model Intent prediction carried out using a Maximum Likelihood estimator directly applied without any hierarchy. This experiment is used compared with experiment 3 to establish which model to take forward into hierarchical tests. Following comparisons, XGBoost was selected as the better performing model.

Experiment 3 - XGBoost flat model Intent prediction carried out using an XGBoost classifier directly applied without any hierarchy. This experiment is also an ablation study to evaluate the hierarchical aspect.

Experiment 4 - XGBoost hierarchical model Intent prediction carried out using an XGBoost classifier with a hierarchical formulation. First, an XGBoost model is trained at higher level to predict the cluster label. At the lower level, a *separate* XGBoost model is then trained *for each cluster* to

predict intent. These lower level models are each trained on only the data pertaining to their respective cluster or group.

The performance of the framework is evaluated using the corresponding feature representation and time discretisation parameters that were evaluated in detail in [162]. Specifically, a time discretisation of *60s* and *last sensor fired* feature representation is used. This means that every time slice in the training data represents 60s intervals and a sensor firing event is propagated until the next sensor fire event (see figure 4.5b for a visual representation).

For the intent recognition experiments, two additional parameters are introduced; the *intent horizon* and *evaluation window*. The intent horizon is the period of time in the future at which the user plans to initiate or begin acting upon their intentions.

The evaluation window captures the idea that given a forward looking time period for an application such as assisted living, it is not just relevant to predict if the resident will carry out a certain intent in exactly 30 minutes. Rather it is also a useful and relevant result to know whether the resident carries out their intention in a certain window around the 30 minute mark. Therefore, the evaluation window is defined as the number of minutes before and after the intent horizon in which the user initiating the predicted intention is still a relevant result

The intent horizons evaluated are 10 minutes ahead, 30 minutes ahead and 60 minutes ahead. The evaluation windows are 5 minutes, 10 minutes and 15 minutes. Putting these together, the interpretation of "intent horizon of 30 minutes with evaluation window of 5 minutes" means if the user initiates the predicted intent within 25 and 35 minutes from current point in time, this is a correct and successful prediction.

It is noted that some combinations of intent horizon and evaluation window do not make sense. For instance, one cannot have an intent horizon of 10 minutes but an evaluation window of 15 minutes. Other combinations of horizon and window, though feasible are not the most ideal. For example, a 30 minute horizon with 15 minute window means that the resident has anywhere from 15 to 45 minutes to begin the next intent the model predicts at the 30 minute mark.

The logical approach is to pair shorter intent horizons with tighter evaluation windows and longer intent horizons with wider evaluation windows (to capture greater uncertainty both from model side and the resident's planning side where their immediate intents are more clearly defined than intents within couple of

hours).

Balancing all these considerations, the approach taken in this thesis is to report all feasible combinations. However, a summary is also created where the following logical combinations are highlighted and presented for clarity. The logical combinations selected are listed in table 4.5.

Table 4.5: Logical combinations for intent horizon and evaluation window

Intent Horizon (minutes)	**Evaluation Window (minutes)**
10	± 5
30	± 5
60	± 10
120	± 15

All experiments and models are run and compared for all feasible combinations consistently. Therefore when a comparison is made between models or against the baseline at a given intent horizon and evaluation window, all the models and baseline figures are calculated with the same parameters to ensure fair and accurate comparisons.

4.7.2 Experiment evaluation

The goal of these experiments is to demonstrate how the intent recognition framework may be applied to this implementation example specifically and to evaluate the performance of the proposed framework on predicting *intent* against a baseline to give a sense of comparative performance and demonstrate how the hierarchical framework is superior to simpler methods.

The definition of intent is the activity recorded by the user a certain number of time slices in the future (known as the *intent horizon*). The intent recognition experiments are evaluated with a model trained on data discretised at 60 seconds per time slice. Therefore labels for intent were created as the target variable which records the activity recorded by the resident at each intent horizon. Evaluation logic was also coded to capture the evaluation window (please refer to section 4.7.1 for more details on intent horizon and evaluation window and the values for each adopted in these experiments).

Time slices without an activity recorded are not included in creating the *intent* labels. In formulating the intent, the more reasonable assumption would be that the next activity the resident begins would be a better indicator of intent. Given this formulation, the label *GAP* will not appear in the intent prediction tables.

The performance of the classification models in this implementation example is evaluated using the metrics of *precision*, *recall* and *f-score* [104] which are described in section 2.4.1. For each of these metrics, an average number is reported. This average was calculated over the folds of the data in the cross-validation (CV). The CV was done using a *leave-one-day-out* strategy as described in section 4.7. Each validation fold, therefore, represents one full day in the training data mirroring the approach of the benchmark paper [162]. To give the reader a sense of the variability and spread of the predictions across the validation folds, a standard deviation is also included in the summary measures reported.

When documenting the results first an overall summary is presented. The overall summary captures the high level trend across all the approaches for all the houses. The overall summary showcases these patterns using the logical combinations for intent horizon and evaluation window as described in 4.5.

After the overall summary, the detailed results for each house are tabulated and house-specific trends are discussed. While the overall summary focused only on the most logical combinations of intent horizon and evaluation window, in the house specific results, all feasible combinations of the two parameters are tabulated. A consistent approach is adopted across all 3 houses in reporting the detailed results. The house level results begin with a series of plots capturing the house specific trends for all feasible combinations of intent horizon and evaluation window. The numeric results are then tabulated in this section.

4.7.3 Summary of Experimental Results (overall)

Figures 4.6, 4.7 and 4.8 show the variation in predictive performance as intent horizon/evaluation window varies for the 3 houses. In each plot, a comparison is made between the baseline, flat and hierarchical models.

For house A (figure 4.6), both flat and hierarchical models exceed the baseline for all combinations of intent horizon and evaluation window. The advantage secured by the hierarchical approach becomes more pronounced in the longer intent horizons.

For house B (figure 4.7), both flat and hierarchical models again exceed the

baseline for all combinations of intent horizon and evaluation window. However, in the shortest intent horizon (10 minutes), it is observed that the flat model exceeds the hierarchical approach by 0.4% (in absolute terms, 84.3% vs. 83.9%). This is in line with an observed trend where the hierarchical approach yields greatest benefit for longer time horizons. This same pattern is observed even in this house where the hierarchical model outperforms the flat model for all other intent horizons with a particularly pronounced benefit in the longer horizons.

In the case of house C (figure 4.8), it is observed that the hierarchical approach consistently outperforms the flat model and the baseline model for all combinations of intent horizon and evaluation window.

Tables 4.6, 4.7 and 4.8 summarise the weighted f1 score averaged across all the cross-validation folds in all the experiment configurations that were run. The numeric values that produced figures 4.6, 4.7 and 4.8 are highlighted in bold.

In addition, the tabulated results indicate how, across the board, the Bayesian model under-performed against the XGBoost model for the flat formulation, thus setting the stage for the XGBoost models to be evaluated in the hierarchical experiments.

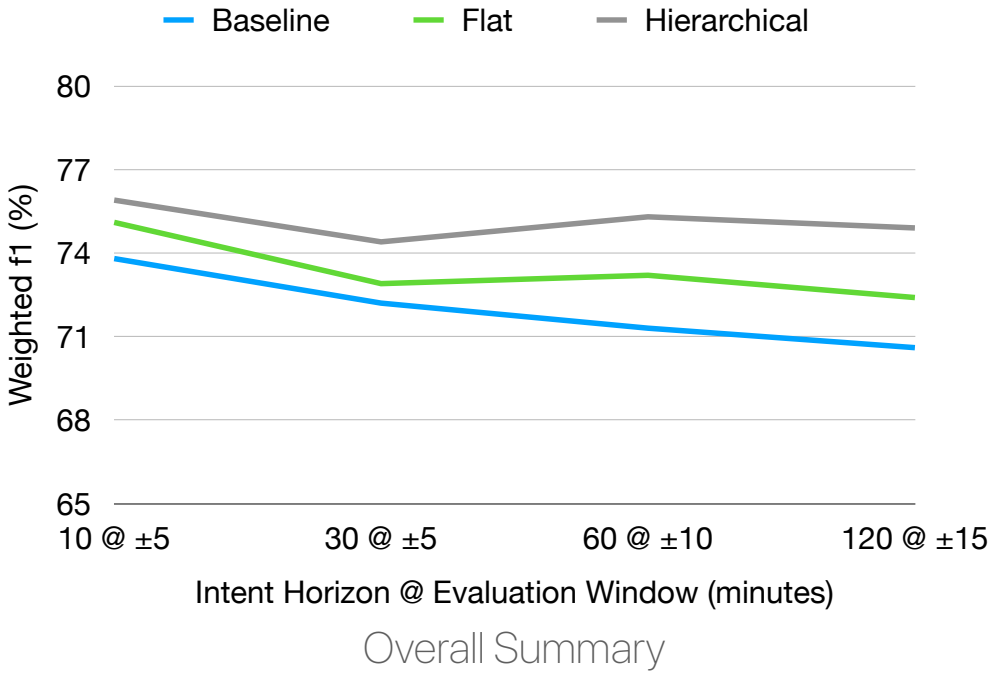


Figure 4.6: House A - Plotted Overview Results

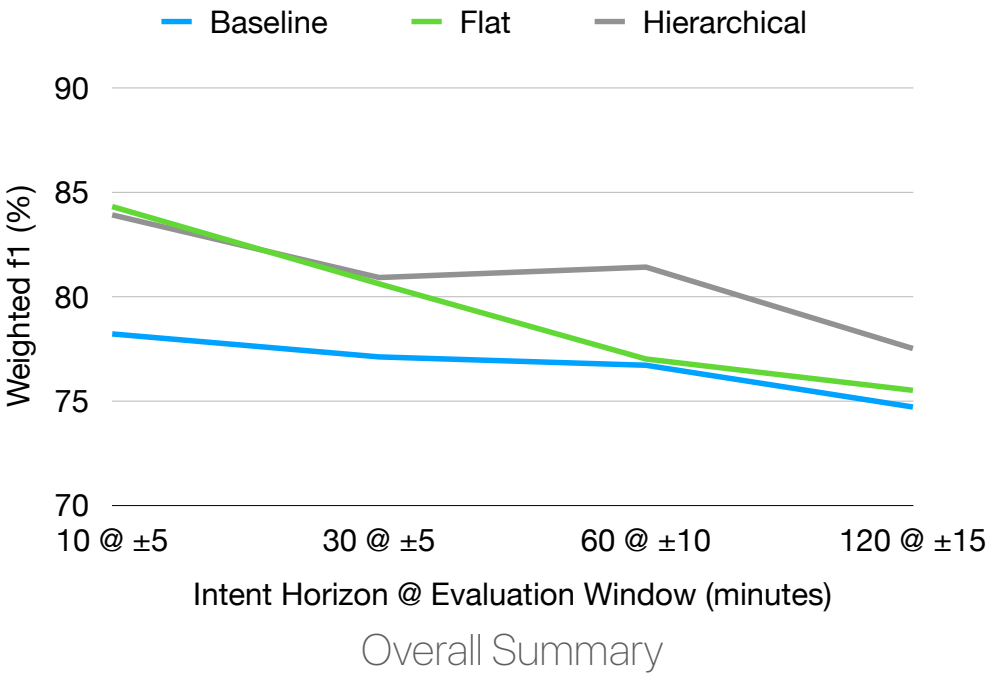


Figure 4.7: House B - Plotted Overview Results

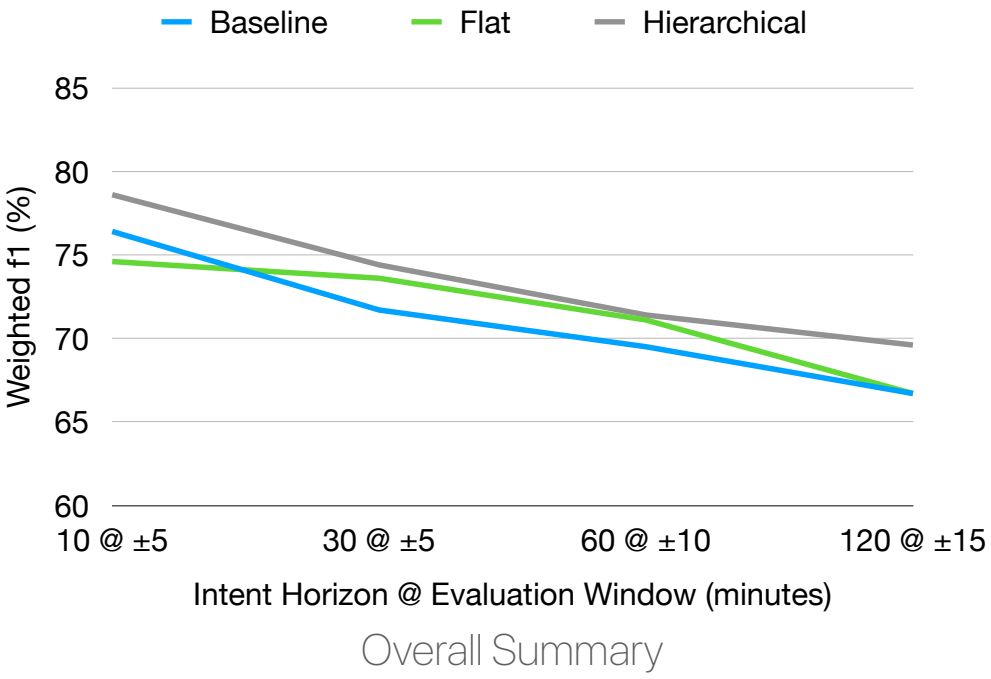


Figure 4.8: House C - Plotted Overview Results

Table 4.6: House A - Tabulated Overview Results

Model		Baseline				Bayesian Flat				XGBoost Flat				XGBoost Hierarchical			
Intent Horizon (Minutes)		10	30	60	120	10	30	60	120	10	30	60	120	10	30	60	120
Eval Window (Minutes)	@ ±5	**73.8**	**72.2**	69.6	67.4	32.5	27.8	37.6	42.4	**75.1**	**72.9**	71.4	69.5	**75.9**	**74.4**	71.7	69.2
	@ ±10	75.5	73.9	**71.3**	69.1	34.1	30.3	41.3	46.4	76.8	74.6	**73.2**	71.1	78.6	77.1	**75.3**	72.6
	@ ±15	-	75.4	72.7	**70.6**	-	33.0	44.3	48.1	-	76.1	74.9	**72.4**	-	78.9	77.6	**74.9**

Table 4.7: House B - Tabulated Overview Results

Model		Baseline				Bayesian Flat				XGBoost Flat				XGBoost Hierarchical			
Intent Horizon (Minutes)		10	30	60	120	10	30	60	120	10	30	60	120	10	30	60	120
Eval Window (Minutes)	@ ±5	**78.2**	**77.1**	75.4	71.8	33.4	31.2	32.3	29.4	**84.3**	**80.6**	75.5	72.7	**83.9**	**80.9**	78.4	73.2
	@ ±10	79.7	79.0	**76.7**	73.5	39.5	38.3	41.0	36.9	85.8	82.0	**77.0**	74.3	86.4	83.8	**81.4**	75.6
	@ ±15	-	80.8	77.8	**74.7**	-	43.5	45.8	42.0	-	83.2	78.0	**75.5**	-	85.8	83.1	**77.5**

Table 4.8: House C - Tabulated Overview Results

Model		Baseline				Bayesian Flat				XGBoost Flat				XGBoost Hierarchical			
Intent Horizon (Minutes)		10	30	60	120	10	30	60	120	10	30	60	120	10	30	60	120
Eval Window (Minutes)	@ ±5	**76.4**	**71.7**	66.7	61.3	54.9	51.1	51.1	53.0	**74.6**	**73.6**	68.9	62.7	**78.6**	**74.4**	66.8	62.0
	@ ±10	79.1	75.0	**69.5**	64.2	59.6	56.2	56.0	58.1	77.0	75.9	**71.1**	64.9	81.6	78.8	**71.4**	66.3
	@ ±15	-	77.6	71.8	**66.7**	-	57.8	55.4	60.6	-	77.7	72.9	**66.7**	-	81.9	74.8	**69.6**

4.7.4 Summary of Experimental Results (per house)

4.7.4.1 House A - Results Summary

Both flat and hierarchical approaches out perform the baseline for all intent horizons at all evaluation windows in this house.

In the case of the tightest evaluation window, the flat approach just exceeds the hierarchical approach by 0.3% (69.2% vs. 69.5%). This artifact may be attributed to the tight evaluation window (5 minutes) given the long intent horizon (120 minutes). It is noted that it is only in the tightest evaluation window (5 minutes) that the flat model exceeds the hierarchical performance. In all other evaluation windows, the hierarchical model exceeds the flat model performance. As the reader reviews the results for the other houses, this pattern will become evident where evaluating a long intent horizon under a tight evaluation window reduces the lift observed from the hierarchy though not always to zero.

As for the present house A, across all intent horizons at the 10 minute and 15 minute evaluation windows, the expected pattern is observed consistently; the flat model exceeds the baseline and the hierarchical model exceeds both the baseline and the flat model in performance.

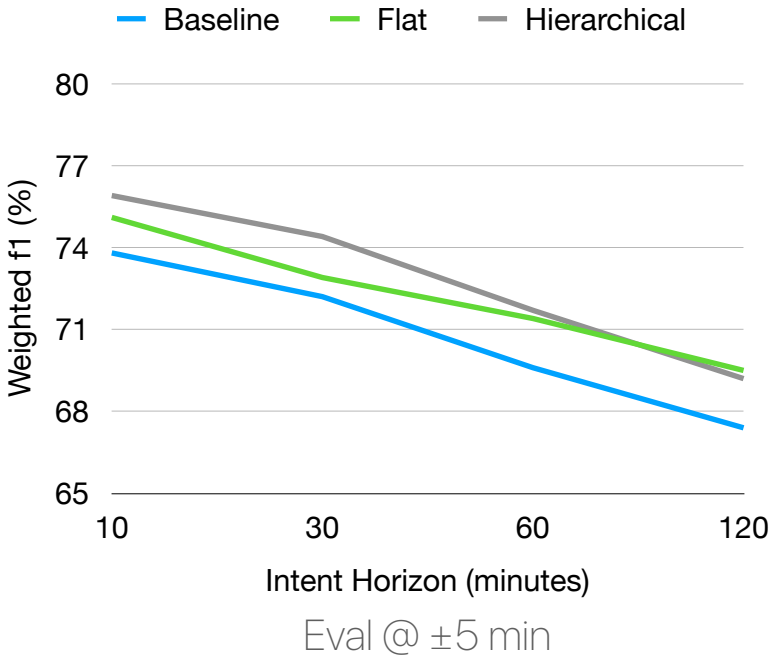


Figure 4.9: House A - Evaluation Window 5 Results

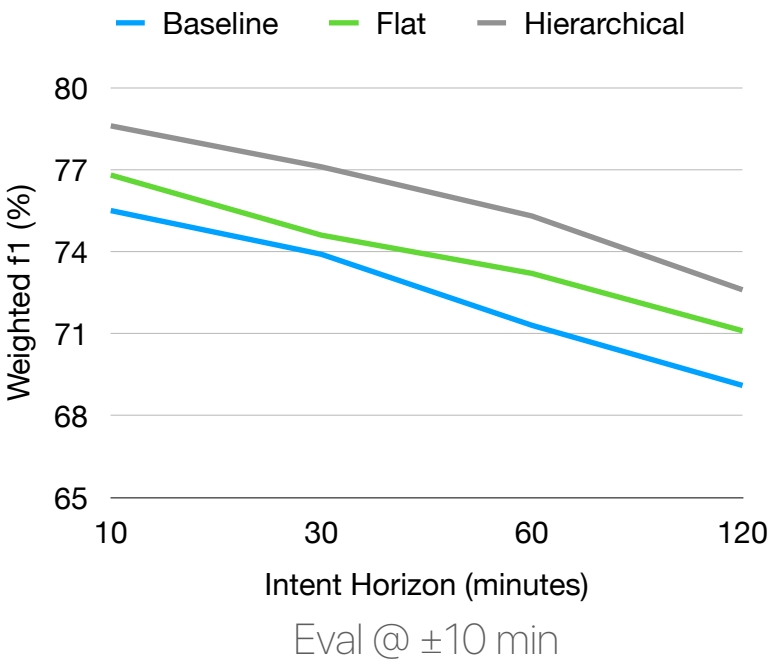


Figure 4.10: House A - Evaluation Window 10 Results

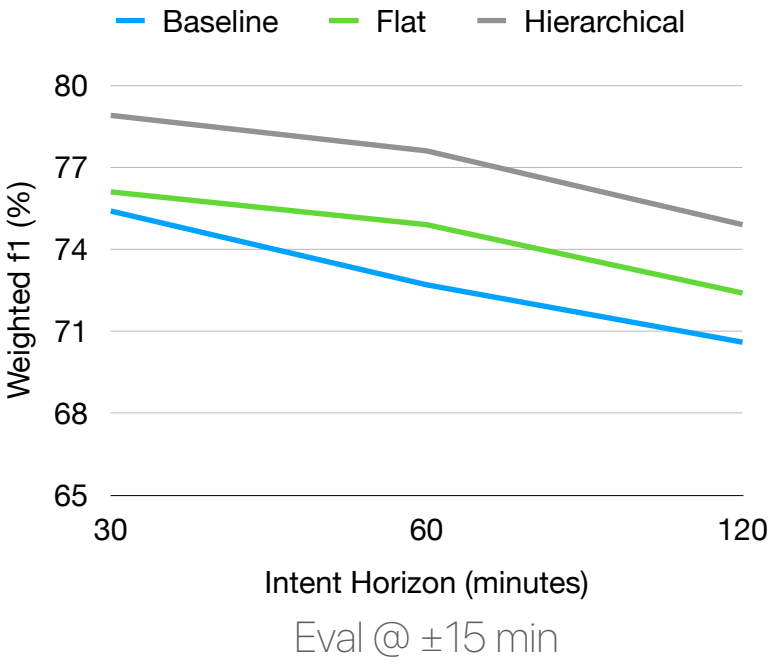


Figure 4.11: House A - Evaluation Window 15 Results

4.7.4.2 House B - Results Summary

For house B, it is again observed that both flat and hierarchical approaches out perform the baseline for all intent horizons at all evaluation windows. The closest any of the models got to the baseline was at intent horizon 60 minutes where the flat model scored very close to the baseline (while still exceeding it).

As noted earlier in the overall summary (section 4.7.3), for the shortest intent horizon (10 minutes) in this house, it is observed that the flat model exceeds the hierarchical approach by 0.4% (84.3% vs. 83.9%). This is attributed to the tight intent horizon where the benefits of the hierarchy is not as pronounced in these near term predictions.

However, for every other data point, including longer intent horizons at 5 minute evaluation window and all intent horizons for the 10 and 15 minute evaluation windows, the hierarchical model exceeds the performance of the flat model. In particular, it is observed that as the intent horizon extends from 30 minutes to 60 minutes, the performance of the flat model drops across all evaluation windows while the hierarchical model sustains its performance. At the 120 minute horizon, the hierarchical model does drop in performance but still fares better than the flat model.

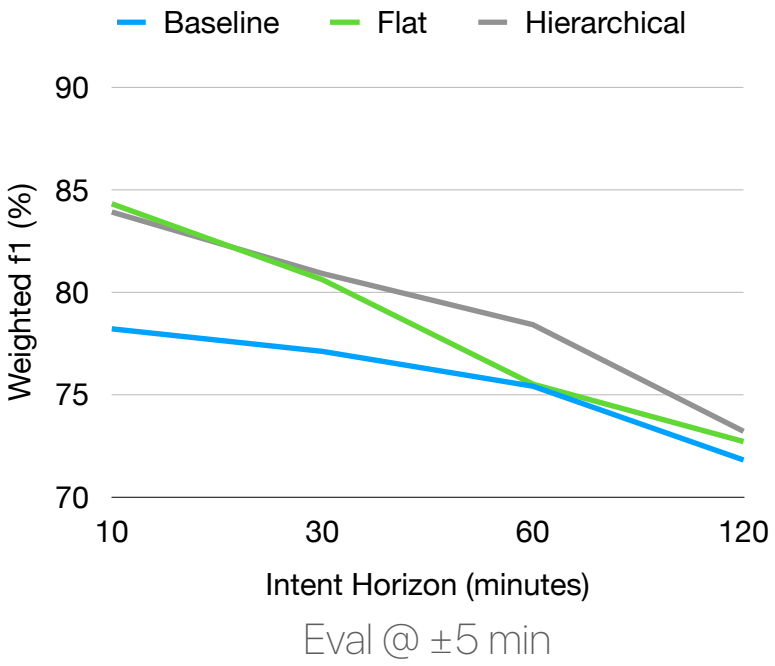


Figure 4.12: House B - Evaluation Window 5 Results

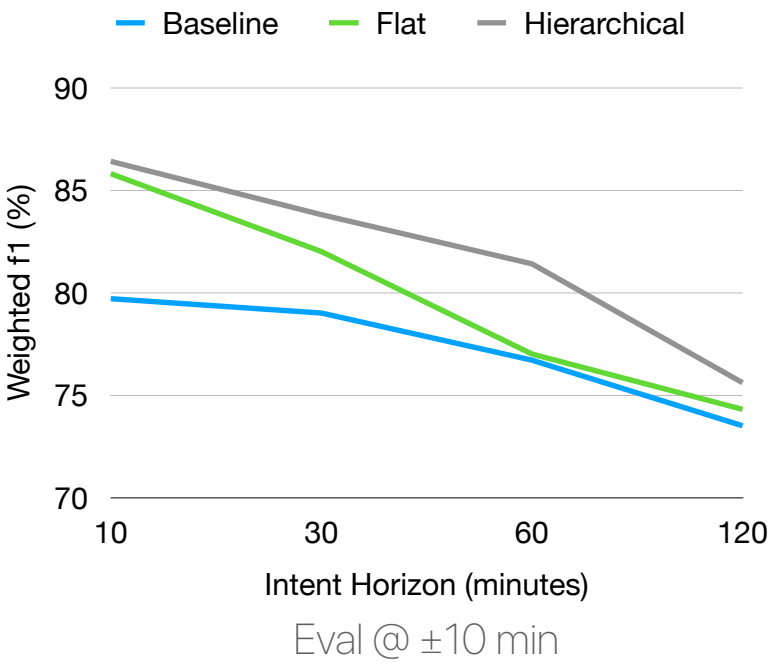


Figure 4.13: House B - Evaluation Window 10 Results

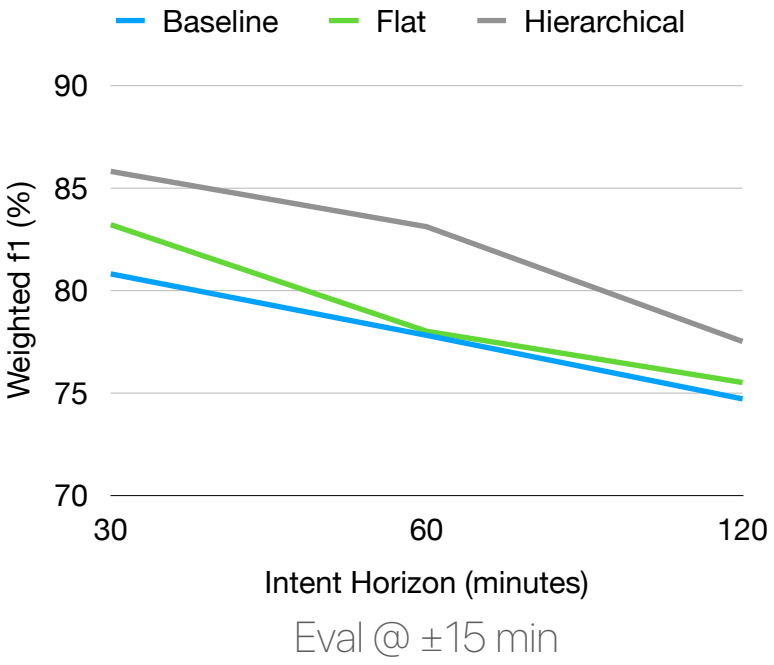


Figure 4.14: House B - Evaluation Window 15 Results

4.7.4.3 House C - Results Summary

In the case of House C, the hierarchical approach consistently out performs the baseline for all combinations of intent horizon and evaluation window. The closest the two got was for the 60 minute intent horizon with 5 minute evaluation window where the baseline is at 66.7% and the hierarchical model records 66.8% performance.

At the tightest evaluation window (5 minutes), it was observed in this house that the hierarchical model out performs the flat model for the shorter horizons however the flat model exceeds the hierarchical model at the longer intent horizons. This is in line with trends observed in the other houses where employing the hierarchical model with an extremely tight evaluation window at a long intent horizon masks the benefits of the hierarchy.

For the shortest intent horizon (10 minutes) it was observed that the flat model under performs even the baseline for two out of the three evaluation windows (5 minutes and 10 minutes) while barely breaking even on the longest evaluation window (15 minutes). In contrast, from evaluation window 10 minutes and up, the benefits of the hierarhical model are evident and the hierarchical model out performs both baseline and the flat model for all intent horizons at the longer evaluation windows.

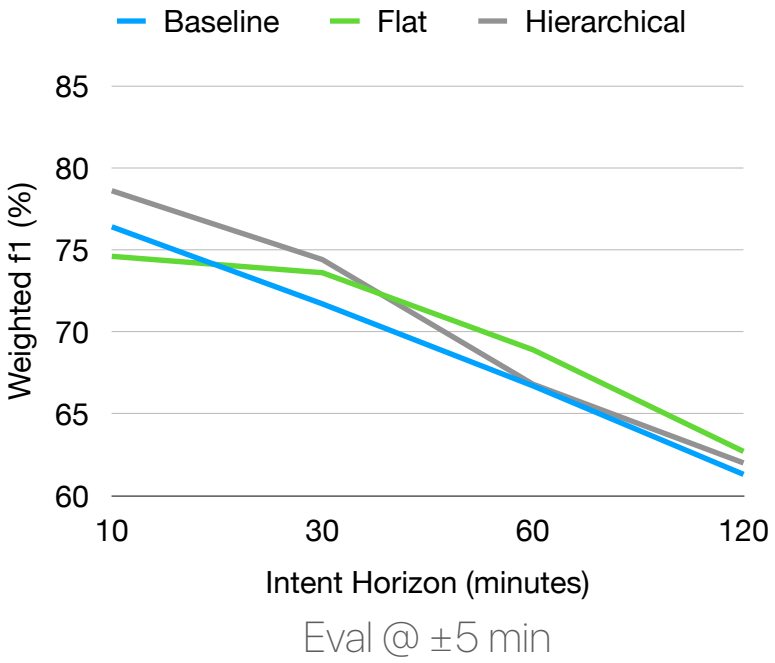


Figure 4.15: House C - Evaluation Window 5 Results

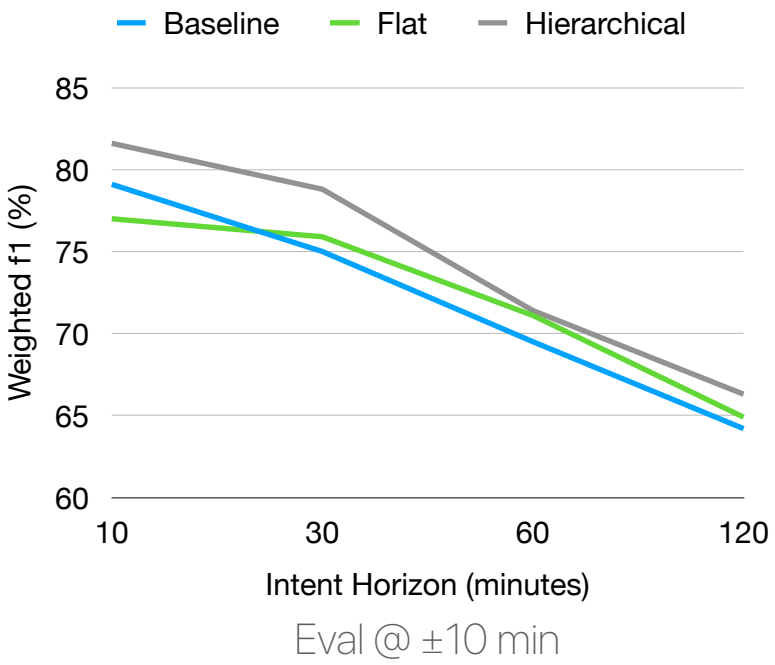


Figure 4.16: House C - Evaluation Window 10 Results

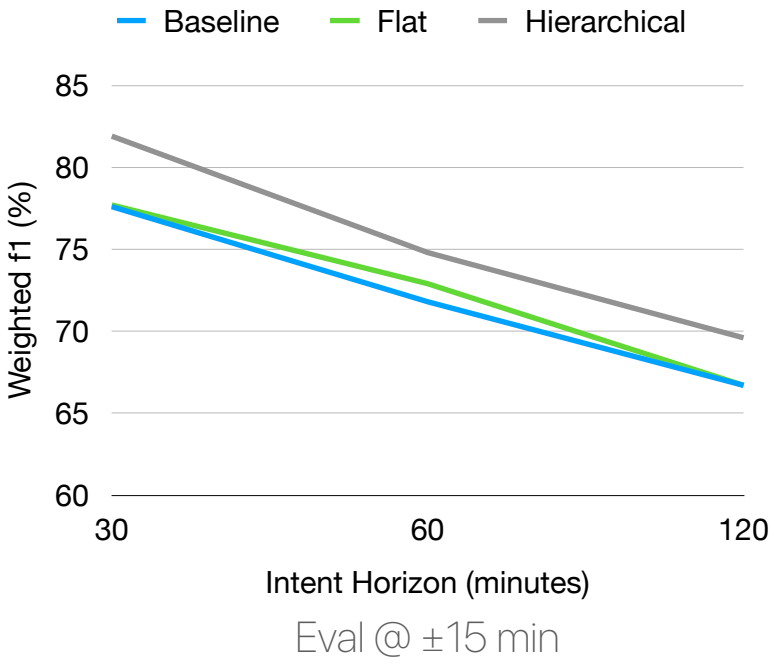


Figure 4.17: House C - Evaluation Window 15 Results

4.8 Conclusion

In this chapter the hierarchical framework was applied to the problem of predicting activities of daily living in a smart home. After instantiating the framework and defining the terms action, activity and intent within this context, a series of experiments were carried out to validate the performance of the framework. All experiments used the same time discretisation (60s) as the benchmark paper [162]. The evaluation metric used was the weighted average of f1-score across all the activities and across all the cross validation folds.

The initial experiment established a baseline to compare other methods against. The baseline was defined as the current activity predicted by the model is the intent of the resident (i.e.: it was assumed that the intent will continue with whatever they are doing right now until the intent horizon). The second and third experiment compared to model types in a flat formulation, namely, a probabilistic model and a tree-based XGBoost model. These experiments established the superior performance of the XGBoost model for this data set.

Given these results, the fourth and final experiment was setup; the best performing flat model (XGBoost) was re-evaluated, this time in a hierarchical formulation. The original XGBoost experiment then became an ablation study to compare against and establish the value of the hierarchy.

All the intent prediction experiments were run for 4 intent horizons (10, 30, 60 and 120 minutes) and 3 evaluation windows (5 minutes, 10 minutes and 15 minutes). All feasible combinations of intent horizon and evaluation window was evaluated. For example, a horizon of 30 minutes and window of 5 minutes evaluate whether the resident acted on their predicted intent between 25 and 35 minutes from now. An infeasible combination is a 10 minute horizon with a 15 minute window.

The experiments showed that in general, the hierarchical approach adds value and exceeds the flat model and both of these approaches exceed the baseline. Some trends observed in the experiments were firstly on experiments with a long intent horizon and very short evaluation window (for example, a 120 minute horizon with a 5 minute window). In this scenario, the flat model and hierarchical model converges and on occasion the flat model exceeds the hierarchical model. In total 36 data points were analyzed per house and it was observed that on only 3 of those data points did the flat model exceed the hierarchical model. On those 3 occasions, the flat model exceeded the hierarchical model by 0.3%, 0.4% and in the third case 2.1% (the third number was observed in house C).

Reviewing the full set of results, it became apparent both from the experimental results and logically evaluating the problem that shorter intent horizons pair better with shorter evaluation windows and longer intent horizons match with a longer evaluation window to account for the increased uncertainty of the second case. Based on this, a set of logical combinations for intent horizon and evaluation window were devised and plotted. These results established in summary that for an intent horizon longer than 30 minutes, the hierarchical model will out perform both the flat model and the baseline model consistently.

Before concluding the chapter, however, the quality of the data available in this implementation example bears discussion. The quality and volume of the data set available for this implementation example was noticeably lower than the other two implementation examples. For instance, both implementation example II (access control, chapter 5) and implementation example III (vehicle destinations, chapter 6) recorded a full years worth of data whereas in this example, only a few weeks of activity and sensor logs were available. The data that was available suffered from misaligned labelling where the sensor logs were captured by an automated system but the activity logs were human annotated. These data challenges can be attributed to the privacy and logistical concerns of recording human activity.

For whichever reason, these challenges did exist and care had to be taken in applying the framework to this data set. For instance, the clusters for the hierarchy was established manually and some models like probabilistic models which worked very well in the other implementation examples were not feasible for this use case. This meant that probabilistic models had to be abandoned in this example in favour of tree based XGBoost models despite sacrificing the inherent advantages of probabilistic models such as uncertainty quantification. It should be noted however, that despite these changes, the central tenets as laid out by the framework (defining action, activity and intent, creating an abstraction layer to establish higher level groups, a hierarchical approach in the modelling phase, a separate model at the higher level and for every group at the lower level and train/test split evaluation) all still applied here. Therefore, the framework proved a sufficient blueprint for a machine learning practitioner to analyze this noisy and sparse dataset to generate meaningful results for intent recognition.

In conclusion, given the observations above on both the data set and the experimental results, there is strong evidence to accept the benefits conferred by the hierarchical aspect of the intent recognition framework. It is also established that the results from the framework out perform the baseline results.

CHAPTER 5

Implementation Example II: Predicting access requests

5.1 Introduction

In this chapter, the intent recognition framework is applied to access control data of employees working in an office building. The intent of employees in this implementation example is to gain access and move into another zone of the building. This intent is modelled as the next reader at which they swipe their access card (figure 5.1).

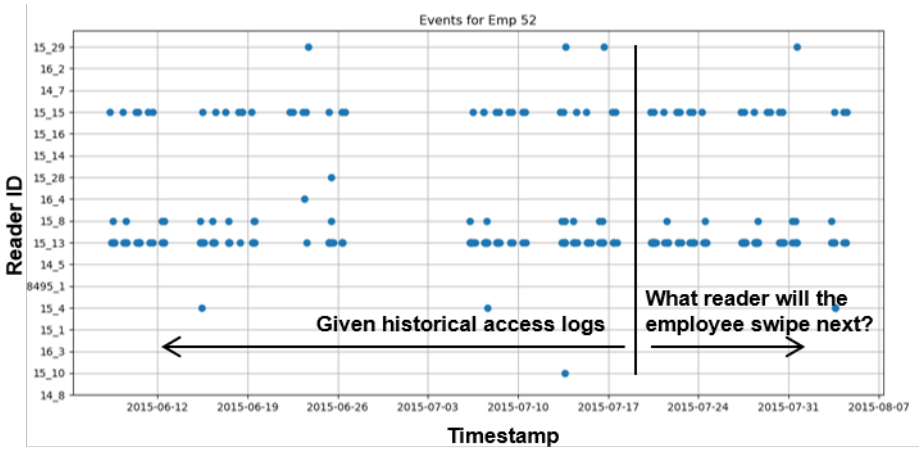


Figure 5.1: Modelling the intent of an employee in an office building

Understanding employee intent in this manner has several benefits in a secure access control context. These benefits include identifying common patterns of movement by employees inside a building such as popular routes taken between two points. This in turn provides valuable information for improving the layout of the building. There are also security applications for having a model of expected employee intent. Namely, suspicious behaviours by employees (such as an unexpected access swipe at an odd time into a sensitive location of the building) can be detected automatically. The advantages can also be more commonplace and immediate such as quicker identification of a faulty reader because the pattern of employees using that reader has changed drastically within a short period of time. Understanding where a person will be in the building also has implications for seamless access and occupancy prediction.

In applying the proposed framework, the following steps are carried out:

- Split the data using some days as training data and remaining as test data.
- Apply a manifold learning technique (in this case t-SNE) to learn clusters of employee IDs in the test data.
- Build a Bayesian model structure for the data.
- Fit conditional probability tables for the model structure using test data.
- Evaluate the fitted Bayesian model on test data.

5.2 Data description

The data set is an access log of all the employee access card swipe events in an office building. The building is split across multiple levels and has several common areas that staff work in. A feature of the particular building is the presence of both entry and exit readers. The data available covers 1 year. An initial exploration of the data reveals expected patterns of card swipes in a working office building. Most of the swipes occur during weekdays and very few occur in weekends (figure 5.2).

The number of swipes per hour of day (a proxy for the level of activity in the building) shows a regular pattern. The number of swipes begin to increase from 7am and peaks at around lunch time. Card swipe activity remains high until 6pm when this tapers off significantly (figure 5.3). The building is closed during the night therefore no activity is observed in those hours.

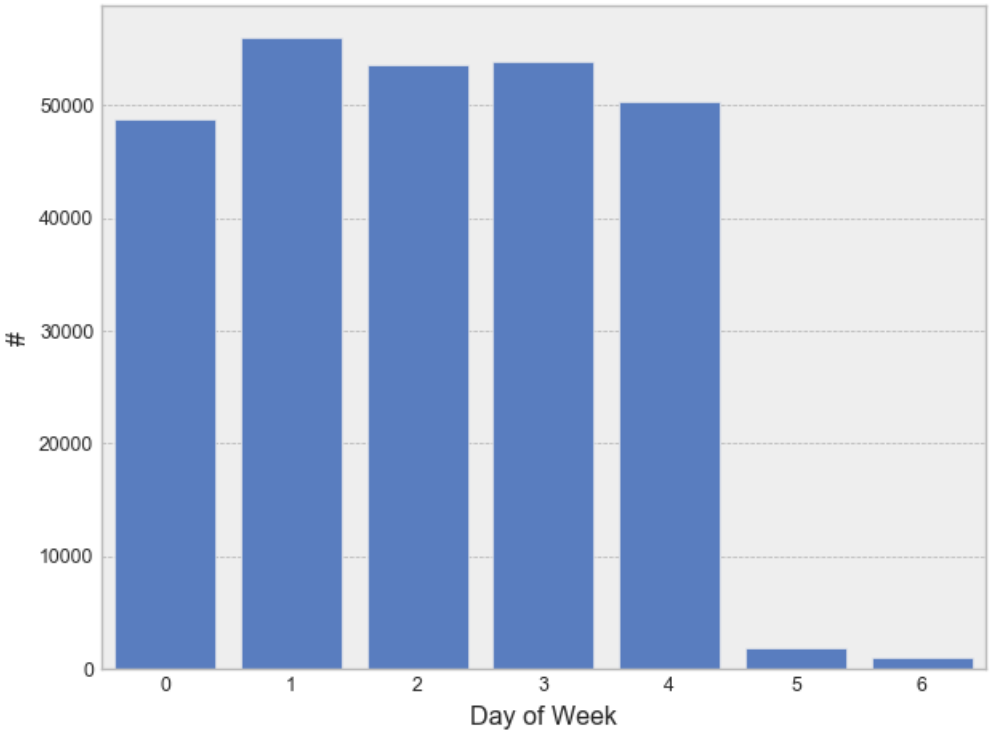


Figure 5.2: Histogram of swipes per day of week

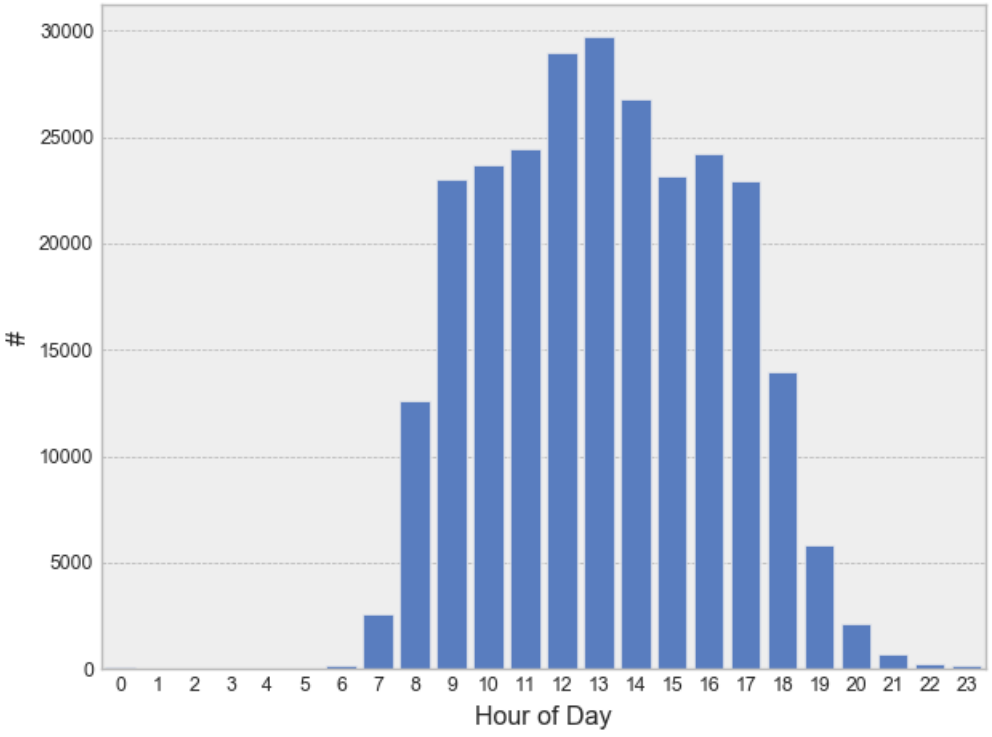


Figure 5.3: Histogram of card swipes per hour of day.

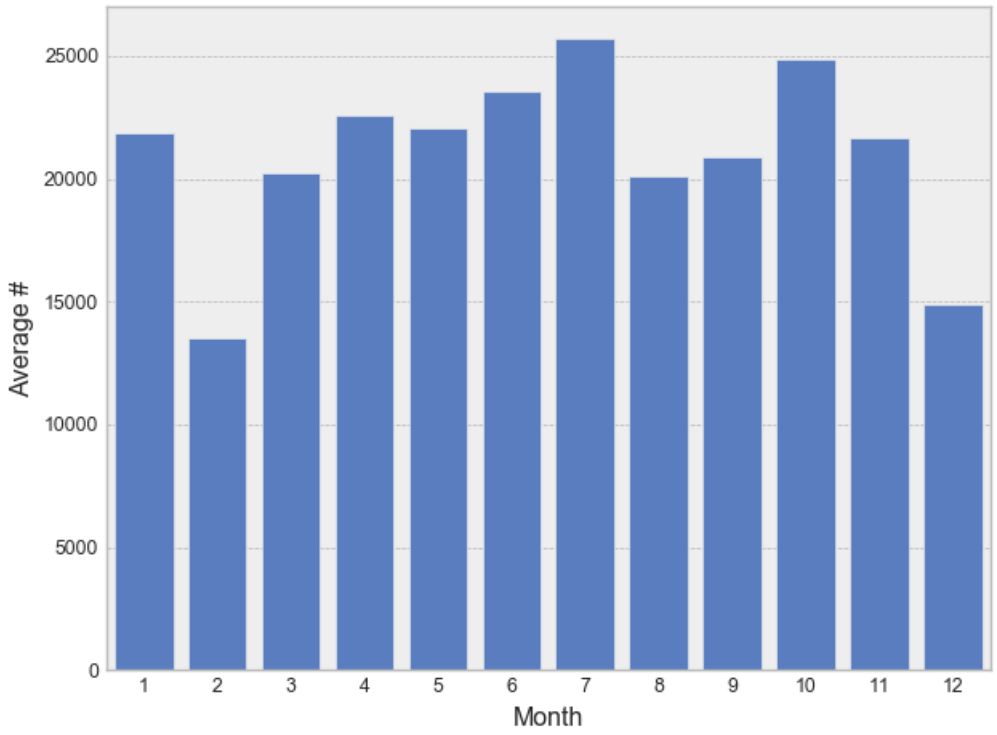


Figure 5.4: Histogram of swipes by month in the year.

A plot of the annual pattern of reader swipes by month is shown in figure 5.4. Some months are busier than others. For example, the month of December shows a marked drop in the activity in the office. This is expected as most of the staff would be away on vacation in the latter part of the month. An interesting pattern is noted in the earlier part of the year. Despite consistently high activity in most months, there is a sudden drop in the level of activity for the month of February. It is not immediately clear what the cause of this drop is, whether it is a one-off occurrence in the year for which the data is collected or a repeating pattern every year. It is possible that this is also associated with February being the shortest month of the year.

Figure 5.5 shows how many swipes per reader were observed in the data. We see 4 readers which are much more frequently swiped than others. There are another 4 which are secondary in terms of frequency. These patterns could be utilized to predict the intent of the employee.

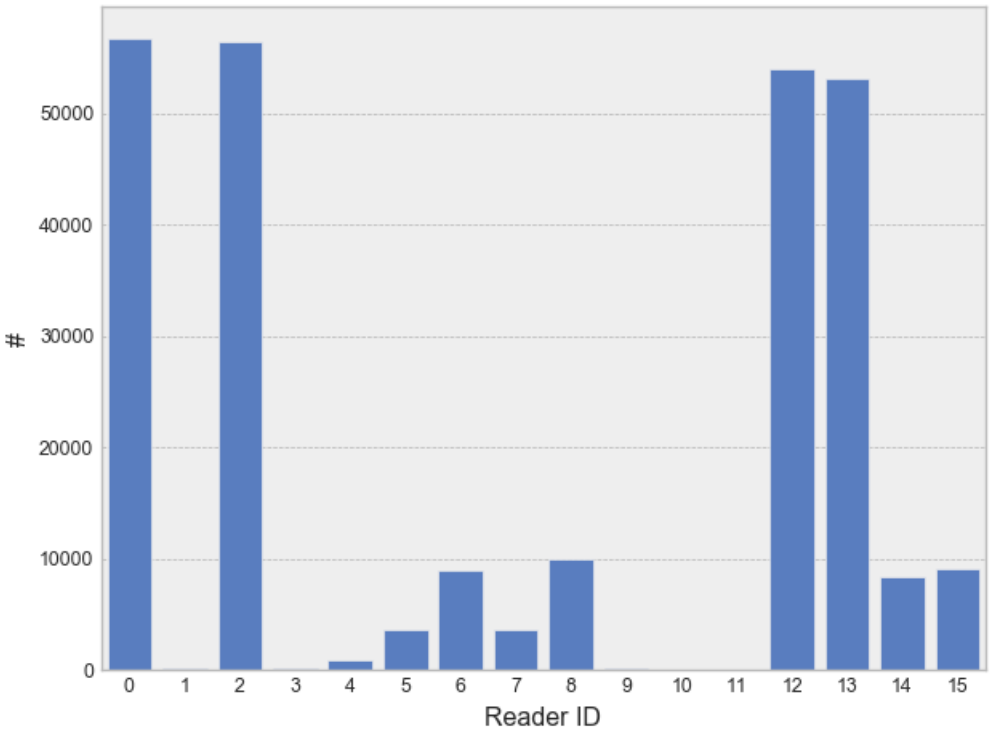


Figure 5.5: Histogram of swipes by reader

5.3 Data pre-processing

The raw data available is an event log of card swipes at readers fixed throughout the building. The framework proposed applies manifold learning techniques to reduce the dimensions of the input data. The purpose of this step is to learn latent structure in the data and allow clustering to establish meaningful groups. These groups are used to create the levels in the hierarchy. Another benefit of understanding this latent structure for the predictive model is allowing it to distinguish patterns even from imbalanced classes (explored further in the experimental results section 5.7 where a hierarchical model is compared to a flat model in an ablation study which proved that the hierarchical approach yields superior performance).

Before the clustering process, the data is first split into training and test data sets. The training/test split is carried out on the days recorded in the data using a cross-validation approach; 30 consecutive days are selected as test data and remaining 322 days are used for training in each fold. The final results are averaged over all the folds. The training set is used to calculate the two dimensional embedding.

5.4 Data embedding

The purpose of the embedding is to reduce the dimensionality of the original data (often high given the number of features observed in the environment of the agent). The embedding procedure therefore reduces spurious correlations being detected due to curse of dimensionality [22].

As prescribed in the framework, the embedding step is a precursor to clustering. The framework directs the practitioner to cluster along a dimension of interest that is expected to yield the most informative sub-groups (see section 3.3). For every use case, therefore, an independent decision is to be made on the dimension of clustering most applicable to the particular problem at hand.

The approach adopted in this implementation example for calculating the embedding is to cluster on employees rather than on activities as was the case in implementation example I (activities of daily living, chapter 4). In this particular case, clustering on employees appears to be the most informative. In a security focused application it is more advisable to model behaviour of employees and understand similar groups among employees. Employees from the same department would likely adopt similar routes through the building and their intended trajectory when swiping at a particular reader may differ from colleagues in a different department swiping at the same reader. There is no sound rationale to believe there will be significant routes in a building that are common across everybody (save for the main thoroughfares).

In contrast, clustering on the readers would yield information on similar readers but lack the context on the person swiping and how this impacts their immediate next behaviour. Furthermore, an employee clustering would set the system up for a new employee joining the company. Their individual behaviour may not be known yet but their department or job function would be known. Therefore, an employee clustering would help provide early indications on the expected behaviour of the new employee by comparing against their more seasoned colleagues in the same cluster. For these reasons, the clustering (and therefore the embedding) in this implementation example is calculated on the basis of grouping similar employees.

To calculate the embedding, then, as a first step, a matrix of counts is calculated where the rows represent employee IDs. The columns represent readers in the building. An entry (i, j) in this matrix represents the number of times employee i swiped their access card on reader j.

The counts are normalized by dividing every row with the number of days that the employee has data recorded. This normalization step accounts for the fact that different employees record different number of days in which swipe activity is recorded during the year. This reflects the fact that some employees are permanent employees whereas others are seasonal employees (or contractors) who only work on site for few months of the year. After normalization, each entry (i, j) in this matrix can be interpreted as the average number of times *per day* each employee i swiped their access card on reader j over the year for which data is recorded.

The normalised matrix of counts described above was then used to calculate a 2-dimensional embedding using the *T-distributed Stochastic Neighbor Embedding* (t-SNE) technique [158].

More concretely, given that the data records 184 employees and 16 readers, a matrix is first established of size 184×16. Entry (i, j) in this matrix is a frequency count of how many times employee i swiped on reader j. This matrix is then passed to the embedding algorithm which produces a new matrix of size 184×2. This is now a 2 dimensional representation of the data in the original larger matrix. The 2 dimensional representation can be visualised (as shown in figures 5.6 and figure 5.8). Each point on these plots represent an employee. The clustering algorithm is then applied to this 2 dimensional representation which yields clusters or groups of employees. The approach is driven by the swiping behaviour of employees in the building which is a suitable metric given the goal is to model the intent of employees in this implementation example.

5.5 Clustering of employees

The embedding which results from the procedure described in section 5.4 is then clustered using the KMeans algorithm [105].

Similar to the previous implementation example (activities of daily living, chapter 4) silhouette analysis [138] was used to determine the best value of number of clusters k as described in section 3.5. In this implementation example the range of possible k is selected to be between two and the number of employees recorded in the data.

The number of clusters in practice can vary (with k selected to maximise the silhouette score). Depending on the way that data is split into training and testing sets, the exact assignment of employees to clusters can also vary.

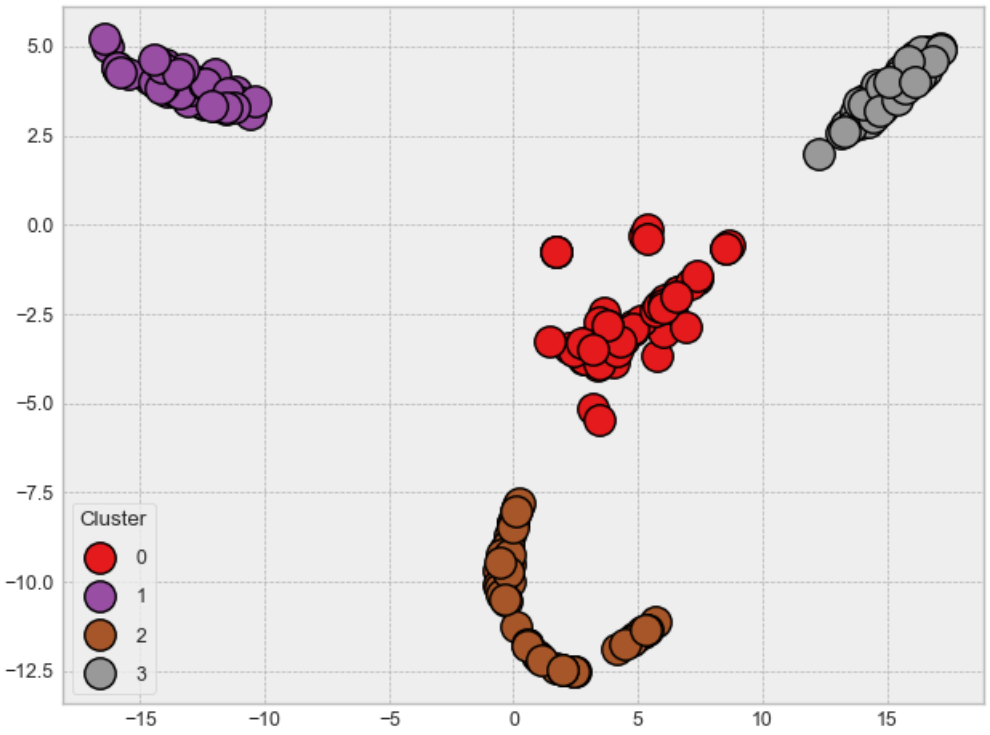


Figure 5.6: An embedding calculated on a training subset

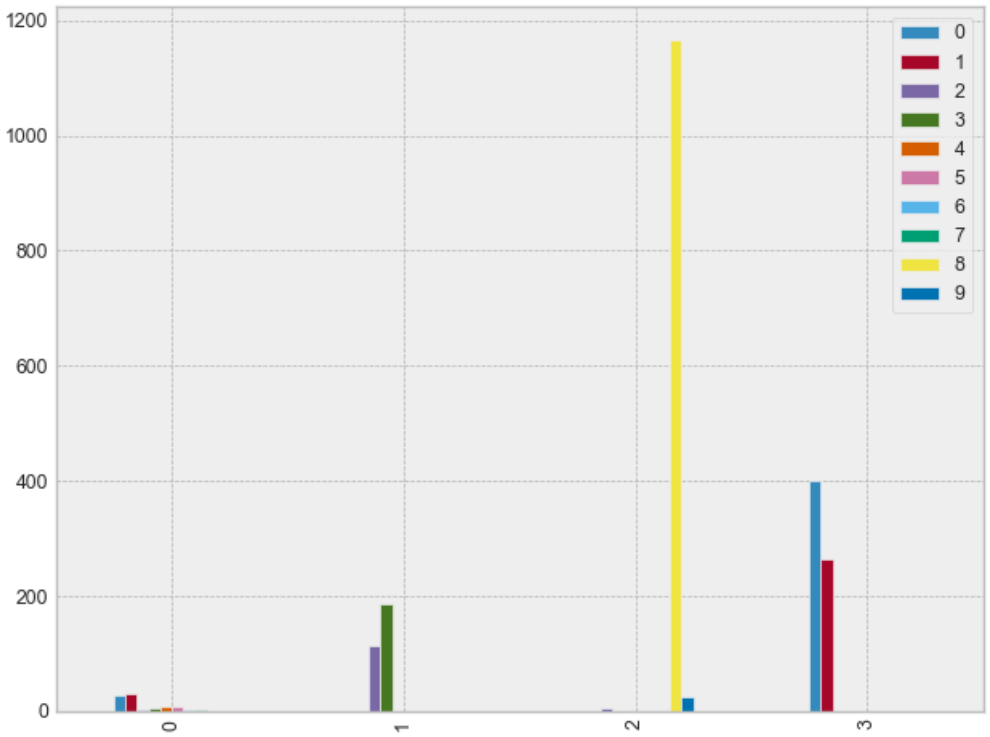


Figure 5.7: Number of swipes per reader in each cluster for the embedding in figure 5.6, the colour of the bars represent distinct readers

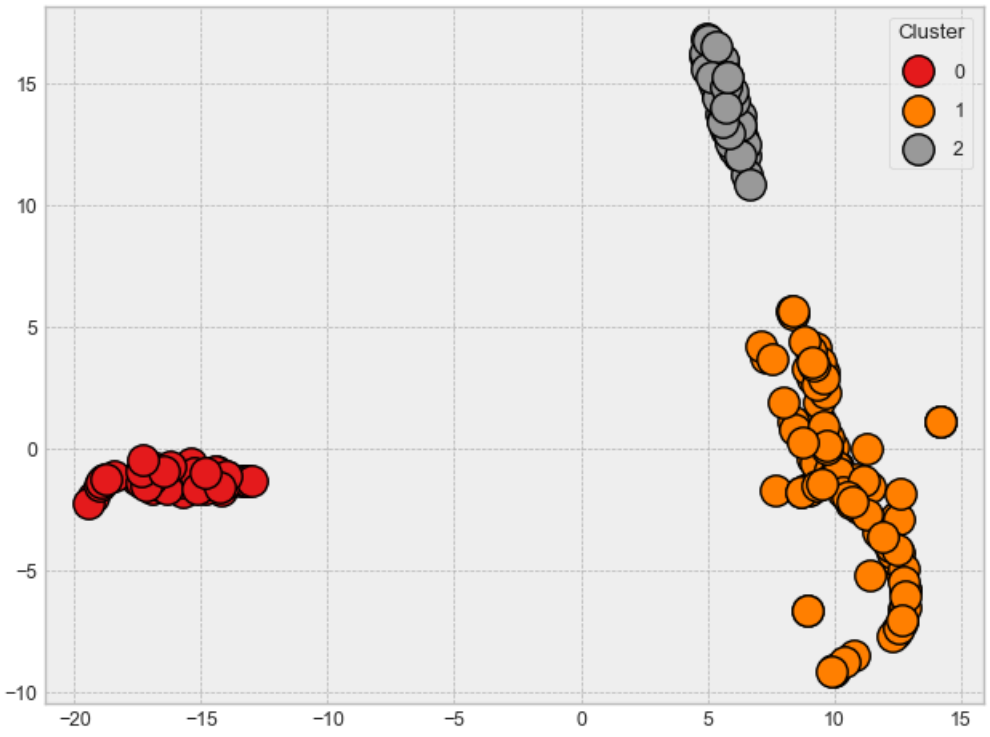


Figure 5.8: An embedding calculated on another training subset

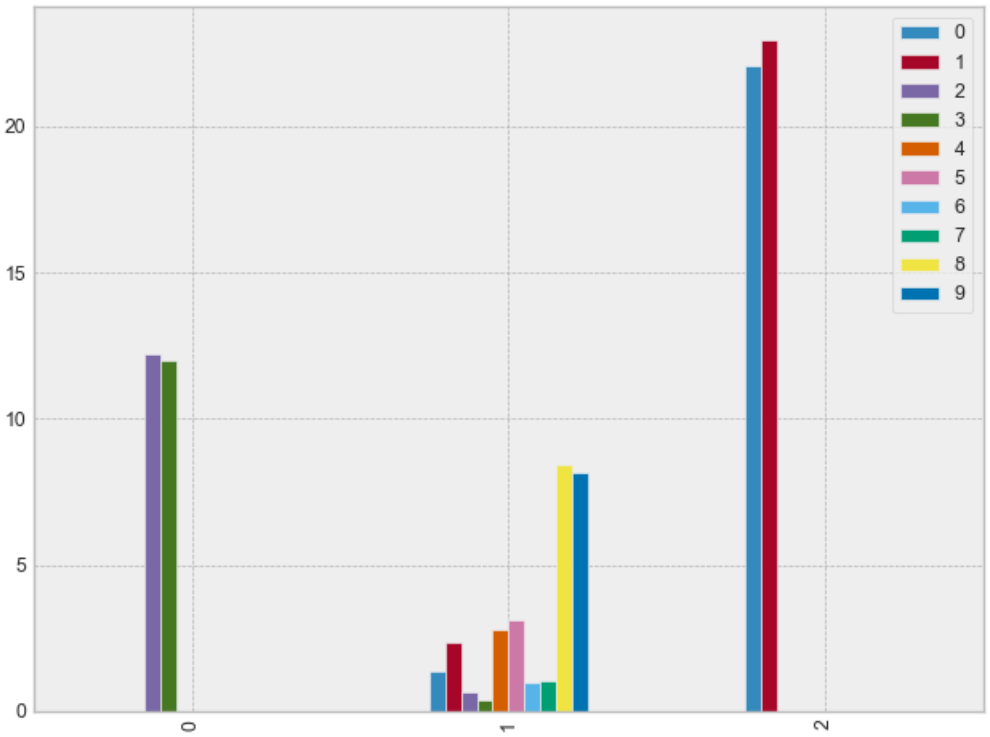


Figure 5.9: Number of swipes per reader in each cluster for the embedding in figure 5.8, the colour of the bars represent distinct readers

Two possible outputs for the clustering procedure are shown in figure 5.6 and figure 5.8. Figures 5.7 and 5.9 show the number of swipes recorded for the individual readers grouped by the clusters they fall in (the colour of the bars represent distinct readers recorded in that subset of the data). The two embeddings are calculated on random subsets of the training data and will form part of the cross validation to calculate final results. Every point in these figures represents a unique employee ID. The technique yields 3 or 4 clusters for this application depending on the subset of the data used as the training set. The experiment results presented later in the chapter are averaged across a 12 fold cross validation to account for such variations.

A visual inspection of the embeddings generated (figures 5.6 and 5.8) indicate that the data set in this implementation example is of a sufficiently high quality to produce very clearly separated distinct clusters. This observation on the clear separation of the clusters along with the approach of favouring the most simple algorithm that yields desired results (as described in section 3.5) indicate that KMeans algorithm is a suitable choice for this implementation example.

A further visual inspection of the figures showing the number of swipes per reader in each cluster (figures 5.7 and 5.9) confirms the distinction between these clusters.

5.6 Framework instantiation

In this section, the procedure to instantiate the general framework described in chapter 3 for this specific example is described.

5.6.1 Definition of *action*, *activity* and *intent*

In this example, the observed *actions* are the card swipes of the employees. Given the details of this implementation example, *activities* are defined as the most recent card swipe by the employee, a collection of one action (the most recent that is observed).

Intent then is defined as the **next card reader** the employee will swipe.

5.6.2 Hierarchical structure

There are two levels in the hierarchical model. As is the case in all implementation examples of the framework, clusters learnt in the data are utilized as abstractions

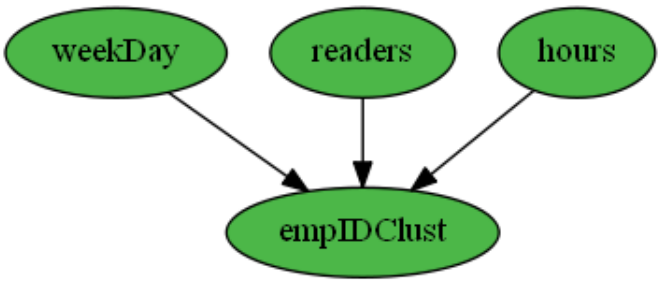


Figure 5.10: Bayesian model from domain knowledge (Upper)

which capture hidden structure in the data. For a description of how clusters are calculated in this particular example, please see section 5.4.

Once the levels are established via clustering, the upper level model is trained on the cluster labels such that it predicts the employee cluster to which the present employee belongs. These predicted cluster labels (at higher level) are propagated to the relevant lower model which predicts the intent of the employee (i.e.: the next location in the building where this particular employee intends to swipe). The relevant lower model is selected based on the cluster predicted at higher level.

More detail on the models are provided in the following sections.

5.6.3 The model

The upper level model predicts the cluster to which the present employee belongs. If the employee ID in the test case is one that is already observed in the training data, their cluster assignment is already known and may be reused. On the other hand, if the employee is previously unseen by the training algorithm, then the upper model is used to predict the cluster to which the employee belongs.

5.6.3.1 Upper model

The structure of the upper model is shown in figure 5.10. The target variable for prediction is *empIDClust*.

5.6.3.2 Lower model

The domain knowledge structure is specified based on common sense relationships between the features of the data set. The structure of the lower model is shown in figure 5.11. The target for prediction is the *readers* variable which indicates the next reader the employee will swipe.

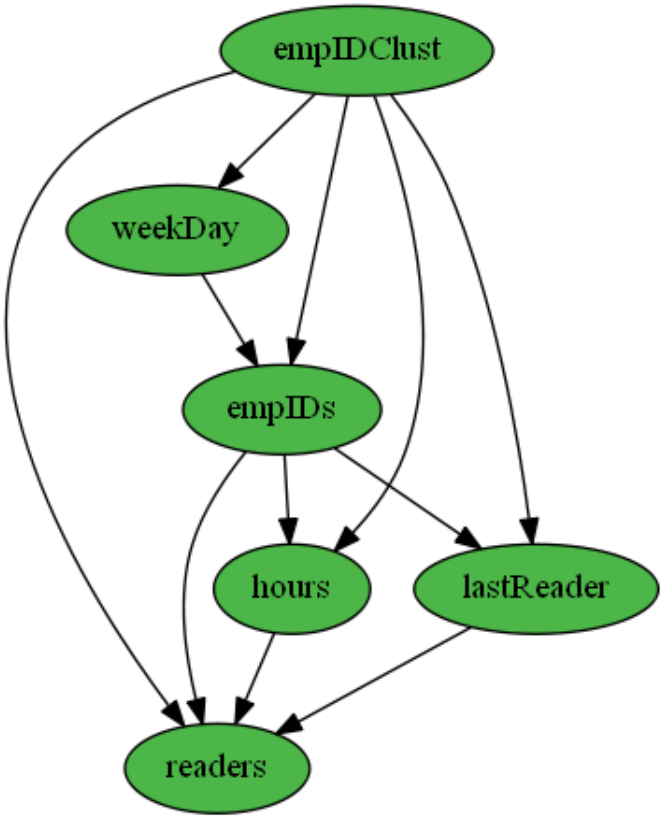


Figure 5.11: Bayesian model from domain knowledge (Lower)

The *empIDClust* variable influences all other variables in the lower model so as to ensure the hierarchical aspect of the framework. The effect of this structure on the conditional probability tables calculated is to induce a conditional dependency of all other variables on the employee cluster.

5.7 Experiments & results

In this section, the results for the experiments that were run to evaluate the framework on this application are presented.

5.7.1 Experiment description

The performance of the proposed framework on this application is evaluated by first splitting the full data set into training and test sets. The split is made on the days recorded in the data. There are 352 days of data recorded in the full data set. In every split, 322 days are selected as the sample of data to fit the model (training set) and the remaining 30 days are selected to provide an unbiased evaluation of the model (test set). There are 12 non-intersecting folds

possible in this scheme and the overall average of all splits are reported.

The metrics used to evaluate the model in this implementation example are *precision*, *recall* and *f-score* which are described in section 2.4.1.

Two sets of experiments were carried out. The first (and main) experiment employs a hierarchical formulation with Bayesian models at the upper and lower level. In addition, a benchmark experiment was also run. In this second experiment, there is no hierarchy and the prediction is carried out with what is effectively a single clustering layer.

The second benchmark experiment serves two purposes. Firstly, it is an ablation study to establish the value of the hierarchical step proposed in the framework presented in this thesis. Secondly, it is also an evaluation of the proposed method against a simpler benchmark. As this is a proprietary data set, there are not publicly available benchmarks to evaluate against.

These experiments were run on a Dell Precision workstation. The system had an Intel Core i5-6600 CPU clocked at a base frequency of 3.30 GHz and a turbo boost up to 3.90 GHz. The system also had 32 GB of RAM installed and the operating system was a 64-bit version of Microsoft Windows 10.

5.7.2 Experiment results with hierarchical model

The classification report of the model in predicting the reader that an employee employee intends to swipe for the test data is shown in table 5.1. The confusion matrix for the lower model results are shown in figure 5.12. The shading on the confusion matrix figures are class-wise, i.e., the darkest shading on each row is for the highest count of that particular label rather than a global value across all labels.

5.7.3 Experiment results with flat model (benchmark)

The classification report of the flat benchmark model is shown in table 5.2. The confusion matrix for the lower model results are shown in figure 5.13. As earlier, the shading on this figure is also class-wise.

Table 5.1: Average classification results for hierarchical model

reader	precision (%)	recall (%)	f1-score (%)	support
0	63.8348	78.1375	70.2078	56752
1	0.0000	0.0000	0.0000	217
2	86.3036	91.7850	88.9541	56452
3	40.9975	10.6604	15.1499	194
4	35.9959	14.2608	19.7030	957
5	36.0305	19.8193	24.9949	3598
6	61.5642	44.6417	51.6898	9014
7	79.8403	78.8676	79.1946	3605
8	69.6469	39.6934	50.2605	10006
9	2.7778	0.4167	0.7246	236
10	16.6667	3.1136	5.2083	39
11	0.0000	0.0000	0.0000	4
12	71.9113	70.4320	71.1283	54023
13	90.2650	92.5180	91.3665	53100
14	56.7747	27.3611	36.3574	8328
15	60.4496	42.4392	49.6148	9041
macro average	52.7406	41.9067	44.6553	265566
weighted average	75.2298	75.9363	**74.8529**	265566
accuracy (%)		75.9363		

Table 5.2: Average classification results for flat model

reader	precision (%)	recall (%)	f1-score (%)	support
0	48.1093	59.9125	53.3542	56752
1	0.0000	0.0000	0.0000	217
2	85.2330	91.6268	88.3106	56452
3	43.7434	14.3911	18.5292	194
4	42.4045	11.7877	18.1510	957
5	30.3072	4.4787	7.6787	3598
6	42.7388	36.3428	39.2354	9014
7	79.8245	83.2417	81.4381	3605
8	54.9614	33.1498	41.3143	10006
9	0.0000	0.0000	0.0000	236
10	0.0000	0.0000	0.0000	39
11	0.0000	0.0000	0.0000	4
12	51.3037	43.1860	46.8638	54023
13	90.4011	92.0982	91.2343	53100
14	40.3589	38.6950	39.4145	8328
15	53.9283	49.6976	51.6293	9041
macro average	45.2599	38.1122	39.3720	265566
weighted average	65.5012	66.3145	**65.3425**	265566
accuracy (%)		66.3145		

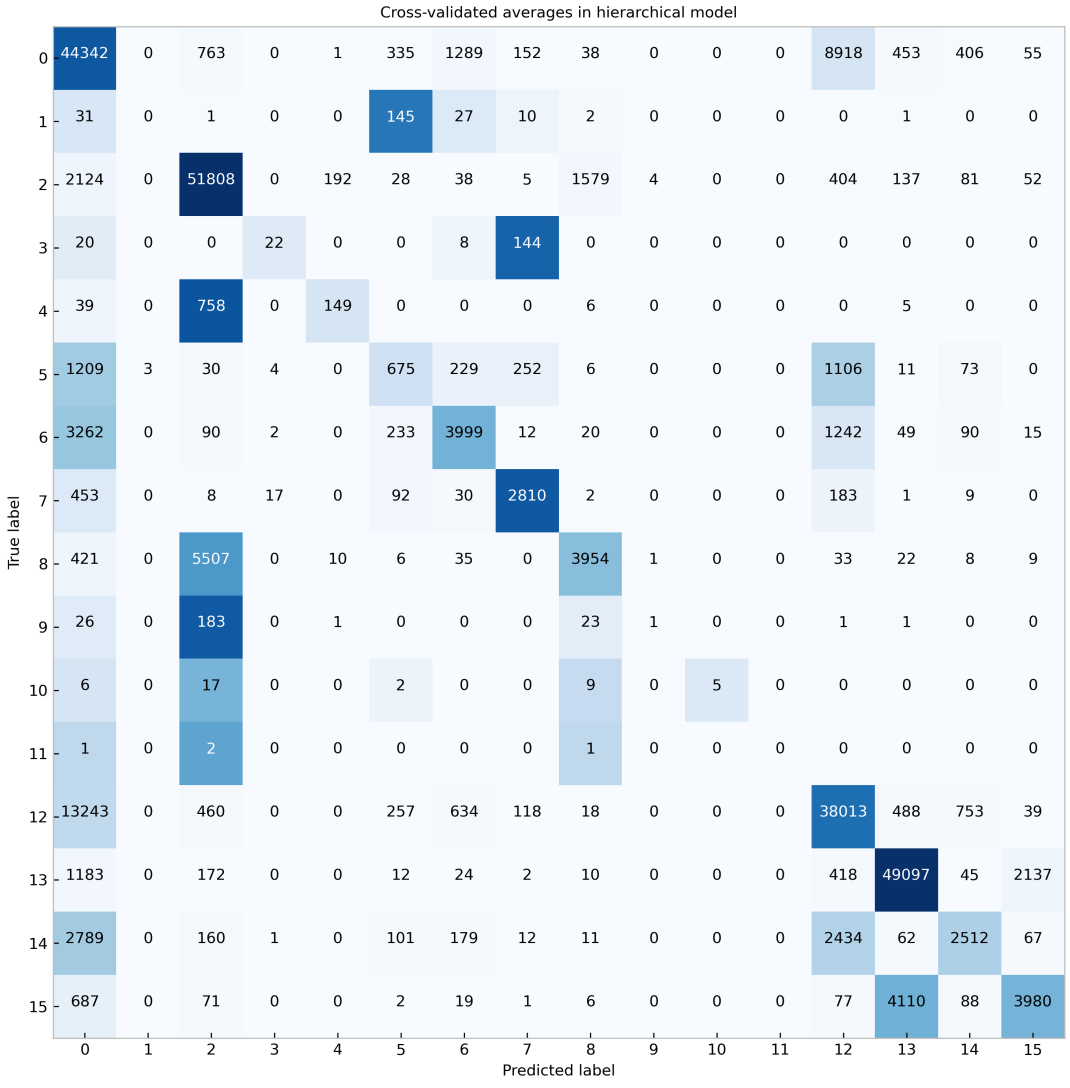


Figure 5.12: Average confusion matrix for hierarchical model, shading is class-wise (the darkest shade on each row represents the highest count for that particular label)

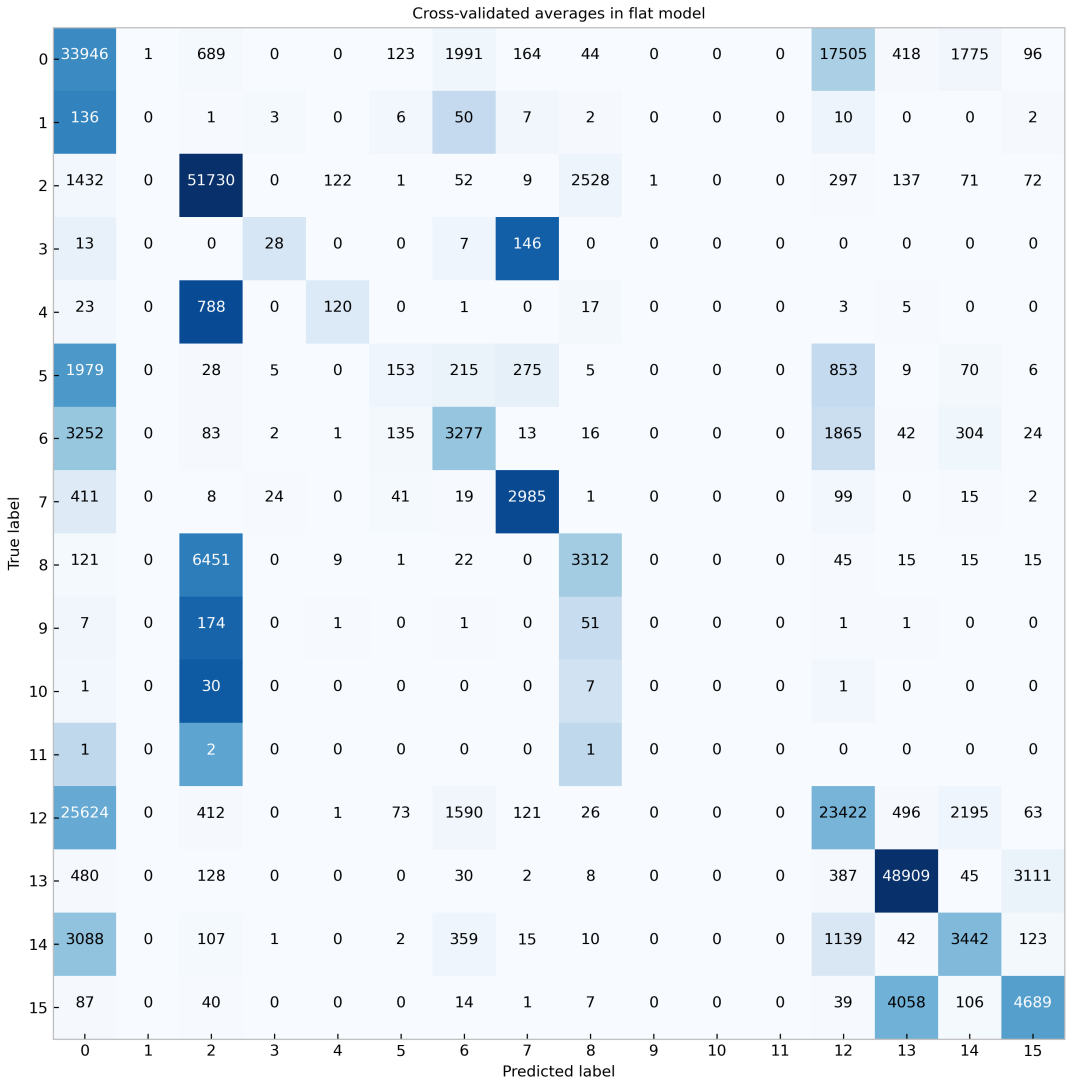


Figure 5.13: Average confusion matrix for flat model, shading is class-wise (the darkest shade on each row represents the highest count for that particular label)

5.8 Discussions & conclusion

The average f1 score of the hierarchical model is 74.9%. This is the weighted average where the individual class f1 scores are averaged while weighting by their frequencies in the data (listed as *support* in the table). The accuracy score for the hierarchical model is 75.9%. Table 5.1 also includes a breakdown of predictive performance by class. The values in the table are averaged across different folds in the cross validation.

Comparing these results with the flat model indicates that performance of the model worsens when the hierarchical aspect is removed (see table 5.2). The flat model reports an average f1 score of 65.3% and an accuracy of 66.3%.

Digging deeper, one also finds that the gain in performance between the hierarchical and non-hierarchical models does not come from purely high frequency classes but also the less frequent ones. For instance, readers 4 and 5 report and f1 score of 19.7% and 25.0% respectively in the hierarchical model. Those same readers report 18.2% and 7.7% in the flat model. Taking a more holistic view, the macro averages, where all the classes are weighted equally, may also be compared between the two models. The f1 score of the model with macro averaging is 44.7% in the hierarchical case but 39.4% in the flat case. This comparison between the hierarchical model and the baseline flat model supports the hypothesis that one of the advantages of the hierarchical aspect in this framework is that the approach handles imbalanced data better with a hierarchy.

An interesting pattern is observed where certain entry readers (for example reader 0 in the anonymisation scheme) are mis-classified as other entry readers in the building. This pattern indicates instances in the data where an employee records two entry swipes back to back which implies a "tailgating" event.

Tailgating is where an employee follows another out of an access controlled area without swiping their own access card first. Detecting such patterns is one example of how the intent recognition framework may be used to detect anomalous behaviors and further enhance an access control system's performance and security.

CHAPTER 6

Implementation Example III: Predicting vehicle destinations

6.1 Introduction

In this chapter, the intent recognition framework is applied to vehicles driving around a city. The intent of the driver is defined as the final destination of their trip.

With rising availability of vast amounts of mobility data, there has been a recent push to develop trajectory mining techniques which achieve greater performance in finding underlying patterns in the data and using that knowledge to aid decision making. The present work uses mobility traces to predict likely destinations of vehicles factoring in time of day patterns and also detecting anomalous behaviour.

The work drew inspiration from [10] where the authors utilise Markov models and probabilistic model checking to characterise user behaviour when using computer software. Our research problem is to develop models for driving behaviour and attempt to make predictions.

A multi-level hierarchical model is proposed. The road network is collapsed into segments. Each segment is considered a state on the Markov model and the state transition probabilities capture the most likely turns drivers take at upcoming junctions. The higher level Markov Chain is a spatial model which captures regions (such as residential areas and downtown areas). The transition probabilities

capture the expected movement from one region to another over different times of the day. For example, in the morning one would expect a lot of traffic from residential areas to downtown office areas whereas at 5pm, the flow is reversed. The lower level model is also a spatial model but at much more fine granularity (for instance transitions between one block to another in a city).

The initial approach was to build distinct Discrete Time Markov chains for each individual driver at the lower level. However, previous research stresses the benefit in sharing information between drivers [10]. In the literature, this idea gave rise to high level Markov models where "classes" of driver are built based on their driving habits. There is a separate Markov Chain for each class of driver with the states being the modes of behaviour within each class and the probabilities capture how drivers transition from one mode to the other. Only the transition probabilities differ between driver classes.

In general, the driving modes in the context of taxis could be on a fare, driving randomly looking for a fare, driving to a specific location (such as a taxi stand at an airport) to pick up a fare, driving to a pre-booked appointment and commuting from area of residence to area of work. Conceptually, having classes of drivers help capture the variations in individual driver behaviour. For instance is a particular driver more likely to drive around randomly after dropping off a fare or return to a taxi stand to locate a new fare?

In the present implementation example, the data available was only for drivers actually on a fare. Therefore the classes of driver represent, for example, drivers who take longer trips across regions of the city or those who restrict themselves to one or two regions only. This behaviour is modelled by building a separate Discrete Time Markov chain for every pick-up and drop-off region pair (more details of the models are provided in section 6.4.2).

With new data streams, the most likely mode of behaviour is inferred given the observed states (positions on the road) and what is known about the class of driver the present subject most closely fits into (based on the pick-up location where the trip originated). Their most likely destination or drop off region is then predicted.

This information feeds into the lower Bayesian model that infers the final destination of a particular driver. This model will have several destination "buckets" the driver is likely to finish at. The probability of each bucket will be constantly updated as the car progresses along the route and more information becomes

available. For instance, if the car passes a particular exit on the motorway, then all destination buckets served by that exit could be ignored.

6.2 Data description

The dataset utilised[1] consists of approximately 1.7 million taxi journeys from 442 taxis operating in the city of Porto in Portugal over a period of one year. The GPS readings are recorded every 15 seconds. Other data such as time of pickup, driver ID and trip ID are also available.

6.3 Data pre-processing

The prediction task was separated into two problems of increasing difficulty. In the first problem, we focus on modelling an area which captures the city centre and surrounding areas of Porto. Any trips that ventured outside this bounding box were removed. The area selected for modelling in problem 1 is shown in figure 6.1. This smaller map covers a distance of approximately 40 km measured diagonally. In the second problem, the area of the bounding box was increased so that it took in Porto and a larger area as far south as Lisbon. This larger box captures the full extent of the trips recorded in the data set. The area covered by the larger bounding box is shown in figure 6.2. This larger map covers a distance of 445 km measured diagonally.

The raw data was cleaned by applying a number of filters. Ten trips that were identified in the data itself as having missing data were removed. However, our data exploration indicated that these were not the only trips with missing data points. To identify these trips, the individual GPS data points were used to calculate the instantaneous speed of travel at every recording made on a trip. It is specified that the GPS data points are recorded once every 15 seconds. Hence, any trips with unreasonably large spikes in the instantaneous speed indicated missing or noisy data. Once a trip is identified as having noisy or missing data, the entire trip was excluded from the data. Secondly, the data is collected by mobile terminals carried by the drivers. Therefore, some "trips" were in fact times where the driver had forgotten to turn off the data collection when outside the vehicle and as such represented pure walking trips. Those trips are often observed to have data points concentrated within a small geographical area such

[1]Data downloaded from
http://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i/

Figure 6.1: Map of area modelled by smaller bounding box.

as a shopping mall. All such trips were identified and removed by excluding any trips that never exceeded a speed of 15 km/hr at any point.

In order to reason about similar trips and locations, we need to discretise the space. Rather than applying a uniform grid, we chose to cluster the GPS data points. This has the advantage that it creates tighter clusters where there was denser traffic hence giving finer predictions in those areas. We used K-Means Clustering at two levels of granularity: a coarse clustering with fewer clusters and a second much finer clustering. The specific values for the number of clusters k at each level were determined by clustering the data for a range of k values and carrying our silhouette analysis.

Silhouette analysis [138] can be used to analyse the separation distance between the clusters as described in section 3.5.

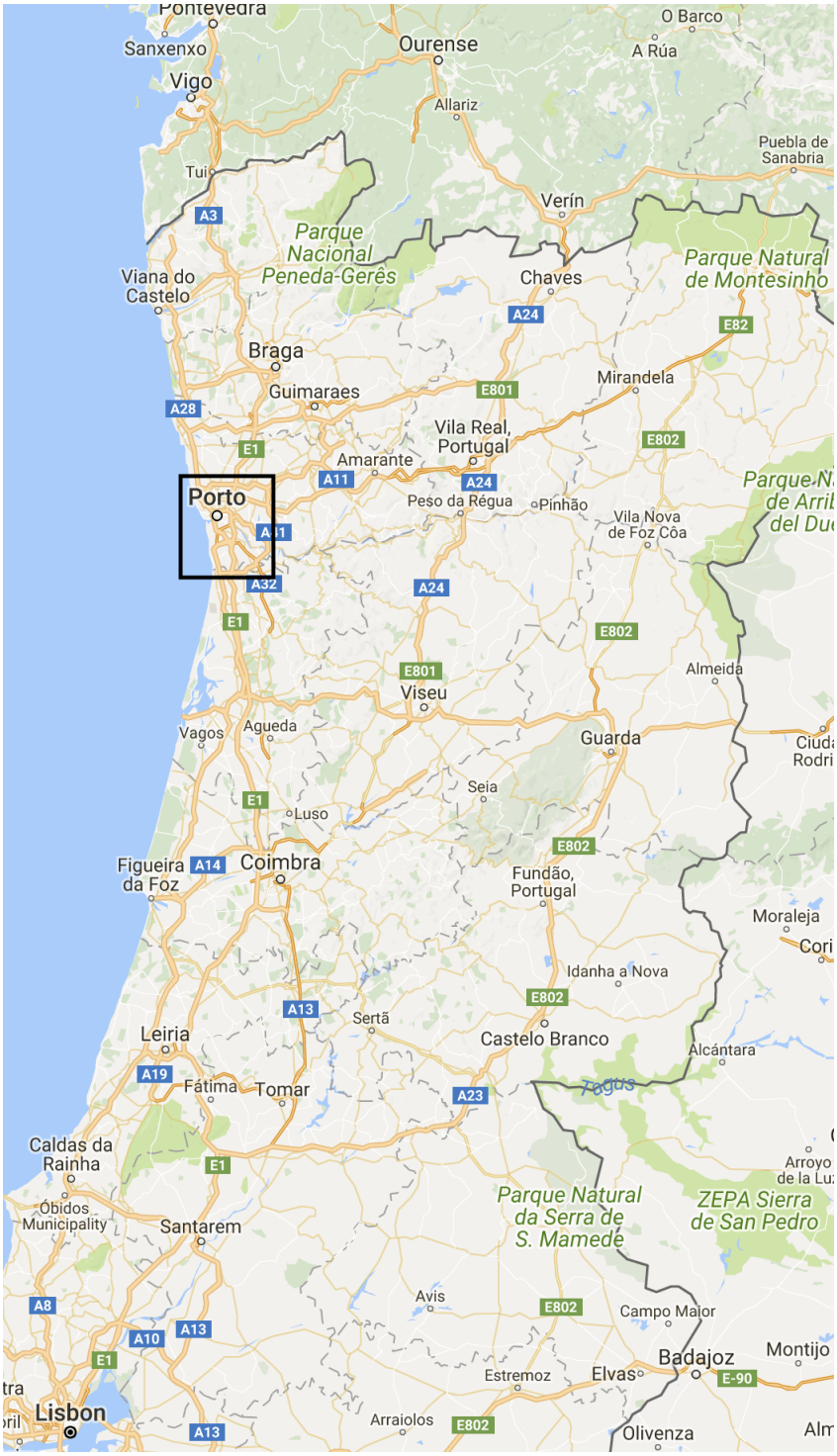


Figure 6.2: Map of area modelled by larger bounding box. Smaller map area shown as the black box in the larger map.

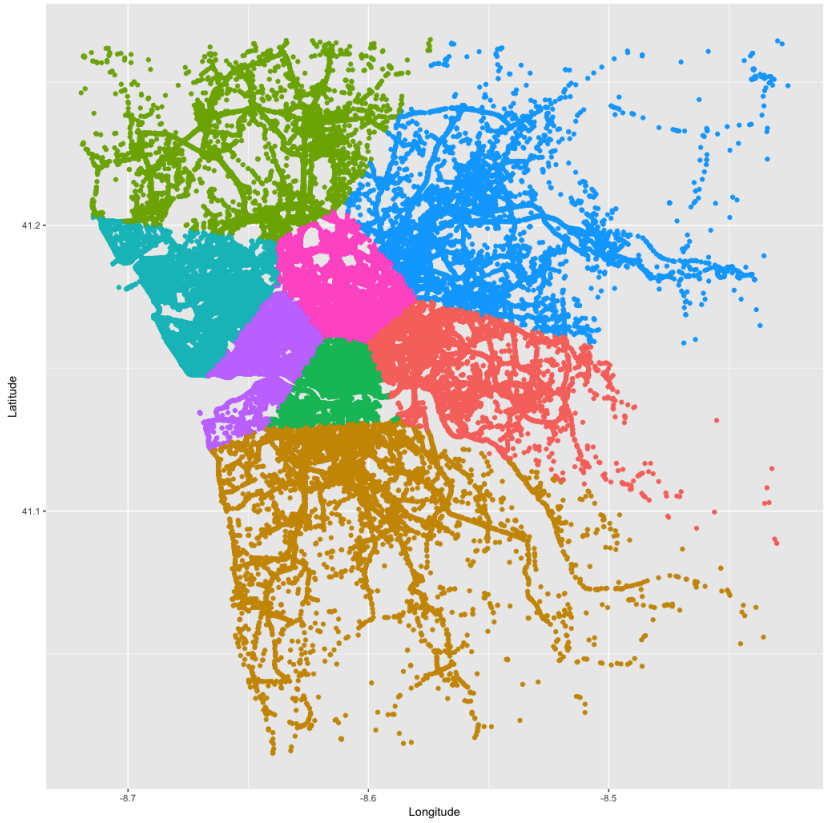


Figure 6.3: Coarse clustering with 8 clusters in smaller bounding box

To determine the best value for k at each level, a range $[8, 12]$ at the coarse level and $[248, 260]$ at fine grained level was specified. The clustering algorithm was then run for each value of k which falls into the range. The mean silhouette coefficient was then calculated for each resultant clustering. The value for k which yielded the highest mean silhouette coefficient at each level was chosen. In the present case, for the smaller bounding box, the best values for k at the coarse level turned out to be 8 clusters and the fine level 259 clusters.

A similar analysis was carried out for the data contained in the larger bounding box. The ideal clustering calculated was 18 clusters at the coarse level. Note that the drop off behaviours were very different between the city center regions and the rural regions. For this reason, rather than attempt a global clustering at the fine level, a hierarchical clustering scheme was devised. In this case, the data within each of the 18 higher regions were isolated and a separate clustering carried out within each of the 18 higher regions. A visualisation of these fine clusters would be too small to be discernible. For illustration purposes, figure 6.5 shows a sample clustering in the larger bounding box. The actual clusters are much finer than this at the lower level.

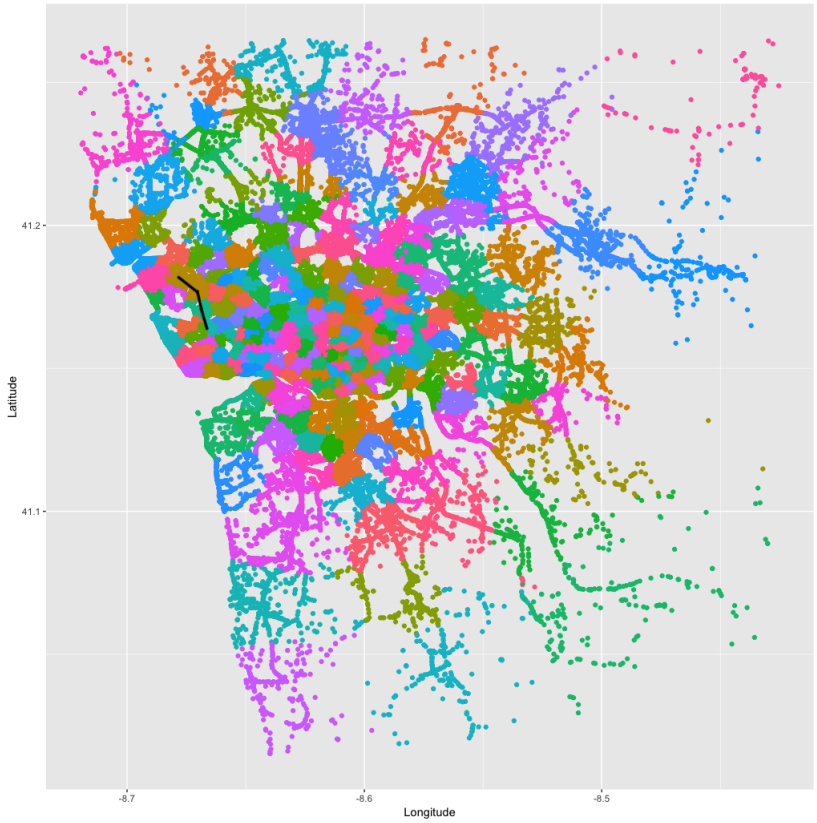


Figure 6.4: Fine clustering with 259 clusters in smaller bounding box. Sample trip shown.

Following the clustering, the trips are now represented as a sequence of cluster visits with repeated cluster IDs for sequences of GPS locations that remain within the same cluster. When a prediction is made, the simplest approach evaluated (beyond the baseline) predicts the destination cluster and reports the coordinates of the centroid of that cluster.

The error of the prediction is measured as the Euclidean distance between the true destination location and the predicted location. Therefore, even if the correct cluster is predicted, by taking the coordinates of the centroid of that cluster, an error is introduced. Assuming the cluster is a circle of radius r, then the expected distance from a random point in the cluster to its centroid will be $\frac{2}{3}r$.

6.4 Framework instantiation

In this implementation example, the observed *actions* are the lower level location clusters the taxi passes through once it leaves a source cluster at the beginning of a trip. Each cluster corresponds to the GPS data point reported by the taxi

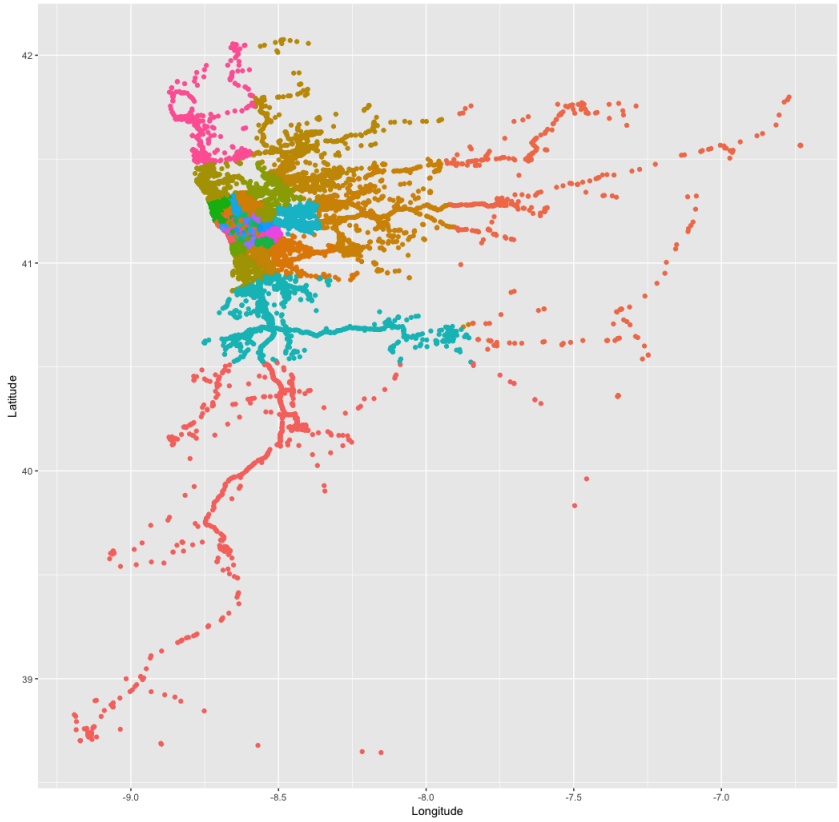


Figure 6.5: Sample clustering in larger bounding box

in the raw data.

The *activities* in this case are the transitions from one cluster to another as the taxi travels along its route. Each transition is represented as $\langle C_i, C_j \rangle$ where C_i and C_j are distinct clusters calculated in the pre-processing step from the raw GPS data (see section 6.3 for details).

Intent then is defined as the **transition into the final drop off location** of the trip by the taxi. More concretely, if the final drop off location is defined as a cluster ϕ then the intent of the driver in the taxi is to make the transition $\langle C_t, \phi \rangle$ where C_t is the final transition cluster visited by the taxi on the trip.

This final transition on the trip, which will occur at some point in the near future as required for an intent within the framework, is used to calculate the prediction error which is reported in the experimental results. Specifically, the prediction error is the distance between the predicted location of cluster ϕ and the actual drop off location (ground truth).

6.4.1 Hierarchical structure

There are two levels in the hierarchical model. As is the case in all implementation examples of the framework, clusters learnt in the data are utilized as abstractions which capture hidden structure in the data. For a description of how clusters are calculated in this particular example, please see section 6.3. A key difference between this instantiation of the framework and the other two examples is that manifold learning techniques such as t-SNE or LLE were not employed. This was due to the specific nature of the data for this implementation example where the observations represent geographical locations. It was a more interpretable approach to cluster on the observed domain (i.e: the pick-up, drop-off and transition GPS readings).

Once the levels are established via clustering, the upper level model is trained on the cluster labels such that it predicts a high level region for the drop-off. These predicted cluster labels (at higher level) are propagated to the lower model which predicts the probabilities of the drop-off cluster. There are maximum likelihood and weighted expectation evaluation models which were then employed to produce the final GPS coordinates predicted. These are described in section 6.4.2.2.

6.4.2 The model

To predict the destination of a taxi using an observed partial trajectory, we propose a hierarchical model. At the high level, the aim is to identify a general region or area the taxi is travelling to. The logic is that the early stages of a trip will be similar for any trip which terminates in destinations within the same neighbourhood. In the clustering scheme, this equates to picking a destination cluster from the coarse clustering. Once the higher destination region is identified, the algorithm will then "zoom" down and make a fine grained destination prediction within the region identified in the previous step. At this stage a cluster from the fine grained clustering is picked.

6.4.2.1 Upper model

The high level model is based on Discrete Time Markov Chains (DTMCs). The states of the DTMCs are defined to be the clusters the taxi passed through during its trip. Before training, the trips are factorised by source cluster and destination cluster pairs. A separate DTMC is trained for every source/destination pair.

Figure 6.6: DTMC model for destination region prediction

During prediction, the source cluster is known. Only those DTMCs associated with this source are considered. The destination which gives maximum likelihood for the observed transition is then chosen as the predicted destination region. A schematic representation of the higher model is shown in figure 6.6.

6.4.2.2 Lower model

The low level model is a Bayesian Inference model. We represent each journey as the sequence of clusters the taxi passed through. We calculate the destination cluster probability conditioned on the high level destination region, source cluster and last two clusters visited. The conditional probabilities are calculated as a set of probability tables by counting the transitions between clusters that are observed in the data set. We represent the source cluster, destination cluster and last two visited clusters x_i and x_{i-1} as unique cluster IDs. The notation $x_{i-1} \rightarrow x_i$ represents the last observed transition between two clusters in the most recent time step on the trip. Applying Bayes Rule, we have the following:

$$P(destination_{lower}|x_{i-1} \rightarrow x_i, source, destination_{higher}) =$$

$$\frac{\begin{aligned} P(x_{i-1} \rightarrow x_i|source, destination_{higher}, destination_{lower}) \times \\ P(x_{i-1}|source, destination_{higher}, destination_{lower}) \times \\ P(source|destination_{higher}, destination_{lower}) \times \\ P(destination_{higher}|destination_{lower}) \times P(destination_{lower}) \end{aligned}}{P(x_{i-1} \rightarrow x_i, source, destination_{higher})}$$

However, $P(x_{i-1} \rightarrow x_i, source, destination_{higher})$ is a normalization term and not required for the purpose of comparing different probabilities. In addition, this joint probability distribution is computationally expensive to calculate. Therefore, we simplify the above expression and compare based on proportionality.

Thus, we look for the destination cluster which maximises:

$$\begin{aligned}
&P(destination_{lower}|x_{i-1} \rightarrow x_i, source, destination_{higher}) \propto \\
&\quad P(x_{i-1} \rightarrow x_i|source, destination_{higher}, destination_{lower})\times \\
&\quad\quad P(x_{i-1}|source, destination_{higher}, destination_{lower})\times \\
&\quad\quad\quad P(source|destination_{higher}, destination_{lower})\times \\
&\quad\quad\quad\quad P(destination_{higher}|destination_{lower}) \times P(destination_{lower})
\end{aligned}$$

For example assume a trip starts in cluster 70 and in the current time step is at cluster 206 having previously been in cluster 115. Assume further that the high level model has predicted a destination region 5. Then we have:

$$P(destination_{lower}|x_{i-1} = 115 \rightarrow x_i = 206, source = 70, destination_{higher} = 5)$$

The experiments were run using different variations of the hierarchical model described. These are as follows:

Baseline The baseline model always predicts the centroid coordinates of the current cluster the taxi is in as the destination location ϕ.

Model 1 Reports the centroid coordinates of the most likely lower level destination cluster within the high level region as ϕ (a Maximum A Posteriori estimate).

Model 2 Reports the weighted average coordinates of all lower level destination cluster centroids within the high level region. The centroid locations of all lower clusters are weighted by the probability of each cluster and summed to give the expected destination coordinates ϕ.

Model 3 Reports the weighted average coordinates of the top 5 most likely lower level destination cluster centroids within the high level region. The centroid locations of the top 5 lower clusters are weighted by the probability of each cluster and summed to give the expected location ϕ. The probabilities of the top 5 clusters were rescaled before weighting was carried out.

Model 1 simply picks the most likely destination cluster at the lower level and reports the cluster centroid as the predicted location. However, given the disparity between the actual drop off location within the cluster and the location of the cluster centroid which the algorithm for this model reports, a discretisation error arising from the clustering scheme will always be incurred.

Models 2 and 3 are motivated as a means to try and address this discretisation error. In models 2 and 3, the algorithm reports an expected location by multiplying the lower cluster centroids with their probability and summing. This means that if a particular dropoff is within cluster A but very close to or bordering cluster B, then the probability of cluster B will be higher than normal therefore pulling the reported location somewhere between clusters A and B which is the desired behaviour.

6.5 Data sparsity

A problem encountered with the above method is the high dimensionality of the search space. The number of parameters mean that an exceedingly large number of combinations for those parameters are possible. Assuming a clustering scheme of 259 clusters (which comprises source and visit clusters) and the high level clustering of 8 clusters (the destination regions) then there are 259 sources by 259 previous locations by 259 current locations by 8 destination regions giving 138991832 possibilities at the higher level alone. This is not including the lower level destination prediction which must be made once a destination region is decided.

Some of these possibilities are ruled out by geographic features of the map itself. There are some clusters which cannot be reached from others in one transition and hence cannot form a $\langle$*previous location*, *current location*$\rangle$ pair. Even excluding these impossible transitions however, we found that the dataset did not contain information on every feasible transition. In absence of any observed data, the model predicts a zero probability for these transitions, which is clearly not the case. A common technique to counter this is to assign a non-zero probability to every possible destination (a uniform prior).

A reasonable approximation of what the missing probabilities might be is proposed as a smoothing technique. The idea is given a certain current location and candidate destination, assign higher probabilities to next locations which are in the general direction of that candidate destination.

In figure 6.7, the vehicle has travelled to its current location (marked C) from the source S. From there the vehicle has some choices around it for the next cluster to visit (marked green). Given the source and candidate destination D_2, the model assigns higher probabilities to next locations in the general direction of D_2. The model achieves this by calculating the angle $\Delta\theta$ which is the difference in

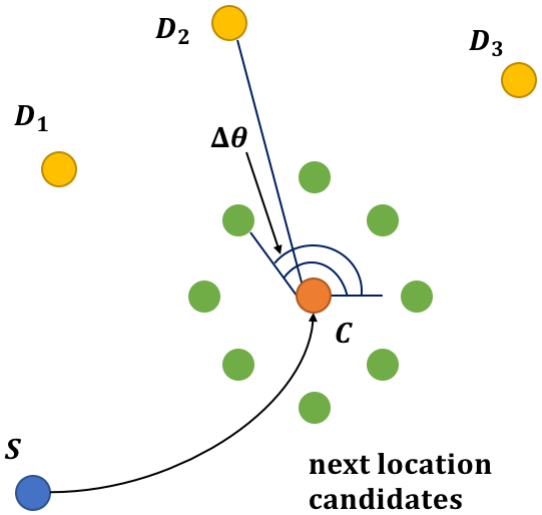


Figure 6.7: Estimation of missing probabilities with directional prior

heading between the possible next location and candidate destination. This angle is calculated for all possible next locations. The algorithm then ranks the next locations according to increasing $\Delta\theta$. Probability mass is applied to entries in this ordered list by a negative exponential function. When making a prediction, the algorithm first checks if there is information observed in the training data itself. If there is none available, it then uses the estimated probability from the model described above.

6.6 Experiments & results

6.6.1 Experiment description

To conduct the experiments, for each problem, the data was split into a training and test set. Approximately 80% of the trips were randomly sampled for the training set with the remaining 20% being allocated for testing. The evaluation metric used is the mean *Haversine distance* between our predicted drop off coordinates and actual drop off coordinates.

Haversine distance [155] is a metric used to quantify the distance between two points on the surface of a sphere, given their latitudes and longitudes. Also known as the "*great circle* distance", the metric measures the shortest distance between two points along the surface of a sphere (as opposed to a straight line through the sphere's interior). Therefore, using the Haversine distance formula improves on the more standard Euclidean distance metric by factoring the curvature of

the Earth into the calculation. Given two points x and y, the Haversine distance between them is calculated as follows:

$$d(x, y) =$$

$$2 \cdot R \cdot \arcsin\left(\sqrt{\sin^2\left(\frac{x_{lat} - y_{lat}}{2}\right) + \cos(x_{lat}) \cdot \cos(y_{lat}) \cdot \sin^2\left(\frac{x_{lon} - y_{lon}}{2}\right)}\right)$$

Where:

- x_{lat} and y_{lat} are the latitudes of the two points in radians.
- x_{lon} and y_{lon} are the longitudes of the two points in radians.
- R is the radius of the Earth (or sphere) in the chosen units (e.g., kilometers).
- d is the Haversine distance between the two points.

The value for the radius of the Earth R in this implementation example is 6371 km. This value was obtained by following the figures recommended by NASA which lists the equatorial radius of the Earth as 6378 km, the polar radius as 6357 km and finally, a volumetric mean radius of 6371 km [178].

6.6.2 Experiment results

The results from the different model variants (as described in the previous section) in the case of the smaller bounding box are shown in figure 6.8 and for the larger bounding box in figure 6.9. In the figures, the x axis indicates the percentage of trip travelled. The y axis is the average Haversine distance error over all the test trips at each point in the trip.

In all cases, it is expected that the baseline will out-perform predictive models at the very end of each trip. The reason for this is because, in the baseline model, the final destination of the trip is always defined as the centroid of the current location cluster the vehicle is in at that point. Therefore, the deeper a vehicle gets into the trip, the prediction error reduces until finally at the very end of the trip, the discrepancy between actual destination and predicted destination is the average discretisation error of the clustering (the discretisation error was discussed in more detail at the end of section 6.3).

In the smaller bounding box, the best performance is an error of about 0.6 km in the smaller map and 1.6 km dipping below 1 km as the trip progresses in the

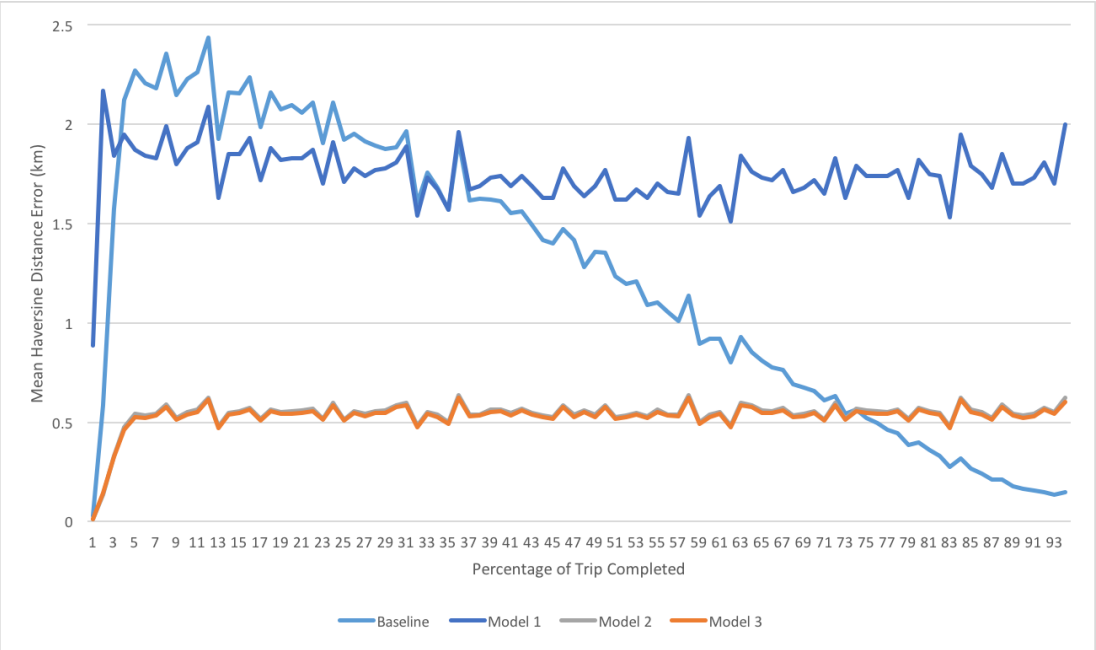


Figure 6.8: Experiment Results for smaller bounding box. Models 2 and 3 coincide.

Figure 6.9: Experiment Results for larger bounding box. Models 2 and 3 coincide.

larger map. This performance is seen with Models 2 and 3 which show the best performance in both smaller and larger area maps. On the graphs these lines coincide with minimal difference between them. This indicates that picking only the top 5 lower clusters does not alter the prediction performance too much as compared to weighting over all lower level destination clusters. The peak at the end of the trip is because at this stage the taxi is actually at its destination but the model still attempts to make a prediction which increases the error. It is noted that within the city centre region, there are patterns to learn and leverage to make a fairly accurate prediction.

Increasing the area covered by the map captures longer trips, however these longer trips were found to exhibit a distinct and more variable behaviour compared to the shorter trips. This difference resulted in a somewhat lower prediction accuracy particularly earlier on in the trip. Furthermore, as there is a larger number of destination clusters to choose from, the best result from the larger bounding box is still less accurate than those from the smaller bounding box. Finally the clusters themselves are larger which in turn amplifies any penalty from picking the wrong cluster.

It is noteworthy that in the smaller bounding box, the model reaches peak prediction accuracy very early on (from around 5% into the trip). We always report the centroid of the most likely cluster in Model 1. Our analysis showed that even if we predicted the correct cluster 100% of the time, the discretisation would still introduce an error. To address this problem, in Models 2 & 3, we report the weighted mean location of the low level cluster centroids (weighted by the probability of the cluster). The weighting also helps us in cases where our high-level cluster prediction is wrong. We believe the majority of these cases are where the true destination coordinates are close to the boundary between high level clusters so we end up picking an adjacent high level cluster instead. The low-level prediction is still able to predict the neighbouring low level clusters giving a final predicted location closer to the actual destination.

In the smaller map, while the prediction accuracy reaches its peak 5% into the trip, the prediction does not improve from there as in the larger map. Analysing the predictive behaviour of the model, we note the model soon starts to pick either the actual destination cluster or an adjacent cluster and this behaviour continues until the end of the trip. The Markov property may have an effect here as the model makes a prediction based only on the current information at each time step. In the current case, the best models outperform the baseline

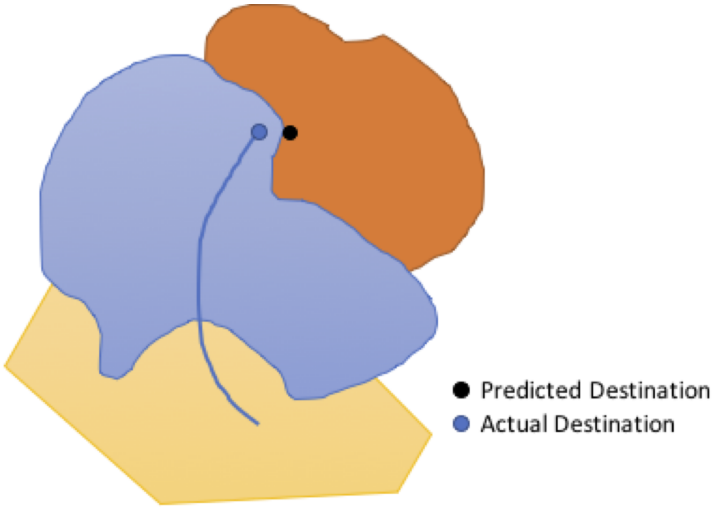


Figure 6.10: Weighted average prediction at lower level

until about 75% into the trip in the smaller map (considering the average error across all test trips at the 75% point). This indicates that the prediction model is delivering an insight of value for about three quarters into each trip on average.

The Predestination [89] algorithm reports an error of 2.8 km at the start of the trip and dropping to ∼1 km at the end (on a different data set from another city). In other papers where this data from Kaggle has been utilized, the authors note that the test set provided is very small for reliable model comparison and therefore test on their own local validation sets as we have done.

The results reported by the present framework will now be compared to the public benchmarks available on Kaggle for the same data set. This comparison comes with a caveat; all the results published on the Kaggle leaderboard have different data pre-processing and cleaning and the exact steps taken by the authors of those entries are not publicly known unless they have published their work. A comparison of our approach to few studies that have been published on this data set [92, 40] indicate that the authors employed data cleaning and modelling approaches very specific to this problem and data set. In other words, these approaches are finetuned for this specific problem to get the best result possible on this particular problem and this particular data set. This is likely true for all the results posted for this particular competition on the Kaggle leaderboard. Therefore, it must be noted that the results from the general framework proposed in this thesis serve to establish that the framework performs comparably to the benchmark results though not necessarily exceeding approaches finetuned for this problem specifically.

The framework reported a prediction error of 2.76 km on the Kaggle test set which would have placed 71^{st} on the final public leader board for the Kaggle competition. This is out of 382 total submissions which makes 71^{st} place a top 20% result. For further context, the best result obtained by bespoke models developed for this particular problem is 2.24 km which means the result from the proposed framework is only 0.32 km away.

Revisiting the two papers published in the literature cited above [92, 40] which utilise this dataset and report on the authors' approach and evaluation results for the Kaggle competition, a comparison may be made to the approach presented in this thesis. These studies report errors on the public Kaggle leader board of 2.28 km and 2.53 km respectively. Our model on the same small Kaggle test set of 320 trips scored 2.76 km on the public leader board, a directly comparable number to the public leader board figures of 2.28 km and 2.53 km cited above. In terms of the methods that were used, [92] used an ensemble tree-based approach with Random Forests and [40] applied deep learning. In both cases, as noted previously, the models were finetuned for this problem and bespoke approaches developed that took advantage of the data points specific to this use case.

The comparable performance of the proposed framework to state of the art results for the same problem support the claim that the framework does well across a range of applications though it may not always match the best results from a bespoke approach tuned specifically for a particular problem.

Another implication of the caveat that these external results have different data pre-processing is that the Kaggle leaderboard results may have come from models trained on data which has been less aggressively cleaned than our approach. The increased variability in the training data would help those models generalise better to unseen data.

Given these observations on different data pre-processing approaches on the Kaggle results, the comparisons made between the numbers from Kaggle and those from the thesis framework are actually evaluating two things; the pre-processing approach and the actual modelling approach of the framework. It is not immediately clear how much of the disparity between the results may be attributed to the framework and the pre-processing. It is also noted that size of the public test set is very small. Therefore, the local validation results presented in section 6.6.2 are viewed more significant as those results are calculated over a much larger number of trips and are evaluated purely on only differences in modelling approach. The pre-processing of the data is the same across all the local validation experiments.

6.7 Discussions & conclusion

We presented a hierarchical model for predicting the destination of a moving vehicle. The model made an initial prediction at a high level using Discrete Time Markov Chains for each possible source and destination pair. It then zoomed down to refine the prediction within that higher level cluster using Bayesian Inference. We showed that in earlier phases of the trip, the model correctly predicts the destination to within 0.5 km on average in our smaller map and around 1.6 km on the larger map (this figure improves as trip progresses on the larger map).

An improvement on these results may be achieved by considering extra pieces of information. For instance, factoring trips by time buckets and attempt to learn time based patterns. This captures the idea that the patterns of pick-ups and drop-offs will vary a lot depending on time of day. E.g.: Drop-offs at restaurants and entertainment venues are more likely in the evening. A second avenue to explore is to relax the Markov assumption and consider greater number of locations (such as last 5) in making our prediction. Finally an attempt can be made to learn driver preferences (such as favouring a particular section of the city for drop offs and working at a specific time of the day) for more flexible models.

CHAPTER 7

Concluding remarks

In conclusion, this Ph.D. thesis has presented a hierarchical framework for intent recognition in autonomous agents, demonstrating its effectiveness through three distinct implementation examples.

A key contribution of this research lies in the combination of hierarchical modeling, manifold learning, and machine learning techniques for intent recognition in autonomous agents. An extensive literature review was carried out and by integrating these methodologies in the context of intent recognition, this thesis has introduced a novel framework that fills a gap in the existing literature.

The utilization of hierarchical modeling allows for the effective representation and prediction of intent, enabling autonomous agents to make informed decisions in dynamic environments when interacting with other autonomous agents.

The formal thesis claim was that it is possible to infer the intent of an autonomous agent with a hierarchical approach that combines manifold learning techniques with machine learning models. To prove this claim, the intent recognition framework was conceptualised and described. The specific terms of action, activity and intent were also formally described.

The claim was then validated by experiment through three implementation examples. One example was predicting the intent of a resident in a smart home carrying out activities of daily living. The second implementation example was

employee moving around an office building, with applications in access control for enhanced security and occupancy prediction for efficient energy management. The intent was the next reader they will swipe. The third implementation example was moving vehicles. The particular data set used was taxis driving around Porto, Portugal. In this example, the intent of driver was defined as the final destination of the trip.

In the case of Implementation Example I (activities of daily living, chapter 4), a set of baseline experiments were run first which established a benchmark with a simple to compare other experiments against. Further experiments were then carried out for the intent recognition problem and validated against the benchmark.

For Implementation Example II (access control, chapter 5), the framework was applied to a proprietary user access control data set therefore no external benchmarks were available. However, an ablation experiment was setup that compared the results from the hierarchical model to a simple flat model which demonstrated the value of the techniques presented in the framework by enhancing model performance.

For Implementation Example III (moving vehicles, chapter 6) the performance of the framework was bench marked against both an internal baseline model and pre-existing external results from Kaggle for the same dataset. It was demonstrated that the models presented in chapter 6 exceeded the internal baseline model and reported comparable scores on the external results.

7.1 Future work

Looking ahead, there are several promising avenues for future work that can further enhance and extend the contributions made in this thesis. One area of exploration is the improvement of Bayesian sparsity in the hierarchical model. Although the current framework has shown promising results, incorporating more sophisticated Bayesian techniques could potentially enhance the model's performance. Additionally, investigating and addressing the shortcomings of the approach, such as the need for prior knowledge in the case of poor data, would be crucial for real-world applicability.

Future explorations are also relevant to expand the scope and impact of intent recognition for autonomous agents. For instance, one exciting direction would involve further experimentation to gain deeper insight into the functioning and

performance of the hierarchical aspect of the model.

The hierarchical aspect of the model presented in this thesis has proven beneficial in cases where imbalanced classes exist within the data. Its ability to capture the top down structure of intent recognition tasks has helped address this challenge, enabling more accurate predictions. However, further analysis and investigation will comprehensively evaluate that aspect of model performance and its applicability in an even great variety of practical applications.

In short term, within a 6 month period, the work would be extended by investigating further implementation examples and exploring different types of machine learning models for the existing implementations presented. One possible avenue for this exploration is investigating how well the framework can handle continuous data as the implementation examples described in this thesis are all discrete data formulations.

The intent recognition framework will also be evaluated in further real-world scenarios and domains not covered in this work, such as voice assistants, chatbots, or customer service applications. A key requirement would be to source or gather a diverse data set suitable for intent recognition tasks. One recent example of a potential data set to experiment with this framework is Google's *Schema-Guided Dialogue* (SGD) dataset [132, 98]. Relevant techniques for pre-processing in the natural language processing (NLP) domain such as word embeddings or pre-trained language models to capture semantic information would be considered in applying the framework to this data set.

In addition, the architecture of the intent recognition framework would be further developed to encompass these new domains. This research would consider incorporating mechanisms like attention, memory, or multi-task learning to improve performance. This would involve refining the hierarchical structure to handle deeper abstractions, such as word-level, sentence-level, and document-level hierarchies, to capture different levels of semantic information. In the process, the work would explore different hierarchical models suitable for intent recognition, such as *hierarchical recurrent neural networks* (HRNN), *hierarchical attention networks* (HAN), or *hierarchical transformers*. Their performance, scalability, and interpretability would be evaluated to select the most appropriate model for developing this framework further.

Advanced techniques for training and optimising the models within the framework will be evaluated. These techniques would include, for example, transfer

learning, fine-tuning and regularisation to improve generalisation and mitigate overfitting. Depending on the type of machine learning models being employed, experiments will be conducted with different optimisation algorithms, such as Adam or stochastic gradient descent with adaptive learning rates, and hyperparameter tuning methods to achieve optimal performance.

Methods to enhance the interpretability and explainability of this framework will also be investigated. Relevant techniques have emerged in the literature like attention visualisation, saliency maps, or activation mapping to gain insights into the decision-making process of the hierarchical model. In all enhancement work, a comprehensive benchmarking and comparative analysis of the proposed adjustments against current performance of the framework and existing intent recognition approaches would be carried out.

For a future Ph.D. student, some questions worth exploring are; do probabilistic models consistently outperform classic machine learning models (or vice versa)? What are the limits of data quality that the framework can be applied to? How much does the use of Bayesian priors help overcome data sparsity issues? To identify potential future directions for further improvement and extension of the intent recognition framework, emerging techniques like graph neural networks, meta-learning, or reinforcement learning will be considered to enhance its capabilities. In addition, as one of the challenges encountered in the present work was sourcing enough relevant data of sufficient quality, there is value in exploring ways to enhance this framework to handle low-resource or few-shot learning scenarios.

In the longer term, a more fundamental extension will be formulated; the framework was inspired by the *Belief-Desire-Intention* model [108]. Further inspiration could be drawn to extend the present framework to incorporate an aspect of *belief* inference. The key benefit of this approach would be the framework having a greater awareness of the context in which an autonomous agent moves from the present activity to the future activity (intent). The mapping which drives current activity to the agents' choice of next activity/intent would be their belief. For instance, a worker who holds a work from home employment and therefore believes staying at home would be their most beneficial course of action following breakfast would have a very different intent at that point in time to a worker who goes into the office.

In conclusion, this Ph.D. thesis has made novel contributions to the field of machine learning and intent recognition for autonomous agents by developing a hierarchical framework combining 3 specific techniques and validating it through a

diverse set of implementation examples. The identified future work, acknowledgment of current shortcomings and enthusiasm for further explorations have shed light on the contributions made and also the potential for future advancements.

By addressing these aspects and conducting additional experiments, the importance and necessity of frameworks for intent recognition can be further solidified, ultimately advancing the capabilities of autonomous agents to understand human intent; a crucial requirement as artificial intelligence systems and human beings work ever closer together.

References

[1] Mohamed Abdel-Basset, Hossam Hawash, Ripon K. Chakrabortty, Michael Ryan, Mohamed Elhoseny, and Houbing Song. ST-DeepHAR: Deep Learning Model for Human Activity Recognition in IoHT Applications. *IEEE Internet of Things Journal*, 8(6):4969–4979, 3 2021.

[2] Giovanni Acampora, Diane J. Cook, Parisa Rashidi, and Athanasios V. Vasilakos. A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 2013.

[3] Khaled A. Alaghbari, Mohamad Hanif Md. Saad, Aini Hussain, and Muhammad Raisul Alam. Activities Recognition, Anomaly Detection and Next Activity Prediction Based on Neural Networks in Smart Homes. *IEEE Access*, 10:28219–28232, 2022.

[4] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social LSTM: Human Trajectory Prediction in Crowded Spaces. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 961–971. IEEE, 6 2016.

[5] Mark V. Albert, Santiago Toledo, Mark Shapiro, and Konrad Kording. Using mobile phones for activity recognition in Parkinson's patients. *Frontiers in Neurology*, 2012.

[6] David W Albrecht, Ingrid Zukerman, and An E Nicholson. Bayesian Models for Keyhole Plan Recognition in an Adventure Game. *User Modeling and User-Adapted Interaction*, 8(1):5–47, 1998.

[7] David W Albrecht, Ingrid Zukerman, Ann E Nicholson, and Ariel Bud. Towards a Bayesian model for keyhole plan recognition in large domains. In *User Modeling: Proceedings of the Sixth International Conference UM97 Chia Laguna, Sardinia, Italy June 2–5 1997*, pages 365–376, 1997.

[8] James F Allen and George Ferguson. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, 4(5):531–579, 6 1994.

[9] Athanasios Anagnostis, Lefteris Benos, Dimitrios Tsaopoulos, Aristotelis Tagarakis, Naoum Tsolakis, and Dionysis Bochtis. Human Activity Recognition through Recurrent Neural Networks for Human–Robot Interaction in Agriculture. *Applied Sciences*, 11(5):2188, 3 2021.

[10] Oana Andrei, Muffy Calder, Matthew Higgs, and Mark Girolami. Probabilistic model checking of DTMC Models of user activity patterns. In Gethin Norman and William Sanders, editors, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 8657 LNCS, pages 138–153, Cham, 2014. Springer International Publishing.

[11] Alvina Anjum and Muhammad U. Ilyas. Activity recognition using smartphone sensors. In *2013 IEEE 10th Consumer Communications and Networking Conference, CCNC 2013*, 2013.

[12] Ankur Ankan and Abinash Panda. pgmpy: Probabilistic Graphical Models using Python. In *Proceedings of the 14th Python in Science Conference*, pages 6–11. Citeseer, 2015.

[13] G E M Anscombe. Intention. *Proceedings of the Aristotelian Society*, 57:321–332, 1957.

[14] Damla Arifoglu and Abdelhamid Bouchachia. Activity Recognition and Abnormal Behaviour Detection with Recurrent Neural Networks. *Procedia Computer Science*, 110:86–93, 2017.

[15] P. Arpaia, U. Cesaro, M. Chadli, H. Coppier, L. De Vito, A. Esposito, F. Gargiulo, and M. Pezzetti. Fault detection on fluid machinery using Hidden Markov Models. *Measurement*, 151:107126, 2 2020.

[16] Parviz Asghari and Ehsan Nazerfard. Activity Recognition Using Hierarchical Hidden Markov Models on Streaming Sensor Data. *9th International Symposium on Telecommunication: With Emphasis on Information and Communication Technology, IST 2018*, pages 416–420, 2019.

[17] Chris Baker and Rebecca Saxe. Bayesian Theory of Mind: Modeling Joint Belief-Desire Attribution. *Proceedings of the Thirty-Third Annual Conference of the Cognitive Science Society*, 6 2011.

[18] Akram Bayat, Marc Pomplun, and Due A. Tran. A study on human activity recognition using accelerometer data from smartphones. In *Procedia Computer Science*, 2014.

[19] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[20] Graeme Best and Robert Fitch. Bayesian intention inference for trajectory prediction with an unknown goal destination. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5817–5823. IEEE, 9 2015.

[21] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[22] Franco Blanchini and Stefano Miani. Dynamic programming. *Systems and Control: Foundations and Applications*, 153(9783319179322):193–234, 2015.

[23] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, pages 1395–1402. IEEE, 2005.

[24] David M. Blei. Build, Compute, Critique, Repeat: Data Analysis with Latent Variable Models. *Annual Review of Statistics and Its Application*, 1(1):203–232, 2014.

[25] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational Inference: A Review for Statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 4 2017.

[26] Anna Borucka, Edward Kozłowski, Rafał Parczewski, Katarzyna Antosz, Leszek Gil, and Daniel Pieniak. Supply Sequence Modelling Using Hidden Markov Models. *Applied Sciences*, 13(1):231, 12 2022.

[27] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 8 1996.

[28] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[29] David Browne, Michael Giering, and Steven D Prestwich. Deep Learning Human Activity Recognition. In *Proceedings of the 27th AIAI Irish Conference on Artificial Intelligence and Cognitive Science (AICS 2019), CEUR Workshop Proceedings*, volume 2563, pages 76–87, Galway, Ireland, 12 2019.

[30] Andreas Bulling, Ulf Blanke, and Bernt Schiele. A tutorial on human activity recognition using body-worn inertial sensors. *ACM Computing Surveys*, 46(3):1–33, 1 2014.

[31] Eduardo Casilari and Miguel A Oviedo-Jiménez. Automatic Fall Detection System Based on the Combined Use of a Smartphone and a Smartwatch. *PLOS ONE*, 10(11):1–11, 2015.

[32] Qingqing Chang and Jincheng Hu. Application of Hidden Markov Model in Financial Time Series Data. *Security and Communication Networks*, 2022:1–10, 4 2022.

[33] Eugene Charniak and Robert P Goldman. A Bayesian model of plan recognition. *Artificial Intelligence*, 64(1):53–79, 1993.

[34] Kaixuan Chen, Dalin Zhang, Lina Yao, Bin Guo, Zhiwen Yu, and Yunhao Liu. Deep Learning for Sensor-based Human Activity Recognition. *ACM Computing Surveys*, 54(4):1–40, 5 2022.

[35] Tianqi Chen and Carlos Guestrin. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA, 2016. Association for Computing Machinery.

[36] Robin Cohen, Fei Song, Bruce Spencer, and Peter Van Beek. Exploiting temporal and novel information from the user in plan recognition. *User Modeling and User-Adapted Interaction*, 1:125–148, 1991.

[37] Diane J. Cook, Aaron S. Crandall, Brian L. Thomas, and Narayanan C. Krishnan. CASAS: A smart home in a box. *Computer*, pages 62–69, 2013.

[38] Marco F Cusumano-Towner, Alexey Radul, David Wingate, and Vikash K Mansinghka. Probabilistic programs for inferring the goals of autonomous agents. *CoRR*, abs/1704.04977, 2017.

[39] Ji Dai, Jonathan Wu, Behrouz Saghafi, Janusz Konrad, and Prakash Ishwar. Towards privacy-preserving activity recognition using extremely low

temporal and spatial resolution cameras. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2015.

[40] Alexandre De Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent, and Yoshua Bengio. Artificial Neural Networks Applied to Taxi Destination Prediction. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDDC'15, pages 40–51, Aachen, Germany, Germany, 2015. CEUR-WS.org.

[41] Ingo Deutschmann and Johan Lindholm. Behavioral biometrics for DARPA's Active Authentication program. In *International Conference of the BIOSIG Special Interest Group (BIOSIG)*, Darmstadt, Germany, 10 2013. IEEE.

[42] Nidhi Dua, Shiva Nand Singh, and Vijay Bhaskar Semwal. Multi-input CNN-GRU based human activity recognition using wearable sensors. *Computing*, 103(7):1461–1478, 7 2021.

[43] Bradley Efron and Robert J Tibshirani. *An introduction to the bootstrap*. CRC press, 1994.

[44] J Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 6 1990.

[45] Andrés Gómez Escobar and Fernando de la Rosa. Motion Learning Problem in Robotics using Bayes Networks. In *IADIS International Conference Intelligent Systems and Agents (ISA 2011)*, Rome, Italy, 7 2011.

[46] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[47] Michael Fagan and Pádraig Cunningham. Case-Based Plan Recognition in Computer Games. In *Case-Based Reasoning Research and Development*, pages 161–170. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[48] G.D. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[49] William B Frakes and Ricardo Baeza-Yates, editors. *Information Retrieval: Data Structures and Algorithms*. Prentice-Hall, Inc., USA, 1992.

[50] Annalisa Franco, Antonio Magnani, and Dario Maio. A multimodal ap-

proach for human activity recognition based on skeleton and RGB data. *Pattern Recognition Letters*, 131:293–299, 3 2020.

[51] Monica Franzese and Antonella Iuliano. Hidden Markov Models. In *Encyclopedia of Bioinformatics and Computational Biology*, pages 753–762. Elsevier, 2019.

[52] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 8 1997.

[53] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 10 2001.

[54] K. Fukunaga and L. Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1 1975.

[55] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. Chapman and Hall/CRC, 11 2013.

[56] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-6(6):721–741, 11 1984.

[57] Michael Georgeff, Barney Pell, Martha Pollack, Milind Tambe, and Michael Wooldridge. The belief-desire-intention model of agency. In *Intelligent Agents V: Agents Theories, Architectures, and Languages: 5th International Workshop, ATAL'98 Paris, France, July 4–7, 1998 Proceedings 5*, pages 1–10, 1999.

[58] W. R. Gilks, N. G. Best, and K. K. C. Tan. Adaptive Rejection Metropolis Sampling within Gibbs Sampling. *Applied Statistics*, 44(4):455, 1995.

[59] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[60] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[61] Luis Gutiérrez, Juan Bekios-Calfa, and Brian Keith. A Review on Bayesian

Networks for Sentiment Analysis. In *Trends and Applications in Software Engineering*, pages 111–120. Springer, 2019.

[62] Tim Van hamme, Vera Rimmer, Davy Preuveneers, Wouter Joosen, Mustafa A Mustafa, Aysajan Abidin, and Enrique Argones-Rúa. Frictionless Authentication Systems: Emerging Trends, Research Challenges }and Opportunities. *CoRR*, abs/1802.07233, 2018.

[63] Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *IJCAI International Joint Conference on Artificial Intelligence*, 2016.

[64] Mohammed Mehedi Hassan, Md Zia Uddin, Amr Mohamed, and Ahmad Almogren. A robust human activity recognition system using smartphone sensors and deep learning. *Future Generation Computer Systems*, 2018.

[65] David Heckerman. Bayesian Networks for Data Mining. *Data Mining and Knowledge Discovery*, 1(1):79–119, 1997.

[66] David Heckerman. A Tutorial on Learning with Bayesian Networks. In *Learning in Graphical Models*, pages 301–354. Springer Netherlands, Dordrecht, 1998.

[67] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[68] Matthew D Hoffman and Andrew Gelman. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1):1593–1623, 2014.

[69] R A Howard. *Dynamic Probabilistic Systems, Volume I: Markov Models*. Dover Books on Mathematics. Dover Publications, 2007.

[70] Richard Hughey and Anders Krogh. Hidden Markov models for sequence analysis: extension and analysis of the basic method. *Bioinformatics*, 12(2):95–107, 1996.

[71] Anil K Jain and Richard C Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

[72] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer US, New York, NY, 2021.

[73] Nathalie Japkowicz and Mohak Shah. *Evaluating Learning Algorithms: A Classification Perspective*. Cambridge University Press, 2011.

[74] Finn V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2007.

[75] Wenchao Jiang and Zhaozheng Yin. Human activity recognition using wearable sensors by deep convolutional neural networks. In *MM 2015 - Proceedings of the 2015 ACM Multimedia Conference*, 2015.

[76] B. H. Juang and L. R. Rabiner. Hidden Markov Models for Speech Recognition. *Technometrics*, 33(3):251, 8 1991.

[77] Laura Kaikkonen, Tuuli Parviainen, Mika Rahikainen, Laura Uusitalo, and Annukka Lehikoinen. Bayesian Networks in Environmental Risk Assessment: A Review. *Integrated Environmental Assessment and Management*, 17(1):62–78, 1 2021.

[78] S. Katz. Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31(12):721–727, 12 1983.

[79] Henry A Kautz. A formal theory of plan recognition and its implementation. In *Reasoning about Plans*, pages 69–124. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1991.

[80] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pages 3149–3157, Red Hook, NY, USA, 2017. Curran Associates Inc.

[81] John G. Kemeny, J Laurie Snell, and Anthony W Knapp. Stochastic Processes. In *Denumerable Markov Chains: with a chapter of Markov Random Fields by David Griffeath*, pages 40–57. Springer New York, New York, NY, 1976.

[82] Charles Kemp, Amy Perfors, and Joshua B. Tenenbaum. Learning overhypotheses with hierarchical Bayesian models. *Developmental Science*, 10(3):307–321, 2007.

[83] Robert K L Kennedy, Justin M Johnson, and Taghi M Khoshgoftaar. The Effects of Class Label Noise on Highly-Imbalanced Big Data. In *2021 IEEE*

33rd International Conference on Tools with Artificial Intelligence (ICTAI), pages 1427–1433, Washington, DC, USA, 2021.

[84] Boris Kerkez and Michael T Cox. Incremental case-based plan recognition using state indices. In *Case-Based Reasoning Research and Development: 4th International Conference on Case-Based Reasoning, ICCBR 2001 Vancouver, BC, Canada, July 30–August 2, 2001 Proceedings 4*, pages 291–305, 2001.

[85] Zafar A. Khan and Won Sohn. Abnormal human activity recognition system based on R-transform and kernel discriminant technique for elderly home care. *IEEE Transactions on Consumer Electronics*, 2011.

[86] Zanobya N. Khan and Jamil Ahmad. Attention induced multi-head convolutional neural network for human activity recognition. *Applied Soft Computing*, 110:107671, 10 2021.

[87] A. Klaser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *Procedings of the British Machine Vision Conference 2008*, pages 1–99. British Machine Vision Association, 2008.

[88] D Koller and N Friedman. *Probabilistic Graphical Models: Principles and Techniques*. Adaptive computation and machine learning. MIT Press, 2009.

[89] John Krumm and Eric Horvitz. Predestination: Inferring Destinations from Partial Trajectories. In *UbiComp 2006: Ubiquitous Computing*, pages 243–260, Berlin, Heidelberg, 2006. Springer.

[90] Jennifer R. Kwapisz, Gary M. Weiss, and Samuel A. Moore. Activity recognition using cell phone accelerometers. *ACM SIGKDD Explorations Newsletter*, 2011.

[91] Yongjin Kwon, Kyuchang Kang, and Changseok Bae. Unsupervised learning for human activity recognition using smartphone sensors. *Expert Systems with Applications*, 41(14):6067–6074, 10 2014.

[92] Hoang Thanh Lam, Ernesto Diaz-Aviles, Alessandra Pascale, Yiannis Gkoufas, and Bei Chen. (Blue) Taxi Destination and Trip Time Prediction from Partial Trajectories. In *Proceedings of the 2015th International Conference on ECML PKDD Discovery Challenge - Volume 1526*, ECMLPKDDDC'15, pages 63–74, Aachen, Germany, Germany, 2015. CEUR-WS.org.

[93] Laptev and Lindeberg. Space-time interest points. In *Proceedings Ninth*

IEEE International Conference on Computer Vision, pages 432–439. IEEE, 2003.

[94] Oscar D. Lara and Miguel A. Labrador. A Survey on Human Activity Recognition using Wearable Sensors. *IEEE Communications Surveys & Tutorials*, 15(3):1192–1209, 11 2012.

[95] Kathryn Blackmond Laskey and Suzanne M Mahoney. Network fragments: Representing knowledge for constructing probabilistic models. *arXiv preprint arXiv:1302.1557*, 2013.

[96] Isah A Lawal and Sophia Bano. Deep Human Activity Recognition Using Wearable Sensors. In *Proceedings of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '19, pages 45–48, New York, NY, USA, 2019. Association for Computing Machinery.

[97] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 5 2015.

[98] Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. SGD-X: A Benchmark for Robust Generalization in Schema-Guided Dialogue Systems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10938–10946, 2022.

[99] Neal Lesh, Charles Rich, and Candace L. Sidner. Using Plan Recognition in Human-Computer Collaboration. In *UM99 User Modeling*, pages 23–32. Springer, 1999.

[100] Chao Li, Lijun Suna, and Xiangpei Hua. A context-aware lighting control system for smart meeting rooms. *Systems Engineering Procedia*, 4:314–323, 2012.

[101] Yaqing Liu, Dantong Ouyang, Yong Liu, and Rong Chen. A novel approach based on time cluster for activity recognition of daily living in smart homes. *Symmetry*, 9(10), 2017.

[102] S Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[103] Sam Maes, Karl Tuyls, Bram Vanschoenwinkel, and Bernard Manderick. Credit card fraud detection using Bayesian and neural networks. In *Pro-*

ceedings of the 1st international naiso congress on neuro fuzzy technologies, volume 261, page 270, 2002.

[104] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[105] Shie Mannor, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, Xin Jin, Jiawei Han, and Xinhua Zhang. K-Means Clustering. In *Encyclopedia of Machine Learning*, pages 563–564. Springer, Boston, MA, 2011.

[106] Wenji Mao and Fei-Yue Wang. Forecasting Group Behavior via Probabilistic Plan Inference. In Wenji Mao and Fei-Yue Wang, editors, *New Advances in Intelligence and Security Informatics*, chapter 4, pages 33–44. Academic Press, Boston, 2012.

[107] Taylor R Mauldin, Marc E Canby, Vangelis Metsis, Anne H H Ngu, and Coralys Cubero Rivera. SmartFall: A Smartwatch-Based Fall Detection System Using Deep Learning. *Sensors*, 18(10), 2018.

[108] Hugh J. McCann and M. E. Bratman. Intention, Plans, and Practical Reason. *Noûs*, 25(2):230, 4 1991.

[109] Richard McElreath. *Statistical Rethinking*. Chapman and Hall/CRC, New York, 2nd edition, 3 2020.

[110] Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction, 2020.

[111] Geoffrey J McLachlan, Sharon X Lee, and Suren I Rathnayake. Finite Mixture Models. *Annual Review of Statistics and Its Application*, 6(1):355–378, 2019.

[112] Felipe Meneguzzi and Ramon Fraga Pereira. A Survey on Goal Recognition as Planning. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence*, pages 4524–4532, California, 8 2021. International Joint Conferences on Artificial Intelligence Organization.

[113] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 6 1953.

[114] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Yoshua Bengio and

Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR, Workshop Track Proceedings*, 2013.

[115] Tomás Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[116] T K Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.

[117] L E Mukhanov. Using Bayesian Belief Networks for Credit Card Fraud Detection. In *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications*, AIA '08, pages 221–225, USA, 2008. ACTA Press.

[118] Kevin P. Murphy. *Machine Learning A Probabilistic Perspective*. The MIT Press, 8 2012.

[119] Myrto Mylopoulos and Elisabeth Pacherie. Intentions: The dynamic hierarchical model revisited. *WIREs Cognitive Science*, 10(2), 3 2019.

[120] Ali A. Naeem and Kenneth N. Brown. Inferring destination from mobility data. In *CEUR Workshop Proceedings*, volume 2086, pages 153–165. CEUR-WS, 2017.

[121] Radford M Neal. MCMC using Hamiltonian dynamics. In *Handbook of markov chain monte carlo*, chapter 5, pages 113–162. Chapman and Hall/CRC, 2011.

[122] Andrew Ng, Michael Jordan, and Yair Weiss. On Spectral Clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, volume 14, pages 849–856. MIT Press, 2001.

[123] Tuan Anh Nguyen and Marco Aiello. Energy intelligent buildings based on user activity: A survey. *Energy and Buildings*, 56:244–257, 1 2013.

[124] Francisco Javier Ordóñez and Daniel Roggen. Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition. *Sensors (Switzerland)*, 2016.

[125] Omar Oreifej and Zicheng Liu. HON4D: Histogram of Oriented 4D Normals for Activity Recognition from Depth Sequences. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 716–723. IEEE, 6 2013.

[126] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[127] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[128] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[129] Hai Van Pham, Dat Hoang Thanh, and Philip Moore. Hierarchical Pooling in Graph Neural Networks to Enhance Classification Performance in Large Datasets. *Sensors*, 21(18):6070, 9 2021.

[130] M. A Post. An Embedded Implementation of Bayesian Network Robot Programming Methods. In *Proceedings of the IMA Conference on Mathematics of Robotics*. Institute of Mathematics and its Applications, 2015.

[131] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[132] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696, 2020.

[133] Douglas Reynolds. Gaussian Mixture Models. In *Encyclopedia of Biometrics*, pages 659–663. Springer, Boston, MA, 2009.

[134] C J Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, USA, 2nd edition, 1979.

[135] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods*. Springer New York, New York, NY, 2004.

[136] Seyed Ali Rokni, Marjan Nourollahi, and Hassan Ghasemzadeh. Personalized human activity recognition using convolutional neural networks. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, 2018.

[137] Charissa Ann Ronao and Sung-Bae Cho. Human activity recognition with

smartphone sensors using deep learning neural networks. *Expert Systems with Applications*, 59:235–244, 10 2016.

[138] Peter J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20(C):53–65, 1987.

[139] Sam T Roweis and Lawrence K Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323 LP – 2326, 12 2000.

[140] Fariba Sadri. Ambient intelligence: A survey. *ACM Computing Surveys*, 2011.

[141] Sebastian Scheurer, Salvatore Tedesco, Kenneth N Brown, and Brendan O'Flynn. Human activity recognition for emergency first responders via body-worn inertial sensors. In *2017 IEEE 14th International Conference on Wearable and Implantable Body Sensor Networks (BSN)*, pages 5–8, 2017.

[142] C.F. Schmidt, N.S. Sridharan, and J.L. Goodson. The plan recognition problem: An intersection of psychology and artificial intelligence. *Artificial Intelligence*, 11(1-2):45–83, 8 1978.

[143] Paul Scovanner, Saad Ali, and Mubarak Shah. A 3-dimensional sift descriptor and its application to action recognition. In *Proceedings of the 15th ACM international conference on Multimedia*, pages 357–360, New York, NY, USA, 9 2007. ACM.

[144] Jongho Shin, Christine A. Espin, Stanley L. Deno, and Scott McConnell. Use of hierarchical linear modeling and curriculum-based measurement for assessing academic growth and instructional factors for students with learning difficulties. *Asia Pacific Education Review*, 5(2):136–148, 6 2004.

[145] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 1 2013.

[146] Pekka Siirtola and Juha Röning. Recognizing Human Activities User-independently on Smartphones Based on Accelerometer Data. *International Journal of Interactive Multimedia and Artificial Intelligence*, 2012.

[147] D. J Spiegelhalter, J. P Myles, D. R Jones, and K. R Abrams. Methods

in health service research: An introduction to bayesian methods in health technology assessment. *BMJ*, 319(7208):508–512, 8 1999.

[148] Chengwei Su, Angeline Andrew, Margaret R Karagas, and Mark E Borsuk. Using Bayesian networks to discover relations between genes, environment, and disease. *BioData Mining*, 6(1):6, 12 2013.

[149] Gita Sukthankar, Christopher Geib, Hung Hai Bui, David Pynadath, and Robert P Goldman. *Plan, activity, and intent recognition: Theory and practice*. Newnes, 2014.

[150] Joshua B. Tenenbaum, V. De Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 12 2000.

[151] Joshua B. Tenenbaum, Charles Kemp, Thomas L. Griffiths, and Noah D. Goodman. How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285, 2011.

[152] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 17(1):168–192, 1 2021.

[153] The Oxford Dictionary of Philosophy. direction of fit.

[154] Dario Valcamonico, Piero Baraldi, and Enrico Zio. Natural Language Processing and Bayesian Networks for the Analysis of Process Safety Events. In *2021 5th International Conference on System Reliability and Safety (ICSRS)*, pages 216–221. IEEE, 11 2021.

[155] Glen Van Brummelen. *Heavenly Mathematics: The Forgotten Art of Spherical Trigonometry*. Princeton University Press, 2013.

[156] Laurens van der Maaten. Learning a parametric embedding by preserving local structure. *Journal of Machine Learning Research*, 5:384–391, 2009.

[157] Laurens van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. *Journal of Machine Learning Research*, 15(93):3221–3245, 2014.

[158] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[159] Laurens van der Maaten and Geoffrey Hinton. Visualizing non-metric similarities in multiple maps. *Machine Learning*, 87(1):33–55, 2012.

[160] Tim L. M. van Kasteren, Gwenn Englebienne, and Ben J A Kröse. Trans-

ferring knowledge of activity recognition across sensor networks. In *Lecture Notes in Computer Science*, 2010.

[161] Tim L. M. van Kasteren, Gwenn Englebienne, and Ben J A Kröse. Hierarchical Activity Recognition Using Automatically Clustered Actions. In *Ambient Intelligence, Lecture Notes in Computer Science*, volume 7040, pages 82–91. Springer, Berlin, Heidelberg, 2011.

[162] Tim L. M. van Kasteren, Gwenn Englebienne, and Ben J A Kröse. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity recognition in pervasive intelligent environments*, pages 165–186. Springer, 2011.

[163] Tim L. M. van Kasteren and Ben J A Kröse. Bayesian activity recognition in residence for elders. In *2007 3rd IET International Conference on Intelligent Environments*, pages 209–212, 2007.

[164] Tim L. M. van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben J A Kröse. Accurate activity recognition in a home setting. *UbiComp 2008 - Proceedings of the 10th International Conference on Ubiquitous Computing*, pages 1–9, 2008.

[165] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In I Guyon, U Von Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[166] J. David Velleman and Michael E. Bratman. Intention, Plans, and Practical Reason. *The Philosophical Review*, 1987.

[167] Manuela Veloso. *Planning and Learning by Analogical Reasoning*, volume 886. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[168] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[169] Fabio Veronese, Daniele Proserpio, Sara Comai, Matteo Matteucci, and Fabio Salice. SHARON: A Simulator of Human Activities, ROutines and Needs. In *Studies in Health Technology and Informatics*, 2015.

[170] T Vilarinho, B Farshchian, D G Bajer, O H Dahl, I Egge, S S Hegdal, A Lønes, J N Slettevold, and S M Weggersen. A Combined Smartphone and Smartwatch Fall Detection System. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 1443–1448, 10 2015.

[171] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, 4 1967.

[172] Stefan Wahl and Hans Spada. Children's reasoning about intentions, beliefs and behaviour. *Cognitive Science Quarterly*, 1(1):3–32, 2000.

[173] J. Wan, M Li, M O'Grady, X Gu, M A A H Alawlaqi, and G O'Hare. Time-bounded Activity Recognition for Ambient Assisted Living. *IEEE Transactions on Emerging Topics in Computing*, page 1, 2018.

[174] Heng Wang, Alexander Klaser, Cordelia Schmid, and Cheng-Lin Liu. Action recognition by dense trajectories. In *CVPR 2011*, pages 3169–3176. IEEE, 6 2011.

[175] Heng Wang and Cordelia Schmid. Action Recognition with Improved Trajectories. In *2013 IEEE International Conference on Computer Vision*, pages 3551–3558. IEEE, 12 2013.

[176] Zihao W. Wang, Vibhav Vineet, Francesco Pittaluga, Sudipta N. Sinha, Oliver Cossairt, and Sing Bing Kang. Privacy-preserving action recognition using coded aperture videos. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2019.

[177] Geert Willems, Tinne Tuytelaars, and Luc Gool. An Efficient Dense and Scale-Invariant Spatio-Temporal Interest Point Detector. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 650–663, Berlin, Heidelberg, 2008. Springer-Verlag.

[178] David R. Williams. Earth Fact Sheet. Technical report, NASA, https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html, 6 2023.

[179] James R. Williamson, Andrew Dumas, Greg Ciccarelli, Austin R. Hess, Brian A. Telfer, and Mark J. Buller. Estimating load carriage from a body-worn accelerometer. In *2015 IEEE 12th International Conference on*

Wearable and Implantable Body Sensor Networks (BSN), pages 1–6. IEEE, 6 2015.

[180] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1 1992.

[181] A. Y. Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 254–265. IEEE, 4 2013.

[182] Suguru Yabe, Yohei Hamada, Rina Fukuchi, Shunichi Nomura, Norio Shigematsu, Tsutomu Kiguchi, and Kenta Ueki. Quantitative logging data clustering with hidden Markov model to assist log unit classification. *Earth, Planets and Space*, 74(1):93, 12 2022.

[183] Jian Bo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiao Li Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *IJCAI International Joint Conference on Artificial Intelligence*, 2015.

[184] Jing Yuan, Yu Zheng, Xing Xie, and Guangzhong Sun. T-drive: Enhancing driving directions with taxi drivers' intelligence. *IEEE Transactions on Knowledge and Data Engineering*, 25(1):220–232, 2013.

[185] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun, and Yan Huang. T-drive: Driving directions based on taxi trajectories. In *GIS: Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, GIS '10, pages 99–108, New York, NY, USA, 2010. ACM.

[186] Matei Zaharia, Reynold S. Xin, Patrick Wendell, Tathagata Das, Michael Armbrust, Ankur Dave, Xiangrui Meng, Josh Rosen, Shivaram Venkataraman, Michael J. Franklin, Ali Ghodsi, Joseph Gonzalez, Scott Shenker, and Ion Stoica. Apache Spark. *Communications of the ACM*, 59(11):56–65, 10 2016.

[187] Ming Zeng, Le T. Nguyen, Bo Yu, Ole J. Mengshoel, Jiang Zhu, Pang Wu, and Joy Zhang. Convolutional Neural Networks for human activity recognition using mobile sensors. In *Proceedings of the 2014 6th International Conference on Mobile Computing, Applications and Services, MobiCASE 2014*, 2015.

[188] Dalin Zhang, Lina Yao, Kaixuan Chen, Sen Wang, Pari Delir Haghighi, and Caley Sullivan. A Graph-Based Hierarchical Attention Model for Movement Intention Detection from EEG Signals. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 27(11):2247–2253, 11 2019.

[189] Nevin L Zhang and David Poole. A simple approach to Bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, Banff, Alberta, 1994.

[190] R Zhang, F Hoflinger, and L Reindl. Inertial Sensor Based Indoor Localization and Monitoring System for Emergency Responders. *IEEE Sensors Journal*, 13(2):838–848, 2 2013.

[191] Shugang Zhang, Zhiqiang Wei, Jie Nie, Lei Huang, Shuang Wang, and Zhen Li. A Review on Human Activity Recognition Using Vision-Based Method. *Journal of Healthcare Engineering*, 2017:1–31, 2017.

[192] Yuejin Zhang, Hengyue Jia, Aihua Li, Jianbing Liu, and Haifeng Li. Study on Prediction of Activities of Daily Living of the Aged People Based on Longitudinal Data. *Procedia Computer Science*, 91:470–477, 2016.

[193] Brian D Ziebart, Andrew L Maas, Anind K Dey, and J Andrew Bagnell. Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-aware Behavior. In *Proceedings of the 10th International Conference on Ubiquitous Computing*, UbiComp '08, pages 322–331, New York, NY, USA, 2008. ACM.

Appendices

APPENDIX A

Implementation Example I - Current Activity Evaluation

This content discusses current activity prediction using data from Implementation Example I. Current activity prediction is out of scope for the problem considered in thesis (which is intent recognition).

Therefore, these results in Appendix A are not considered a contribution coming out of this thesis towards the PhD. Furthermore, these results in this section are not used anywhere in the thesis.

Nevertheless, these results for current activity prediction are still being included for completeness.

A.1 Overview

The *current activity* prediction experiments were carried out to benchmark the performance of the proposed framework against the literature. Therefore comparisons are made between the results obtained with the framework and the best corresponding results for each house presented in the benchmark paper [162].

Though strictly not intent recognition, the methodology described for the framework in this thesis may be applied to current activity prediction. For these benchmark experiments, the target for classification and prediction was current activity happening right now rather than intent (which occurs at some point in the near future).

To compare the performance against the benchmark paper [162], one must consider the exact evaluation scheme the authors had employed and reproduce the same. In section 8.4.3 of [162], the authors state the following:

> The discretized ground truth labels are used to learn the model parameters. However, if we were to use the discretized ground truth for calculating the performance measures of the model as well, our measure would not take into account the discretization error. Therefore, we evaluate the performance of our models using the actual ground truth, which was obtained with a one second accuracy.

Stating only that model parameters were calculated with discretised ground truth labels (for example at 60 seconds) but the evaluation was at one second accuracy is an ambiguous proposition; there are few interpretations for the above statement which are reasonably valid. One interpretation is that every time slice of 1 second was evaluated independently using a model trained on 60 second time slices. However, this approach was not able to reproduce the figures provided on the benchmark paper. In the experiments conducted, the reproducible interpretation of the above statement was that the authors of [162] have counted an inference as "correct" if any of the model predictions for the 60 slices of 1 second each that correspond to the ground truth data match the ground truth label.

This method, referred to in the rest of this section as *paper evaluation*, can be interpreted as giving the model 60 chances to get it right for any given 60 second discretisation interval. However, the above evaluation scheme from [162] produces some surprising results (such as the almost perfect confusion matrices seen later in this section).

Therefore with the intent recognition experiments, the decision was made to use a more traditional evaluation method. This method, which will be referred to as *thesis evaluation* in this section going forward, is to carry out the evaluation at the same discretisation level as the model training. Therefore, the training and test sets are discretised at 60 seconds and then the experiments are run.

A.2 Evaluating the hierarchical framework for current activity prediction

The goal of the evaluation here is to contrast and compare the thesis and paper evaluation schemes for the same problem and understand what happens when each method is applied. Therefore, the best performing model in the intent recognition experiments (hierarchical formulation of XGBoost models) are trained to predict the current ongoing activity of the resident rather than an intent in the near future.

Once the models were trained, the results were evaluated using both paper evaluation and thesis evaluation schemes. For each evaluation scheme, a summary overview is reported for each house. The summary includes the average of the precision, recall and f1-score. The averages were calculated as described in section 4.7. In addition, the standard deviation is also reported for all relevant measures.

In the case of paper evaluation, a like-for-like comparison may be made to the figures reported in the benchmark paper and this analysis is presented where different models are compared using the same evaluation scheme. This allows the reader to compare the performance of the framework against the benchmark paper figures.

Following that a second head-to-head comparison is also made, this time comparing the two evaluation schemes using the same model. This captures the differences observed between the two evaluation schemes.

Finally a summary of the findings are provided in the conclusion. While this analysis is presented for completeness, these results are not utilised anywhere in the thesis. A separate benchmark was created (the baseline model) in the intent recognition experiments and that model was the basis for comparison in the intent recognition experiments. As such the intent recognition experiments are validated stand alone without relying on the results in this section.

A.2.1 Activity to class label mapping

A mapping of the activity labels to numeric class labels used in the confusion matrices are provided for each house. These mappings are identical to that used in the benchmark paper [162] and provide the source of reference for looking up activity labels when reading the confusion matrices. The activity label *GAP* in the mapping and result tables indicate time slices where the activity log indicates no activity recorded. To the best of our knowledge the resident is between different recorded activities in these time slices.

Table A.1: Activity to class labels for House A

Activity	**Class label**
GAP	0
leave house	1
use toilet	4
take shower	5
brush teeth	6
go to bed	10
prepare Breakfast	13
prepare Dinner	15
get snack	16
get drink	17
put items in dishwasher	18
unload dishwasher	19
store groceries	20
load washing machine	22
unload washing machine	23
receive guest	25

Table A.2: Activity to class labels for House B

Activity	**Class label**
GAP	0
Leaving the house	1
Use toilet	4
Take shower	5
Brush teeth	6
Shaving	9
Go to bed	10
Get dressed	11
Prepare brunch	13
Prepare dinner	15
MISSING LABEL	16
Get a drink	17
Wash dishes	24
Answering phone	29
Eat dinner	31
Eat brunch	32
Setting up sensors	33
Unpacking	34
Install sensor	35
On phone	36
Fasten kitchen camera	37
Wash toaster	38
Play piano	40
Gwenn searches keys	42
Prepare for leaving	43
Drop dish (No dishwash)	44

Table A.3: Activity to class labels for House C

Activity	**Class label**
GAP	0
leave house	1
Eating	3
use toilet downstairs	4
take shower	5
brush teeth	6
use toilet upstairs	7
shave	9
go to bed	10
get dressed	11
take medication	12
prepare Breakfast	13
prepare Lunch	14
prepare Dinner	15
get snack	16
get drink	17
load washing machine	22
relax	28

A.3 Results with the benchmark paper evaluation method

A.3.1 House A

The classification report of the hierarchical framework in predicting the current activity is shown in table A.5. The corresponding confusion matrix for the current activity classification results are shown in figure A.1.

The proposed framework achieves a macro average f1-score of **80.3 ± 13.4**.

The best result achieved for this house in the benchmark paper was an f1-score of **72.4 ± 13.7** as reported in Table 8.4 of [162]. This was achieved with *HSMM* model and macro averaging.

Table A.4: House A - Current activity classification benchmark comparison (paper evaluation)

model	**precision (%)**	**recall (%)**	**f1-score (%)**	**accuracy (%)**
NB	53 ± 18	43 ± 18	47 ± 17	56 ± 19
HMM	70 ± 16	74 ± 13	72 ± 14	92 ± 6
HSMM	71 ± 16	75 ± 12	72 ± 14	92 ± 6
CRF	74 ± 17	68 ± 16	70 ± 16	91 ± 6
Thesis	88 ± 7	84 ± 13	**80 ± 13**	64 ± 28

Table A.5: House A - Current activity results from hierarchical model (paper evaluation)

current activity	**precision (%)**	**recall (%)**	**f1-score (%)**
macro average	88 ± 7	84 ± 13	80 ± 13
weighted average	98 ± 6	64 ± 28	68 ± 26

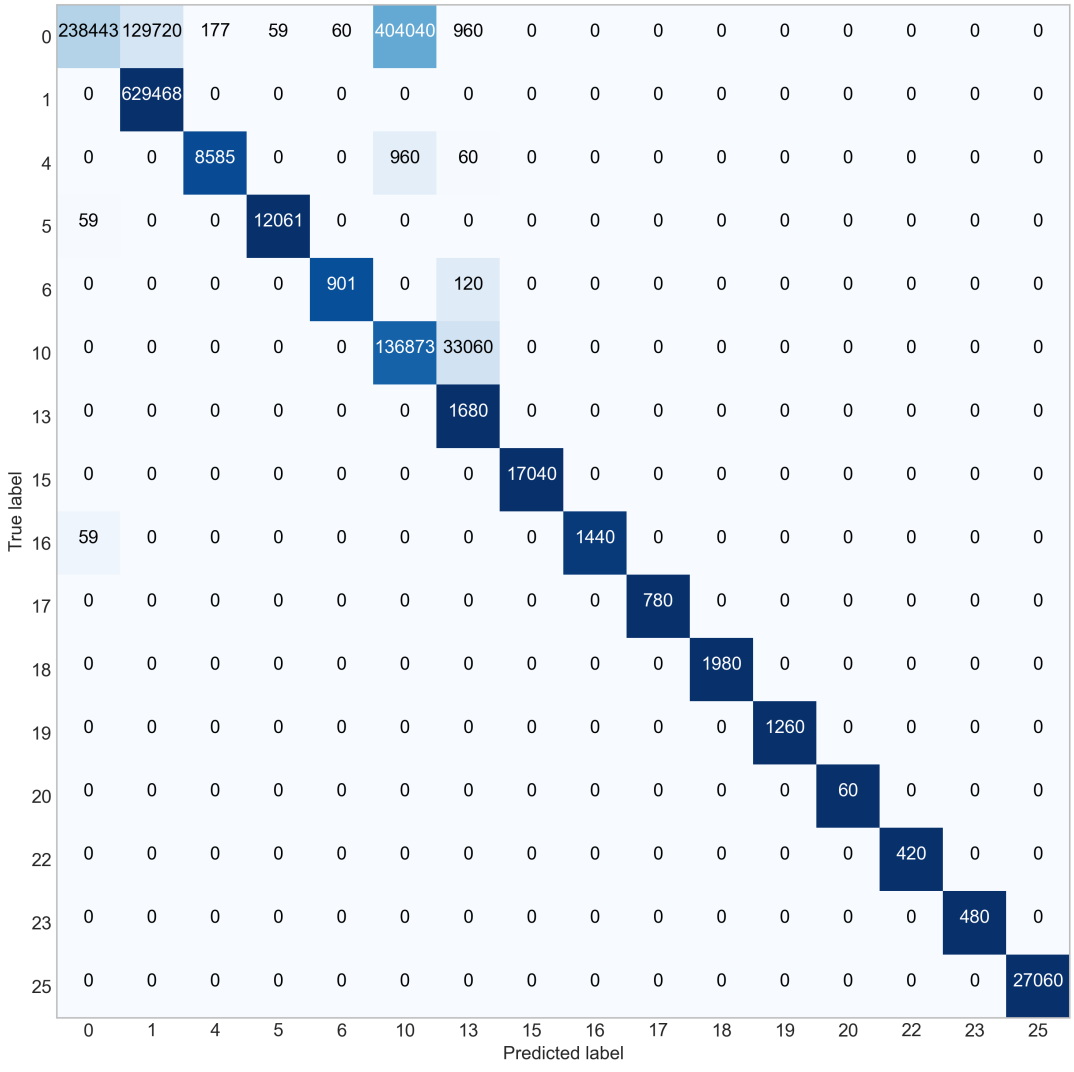


Figure A.1: House A - Confusion Matrix for Current activity prediction (hierarchical model with paper evaluation scheme)

A.3.2 House B

The classification report of the hierarchical framework in predicting the current activity is shown in table A.7. The corresponding confusion matrix for the current activity classification results are shown in figure A.2.

The proposed framework achieves a macro average f1-score of **87.5 ± 15.9**.

The best result achieved for this house in the benchmark paper was an f1-score of **55.7 ± 14.6** as reported in Table 8.5 of [162]. This was achieved with *HSMM* model and macro averaging.

There was one recorded activity in House B for which a label was not available. This activity is recorded in the table with *MISSING LABEL*.

Table A.6: House B - Current activity classification benchmark comparison (paper evaluation)

model	**precision (%)**	**recall (%)**	**f1-score (%)**	**accuracy (%)**
NB	41 ± 7	39 ± 6	40 ± 6	68 ± 19
HMM	48 ± 17	63 ± 14	54 ± 17	81 ± 14
HSMM	50 ± 16	65 ± 13	56 ± 15	82 ± 14
CRF	48 ± 8	52 ± 9	50 ± 8	93 ± 6
Thesis	93 ± 9	92 ± 9	**88 ± 16**	78 ± 31

Table A.7: House B - Current activity results from hierarchical model with paper evaluation

current activity	**precision (%)**	**recall (%)**	**f1-score (%)**
macro average	93 ± 9	92 ± 9	88 ± 16
weighted average	98 ± 3	78 ± 31	79 ± 29

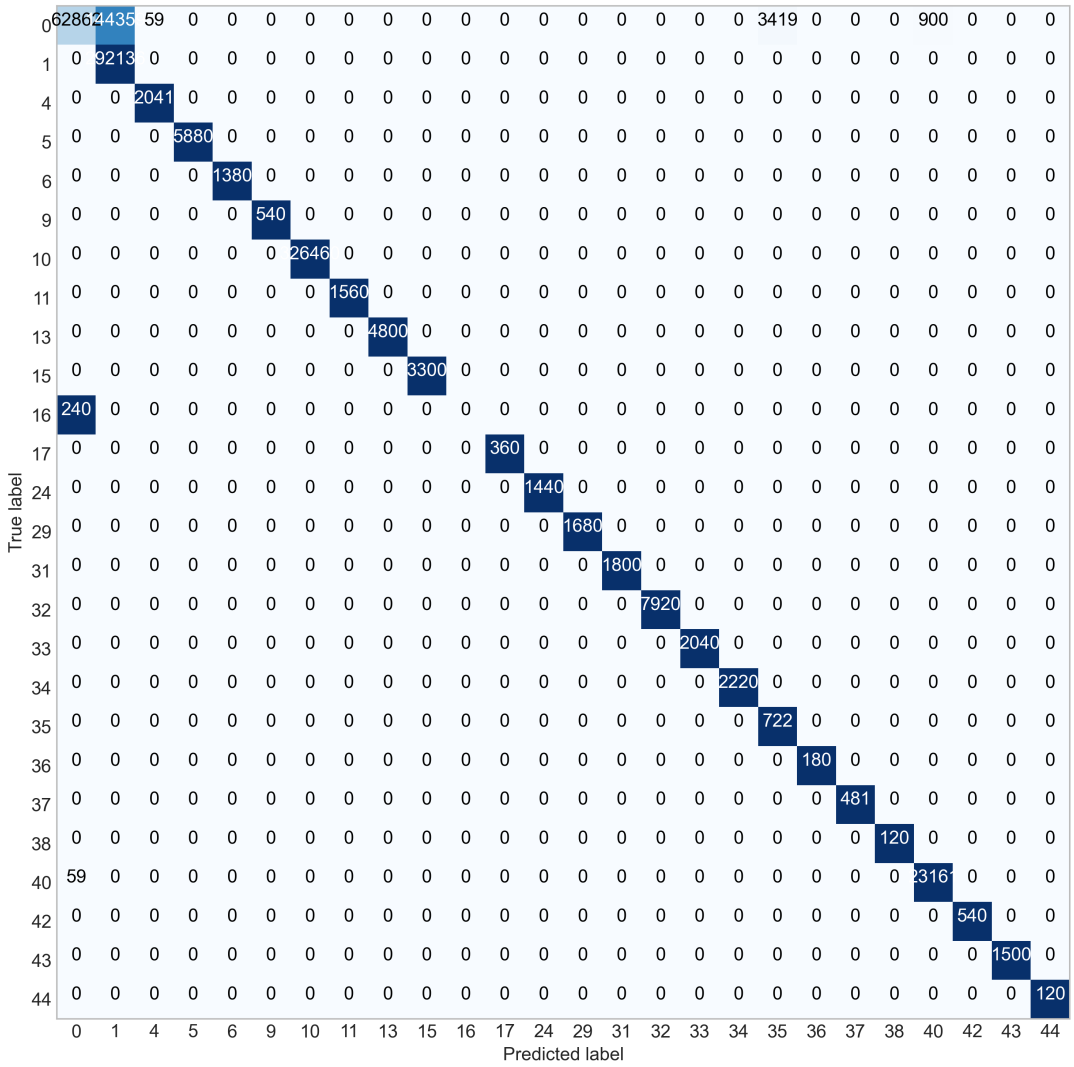


Figure A.2: House B - Confusion Matrix for Current activity prediction (hierarchical model with paper evaluation scheme)

A.3.3 House C

The classification report of the hierarchical framework in predicting the current activity is shown in table A.9. The corresponding confusion matrix for the current activity classification results are shown in figure A.3.

The proposed framework achieves a macro average f1-score of **91.3 ± 14.5**.

The best result achieved for this house in the benchmark paper was an f1-score of **47.9% ± 11.3%** as reported in Table 8.6 of [162]. This was achieved with *HSMM* model and macro averaging.

Table A.8: House C - Current activity classification benchmark comparison

model	**precision (%)**	**recall (%)**	**f1-score (%)**	**accuracy (%)**
NB	40 ± 7	31 ± 5	35 ± 5	58 ± 15
HMM	41 ± 9	50 ± 11	45 ± 9	77 ± 15
HSMM	44 ± 10	52 ± 13	47 ± 11	78 ± 15
CRF	37 ± 18	40 ± 17	38 ± 18	82 ± 14
Thesis	95 ± 4	92 ± 21	**91 ± 21**	89 ± 21

Table A.9: House C - Current activity results from hierarchical model with paper evaluation

current activity	**precision (%)**	**recall (%)**	**f1-score (%)**
macro average	95 ± 8	92 ± 14	91 ± 15
weighted average	97 ± 4	89 ± 21	89 ± 21

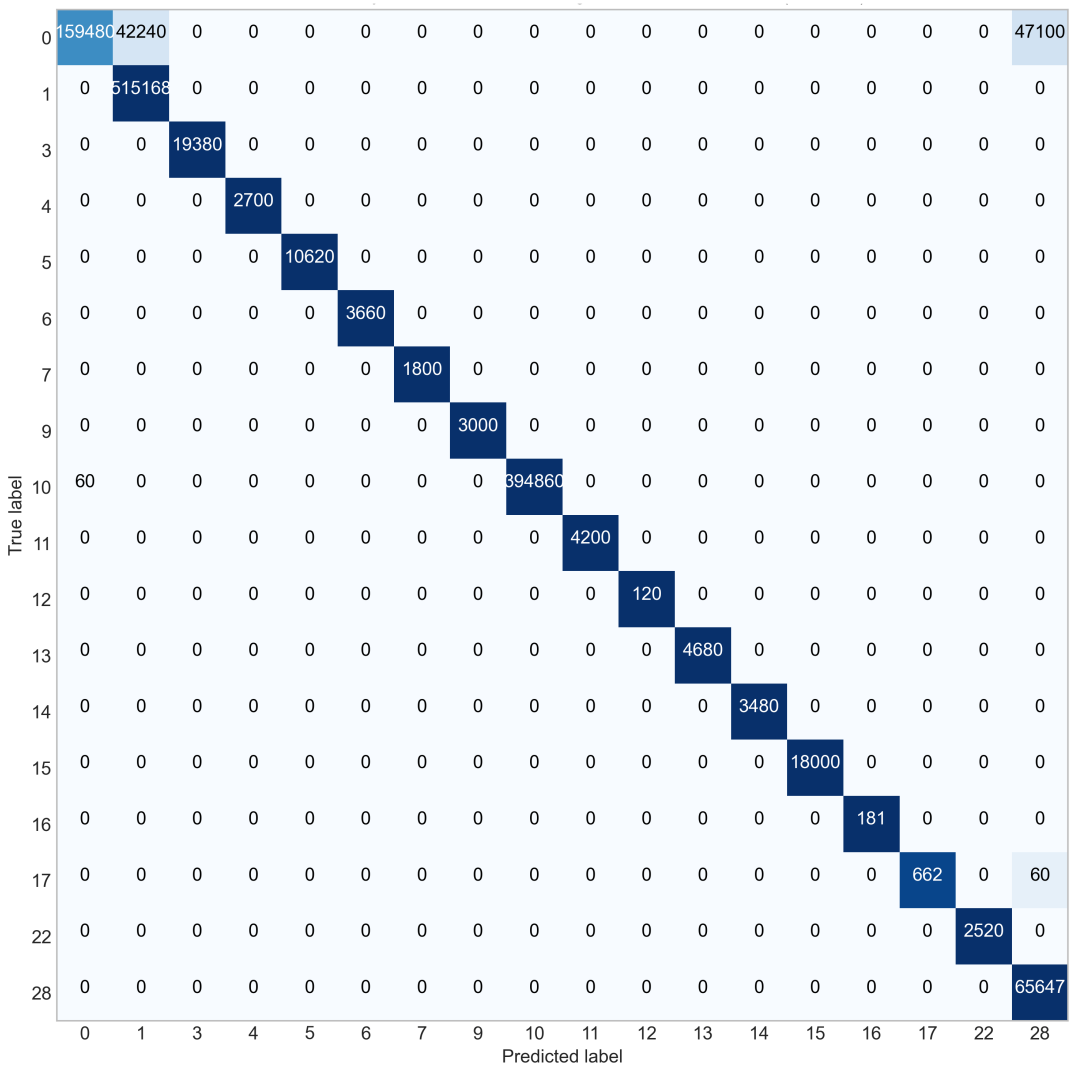


Figure A.3: House C - Confusion Matrix for Current activity prediction (hierarchical model with paper evaluation scheme)

A.4 Results with the thesis evaluation method

A.4.1 House A

Table A.10: House A - Current activity results from hierarchical model with thesis evaluation

current activity	precision (%)	recall (%)	f1-score (%)
macro average	38 ± 14	35 ± 11	33 ± 10
weighted average	79 ± 14	73 ± 14	74 ± 15

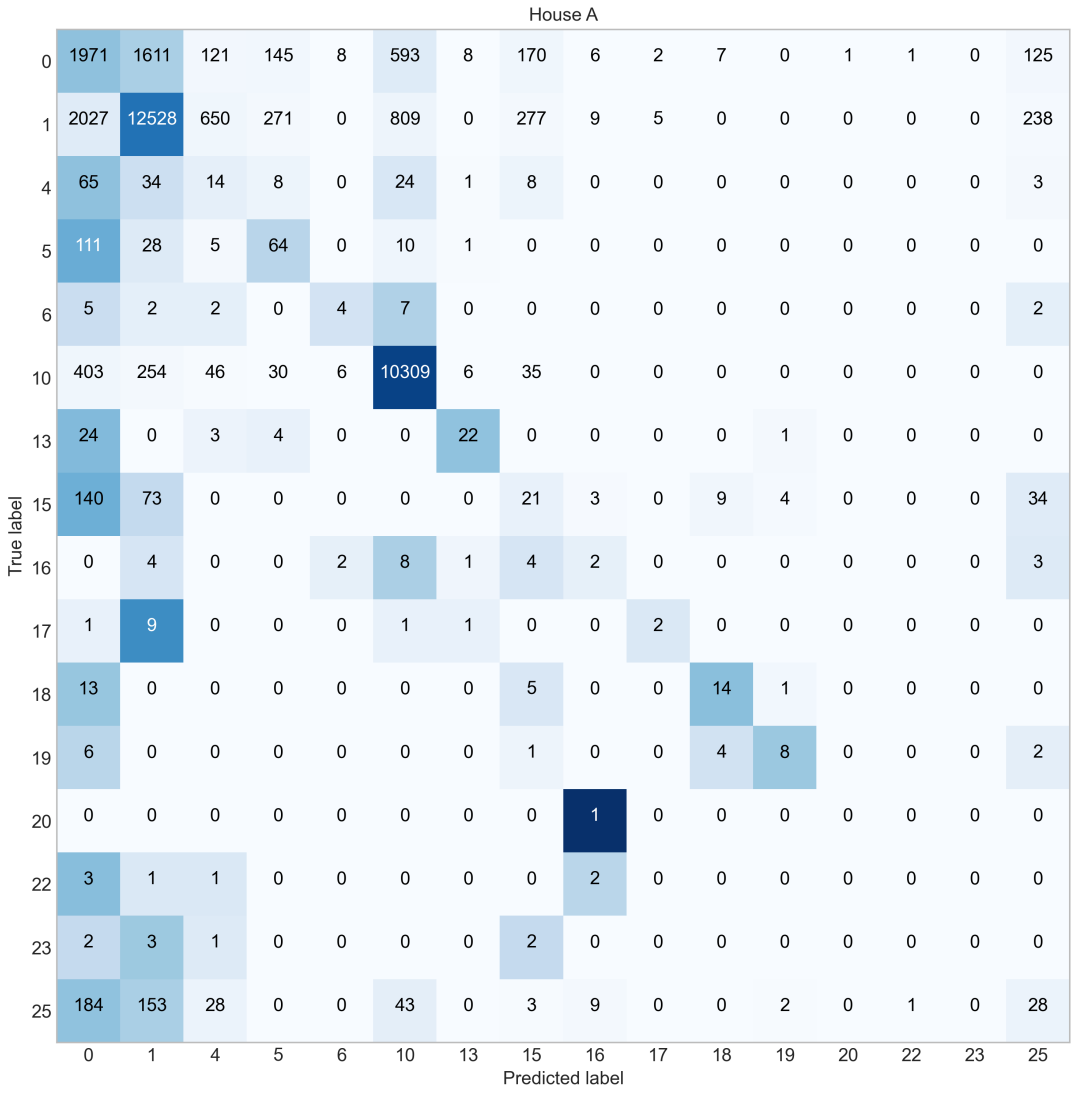


Figure A.4: House A - Confusion Matrix for Current activity prediction (hierarchical model with thesis evaluation scheme)

A.4.2 House B

Table A.11: House B - Current activity results from hierarchical model with thesis evaluation

current activity	precision (%)	recall (%)	f1-score (%)
macro average	33 ± 11	32 ± 11	30 ± 11
weighted average	84 ± 20	78 ± 21	79 ± 21

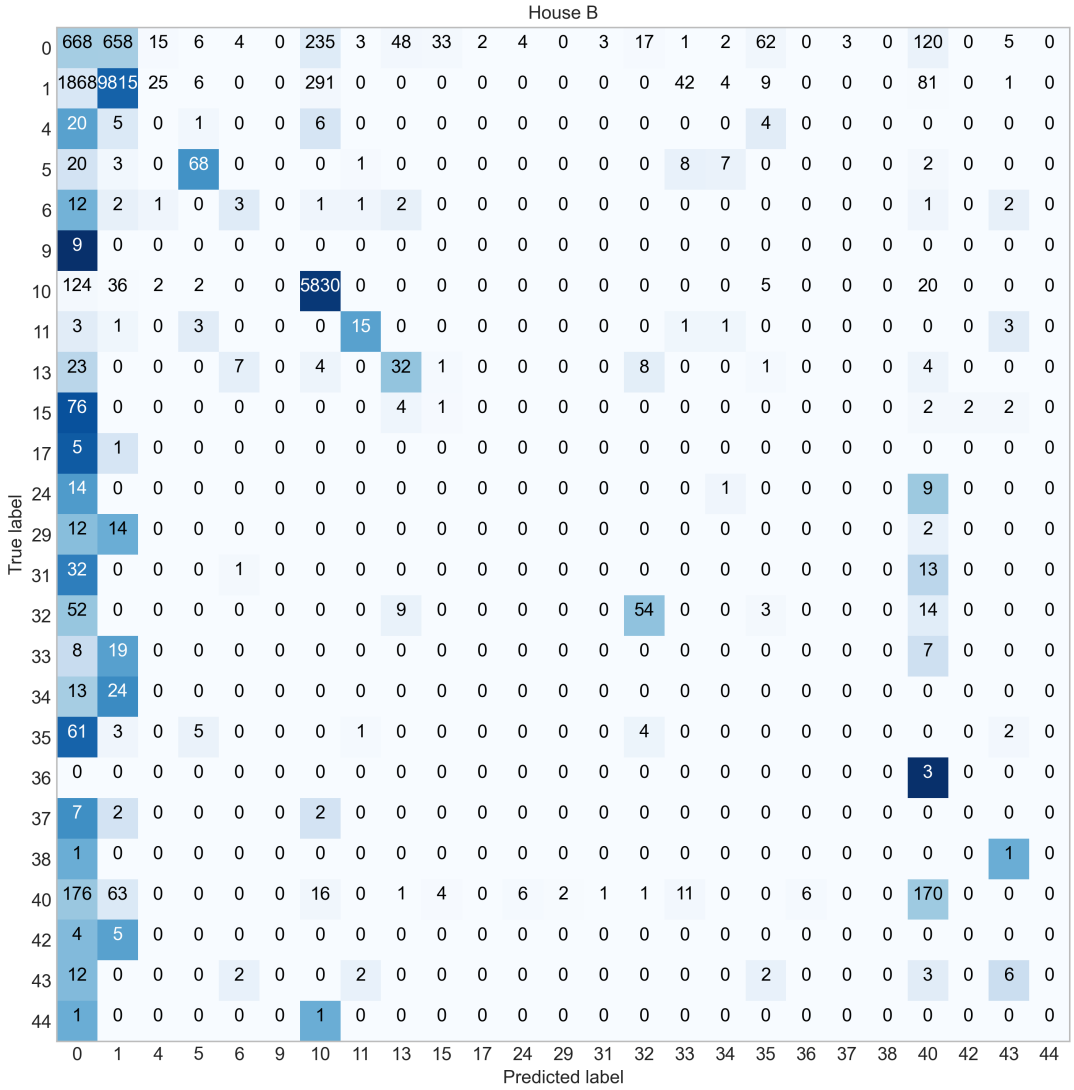


Figure A.5: House B - Confusion Matrix for Current activity prediction (hierarchical model with thesis evaluation scheme)

A.4.3 House C

Table A.12: House C - Current activity results from hierarchical model with thesis evaluation

current activity	**precision (%)**	**recall (%)**	**f1-score (%)**
macro average	32 ± 8	31 ± 7	29 ± 6
weighted average	84 ± 6	74 ± 8	76 ± 7

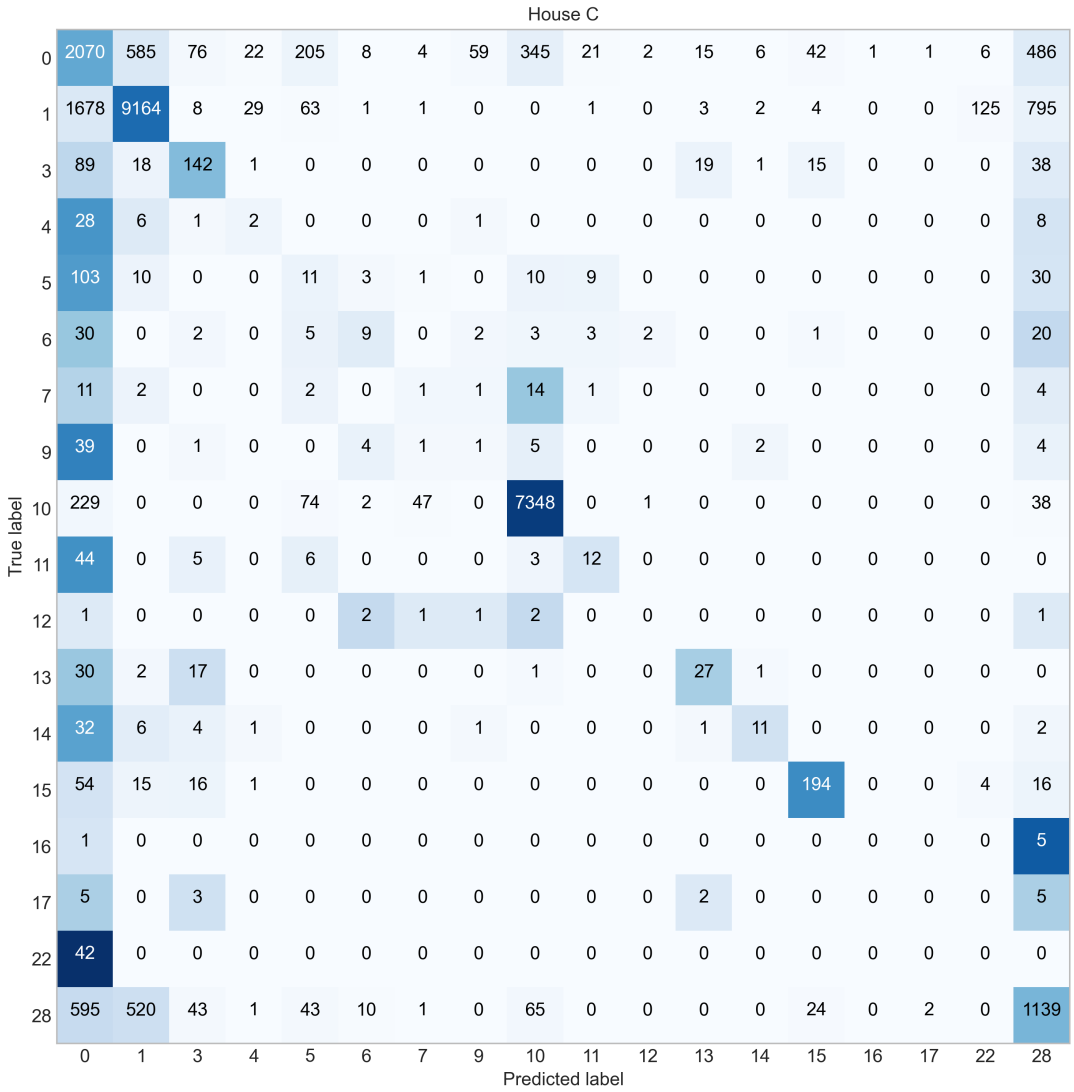


Figure A.6: House C - Confusion Matrix for Current activity prediction (hierarchical model with thesis evaluation scheme)

A.5 Comparing the evaluation schemes

Tables A.13, A.14 and A.15 report the f1 scores with macro and weighted averaging to provide a side-by-side comparison of the changes in moving from one evaluation scheme to the next.

Table A.13: House A - Evaluation scheme comparison

Metric	Paper Evaluation	Thesis Evaluation
Macro Average (%)	80 ± 13	33 ± 10
Weighted Average (%)	68 ± 26	74 ± 15

Table A.14: House B - Evaluation scheme comparison

Metric	Paper Evaluation	Thesis Evaluation
Macro Average (%)	88 ± 16	30 ± 11
Weighted Average (%)	79 ± 29	79 ± 21

Table A.15: House C - Evaluation scheme comparison

Metric	Paper Evaluation	Thesis Evaluation
Macro Average (%)	91 ± 15	29 ± 6
Weighted Average (%)	89 ± 21	76 ± 7

Comparing the two evaluation schemes, it is seen that the macro average is significantly boosted by the benchmark paper evaluation scheme whereas the weighted average is comparable and not significantly different.

This may be attributed to the fact that the approach of giving the classifier "60 tries to get it right" will have an outsized impact on the rare activities and therefore have a bigger impact on the macro average than the weighted average.

A.6 Conclusion

It is observed that the paper evaluation scheme provides a major lift on the macro average figures. While this was the adopted approach in the benchmark paper, in this thesis that evaluation scheme is not adopted for the intent recognition experiments. The paper evaluation scheme produces some very surprising results which are not easily explainable or logical.

Therefore, to support and validate all the results in Implementation Example I for intent recognition (chapter 4), a separate baseline model was created. The baseline model still predicts current activity and then assumes that to be the user's intent (i.e.: the baseline model assumes the users intention is to continue in their current activity until the intent horizon). In chapter 4, the baseline model is evaluated consistently with the experimental models using the thesis evaluation scheme. Therefore, one can be confident that comparisons are being made fairly and no favour or advantage is given to any model in the evaluation.

APPENDIX B

Copyright license

The layouts of the 3 houses in *Implementation Example I: Predicting activities of daily living* (figures 4.1, 4.2, 4.3 and 4.4 in this thesis) are reproduced by permission from *Springer Nature* under license.

The figures originally appeared in:
Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software, van Kasteren T.L.M., Englebienne G., Kröse B.J.A. (2011)

A chapter in:
Activity Recognition in Pervasive Intelligent Environments,
Chen L., Nugent C., Biswas J., Hoey J. (eds)

Part of:
Atlantis Ambient and Pervasive Intelligence book series, vol 4, Atlantis Press and also a *Springer eBook*.

SPRINGER NATURE LICENSE
TERMS AND CONDITIONS

Mar 17, 2024

This Agreement between Ali Naeem ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

License Number	5751601333503
License date	Mar 17, 2024
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Human Activity Recognition from Wireless Sensor Network Data: Benchmark and Software
Licensed Content Author	T. L. M. van Kasteren, G. Englebienne, B. J. A. Kröse
Licensed Content Date	Jan 1, 2011
Type of Use	Thesis/Dissertation
Requestor type	academic/university or research institute
Format	print and electronic
Portion	figures/tables/illustrations
Number of figures/tables/illustrations	3

Will you be translating?	no
Circulation/distribution	1 - 29
Author of this Springer Nature content	no
Title of new work	Intent Recognition of Autonomous Agents
Institution name	University College Cork
Expected presentation date	Apr 2024
Portions	Figures 8.1, 8.2 and 8.3.
Requestor Location	Ali Azzam Naeem 450 West 33rd St NEW YORK, NY 10001 United States Attn: Ali Azzam Naeem
Total	0.00 USD

Terms and Conditions

Springer Nature Customer Service Centre GmbH Terms and Conditions

The following terms and conditions ("Terms and Conditions") together with the terms specified in your [RightsLink] constitute the License ("License") between you as Licensee and Springer Nature Customer Service Centre GmbH as Licensor. By clicking 'accept' and completing the transaction for your use of the material ("Licensed Material"), you confirm your acceptance of and obligation to be bound by these Terms and Conditions.

1. Grant and Scope of License

1. 1. The Licensor grants you a personal, non-exclusive, non-transferable, non-sublicensable, revocable, world-wide License to reproduce, distribute, communicate to the public, make available, broadcast, electronically transmit or create derivative works using the Licensed Material for the purpose(s) specified in your RightsLink Licence Details only. Licenses are granted for the specific use requested in the order

and for no other use, subject to these Terms and Conditions. You acknowledge and agree that the rights granted to you under this License do not include the right to modify, edit, translate, include in collective works, or create derivative works of the Licensed Material in whole or in part unless expressly stated in your RightsLink Licence Details. You may use the Licensed Material only as permitted under this Agreement and will not reproduce, distribute, display, perform, or otherwise use or exploit any Licensed Material in any way, in whole or in part, except as expressly permitted by this License.

1. 2. You may only use the Licensed Content in the manner and to the extent permitted by these Terms and Conditions, by your RightsLink Licence Details and by any applicable laws.

1. 3. A separate license may be required for any additional use of the Licensed Material, e.g. where a license has been purchased for print use only, separate permission must be obtained for electronic re-use. Similarly, a License is only valid in the language selected and does not apply for editions in other languages unless additional translation rights have been granted separately in the License.

1. 4. Any content within the Licensed Material that is owned by third parties is expressly excluded from the License.

1. 5. Rights for additional reuses such as custom editions, computer/mobile applications, film or TV reuses and/or any other derivative rights requests require additional permission and may be subject to an additional fee. Please apply to journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

2. Reservation of Rights

Licensor reserves all rights not expressly granted to you under this License. You acknowledge and agree that nothing in this License limits or restricts Licensor's rights in or use of the Licensed Material in any way. Neither this License, nor any act, omission, or statement by Licensor or you, conveys any ownership right to you in any Licensed Material, or to any element or portion thereof. As between Licensor and you, Licensor owns and retains all right, title, and interest in and to the Licensed Material subject to the license granted in Section 1.1. Your permission to use the Licensed Material is expressly conditioned on you not impairing Licensor's or the applicable copyright owner's rights in the Licensed Material in any way.

3. Restrictions on use

3. 1. Minor editing privileges are allowed for adaptations for stylistic purposes or formatting purposes provided such alterations do not alter the original meaning or intention of the Licensed Material and the new figure(s) are still accurate and representative of the Licensed Material. Any other changes including but not limited to, cropping, adapting, and/or omitting material that affect the meaning, intention or moral rights of the author(s) are strictly prohibited.

3. 2. You must not use any Licensed Material as part of any design or trademark.

3. 3. Licensed Material may be used in Open Access Publications (OAP), but any such reuse must include a clear acknowledgment of this permission visible at the same time as the figures/tables/illustration or abstract and which must indicate that

the Licensed Material is not part of the governing OA license but has been reproduced with permission. This may be indicated according to any standard referencing system but must include at a minimum 'Book/Journal title, Author, Journal Name (if applicable), Volume (if applicable), Publisher, Year, reproduced with permission from SNCSC'.

4. STM Permission Guidelines

4. 1. An alternative scope of license may apply to signatories of the STM Permissions Guidelines ("STM PG") as amended from time to time and made available at https://www.stm-assoc.org/intellectual-property/permissions/permissions-guidelines/.

4. 2. For content reuse requests that qualify for permission under the STM PG, and which may be updated from time to time, the STM PG supersede the terms and conditions contained in this License.

4. 3. If a License has been granted under the STM PG, but the STM PG no longer apply at the time of publication, further permission must be sought from the Rightsholder. Contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

5. Duration of License

5. 1. Unless otherwise indicated on your License, a License is valid from the date of purchase ("License Date") until the end of the relevant period in the below table:

Reuse in a medical communications project	Reuse up to distribution or time period indicated in License
Reuse in a dissertation/thesis	Lifetime of thesis
Reuse in a journal/magazine	Lifetime of journal/magazine
Reuse in a book/textbook	Lifetime of edition
Reuse on a website	1 year unless otherwise specified in the License
Reuse in a presentation/slide kit/poster	Lifetime of presentation/slide kit/poster. Note: publication whether electronic or in print of presentation/slide kit/poster may require further permission.
Reuse in conference proceedings	Lifetime of conference proceedings
Reuse in an annual report	Lifetime of annual report
Reuse in training/CME materials	Reuse up to distribution or time period indicated in License
Reuse in newsmedia	Lifetime of newsmedia
Reuse in coursepack/classroom materials	Reuse up to distribution and/or time period indicated in license

6. Acknowledgement

6. 1. The Licensor's permission must be acknowledged next to the Licensed Material in print. In electronic form, this acknowledgement must be visible at the same time as the figures/tables/illustrations or abstract and must be hyperlinked to the journal/book's homepage.

6. 2. Acknowledgement may be provided according to any standard referencing system and at a minimum should include "Author, Article/Book Title, Journal name/Book imprint, volume, page number, year, Springer Nature".

7. Reuse in a dissertation or thesis

7. 1. Where 'reuse in a dissertation/thesis' has been selected, the following terms apply: Print rights of the Version of Record are provided for; electronic rights for use only on institutional repository as defined by the Sherpa guideline (www.sherpa.ac.uk/romeo/) and only up to what is required by the awarding institution.

7. 2. For theses published under an ISBN or ISSN, separate permission is required. Please contact journalpermissions@springernature.com or bookpermissions@springernature.com for these rights.

7. 3. Authors must properly cite the published manuscript in their thesis according to current citation standards and include the following acknowledgement: '*Reproduced with permission from Springer Nature*'.

8. License Fee

You must pay the fee set forth in the License Agreement (the "License Fees"). All amounts payable by you under this License are exclusive of any sales, use, withholding, value added or similar taxes, government fees or levies or other assessments. Collection and/or remittance of such taxes to the relevant tax authority shall be the responsibility of the party who has the legal obligation to do so.

9. Warranty

9. 1. The Licensor warrants that it has, to the best of its knowledge, the rights to license reuse of the Licensed Material. **You are solely responsible for ensuring that the material you wish to license is original to the Licensor and does not carry the copyright of another entity or third party (as credited in the published version).** If the credit line on any part of the Licensed Material indicates that it was reprinted or adapted with permission from another source, then you should seek additional permission from that source to reuse the material.

9. 2. EXCEPT FOR THE EXPRESS WARRANTY STATED HEREIN AND TO THE EXTENT PERMITTED BY APPLICABLE LAW, LICENSOR PROVIDES THE LICENSED MATERIAL "AS IS" AND MAKES NO OTHER REPRESENTATION OR WARRANTY. LICENSOR EXPRESSLY DISCLAIMS ANY LIABILITY FOR ANY CLAIM ARISING FROM OR OUT OF THE CONTENT, INCLUDING BUT NOT LIMITED TO ANY ERRORS, INACCURACIES, OMISSIONS, OR DEFECTS CONTAINED THEREIN, AND ANY IMPLIED OR EXPRESS WARRANTY AS TO MERCHANTABILITY OR

FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT SHALL LICENSOR BE LIABLE TO YOU OR ANY OTHER PARTY OR ANY OTHER PERSON OR FOR ANY SPECIAL, CONSEQUENTIAL, INCIDENTAL, INDIRECT, PUNITIVE, OR EXEMPLARY DAMAGES, HOWEVER CAUSED, ARISING OUT OF OR IN CONNECTION WITH THE DOWNLOADING, VIEWING OR USE OF THE LICENSED MATERIAL REGARDLESS OF THE FORM OF ACTION, WHETHER FOR BREACH OF CONTRACT, BREACH OF WARRANTY, TORT, NEGLIGENCE, INFRINGEMENT OR OTHERWISE (INCLUDING, WITHOUT LIMITATION, DAMAGES BASED ON LOSS OF PROFITS, DATA, FILES, USE, BUSINESS OPPORTUNITY OR CLAIMS OF THIRD PARTIES), AND WHETHER OR NOT THE PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THIS LIMITATION APPLIES NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY PROVIDED HEREIN.

10. Termination and Cancellation

10. 1. The License and all rights granted hereunder will continue until the end of the applicable period shown in Clause 5.1 above. Thereafter, this license will be terminated and all rights granted hereunder will cease.

10. 2. Licensor reserves the right to terminate the License in the event that payment is not received in full or if you breach the terms of this License.

11. General

11. 1. The License and the rights and obligations of the parties hereto shall be construed, interpreted and determined in accordance with the laws of the Federal Republic of Germany without reference to the stipulations of the CISG (United Nations Convention on Contracts for the International Sale of Goods) or to Germany´s choice-of-law principle.

11. 2. The parties acknowledge and agree that any controversies and disputes arising out of this License shall be decided exclusively by the courts of or having jurisdiction for Heidelberg, Germany, as far as legally permissible.

11. 3. This License is solely for Licensor's and Licensee's benefit. It is not for the benefit of any other person or entity.

Questions? For questions on Copyright Clearance Center accounts or website issues please contact springernaturesupport@copyright.com or +1-855-239-3415 (toll free in the US) or +1-978-646-2777. For questions on Springer Nature licensing please visit https://www.springernature.com/gp/partners/rights-permissions-third-party-distribution

Other Conditions:

Version 1.4 - Dec 2022