Log-based generation of event labels using event attributes

Alasgarov, E.E.

Award date:
2016

Link to publication
Log-based Generation of Event Labels using Event Attributes

Master Thesis

Emin Alasgarov

Supervisors:
Dr. Natalia Sidorova
Dr. ir. Reinder Haakma
Niek Tax, MSc

Assessment committee member:
Prof. Dr. P.M.E. De Bra

Eindhoven, 31st August 2016
Final version
Abstract

A prompt increase in the number of sensor devices, and the progress in sensor data analytics have led to new opportunities as well as challenges. One such challenge is to acquire and analyze life-log data of human activities. The evolution of ubiquitous computing environments and devices has facilitated the recognition of the life-log concept, which addresses the tracking of behavioral data with the help of sensor devices. Many technologies exist these days that allow to generate and store life-logs produced by humans (e.g. wearable devices, sensor-equipped smart-homes). Such real-life data logs are then analyzed by researchers in order to identify behavioral commonalities or discrepancies throughout a particular life-cycle. Despite the fact that traditional activity recognition models have considerably contributed to the understanding and analysis of human behavior within particular application domains, it is important to state that these solutions lack of representational power (i.e. visual assessment and interpretation) when it comes to a description of real-life end-to-end process models of human activities.

In the field of process mining, much work has been done on analysis of events that represent real-life activities within a normative set of rules. Mining of meaningful and reliable process models can be considered a relatively convenient task when applied on event logs obtained from classical business management systems (e.g. work-flow management or transactions systems). Major reason behind this is a traditional event labeling (i.e. the labels of events in such systems are known/clear) used by organizations to differentiate business process activities. On the contrary, life-log events obtained from sensor devices present a challenging application domain for the process discovery algorithms in the absence of a complementary contextual information (e.g. time difference between events, location, duration). The structuredness of the process models that can be discovered from sensor outputs is highly dependent on how the outputs are translated into events. In particular, the identification of a specific event types and assignment of appropriate labels (i.e. events that belong to the same type of activity are refined with a same label) to them are still considered to be a relatively complex problem.

This thesis discusses the autonomous generation of context-based event label refinements or abstractions of smart-home life-logs using event timestamps. Our major contribution is a developed and presented label refinement framework, which incorporates a set of statistical inference methods required to perform a reliable sensor event analysis, and produce a set of possible refinement candidates per available sensor type. In this work, we have shown that starting and duration times of smart-home sensor events derived from their timestamp attribute may establish a proper feature domain for the application of probabilistic mixture model clustering in order to identify repetitive patterns of human behavior in autonomous and personalized manner. This unsupervised machine learning technique constitutes the backbone of our label refinement framework, and based on the observation that every single data point (in our case expressed in form of starting and duration times) within a given population can be described by a parametrized probability density function from which it might be drawn. We have identified and analyzed the importance of selecting a correct density function, by starting from Gaussian mixture modeling for linear data (i.e. duration times of sensor events) and making a transition towards von Mises mixture modeling for circular data (i.e. starting times of sensor events).
In order to demonstrate the applicability of our label refinement framework, two case studies have been performed by using a smart-home event log data from two different sources. During the first case study, publicly available van Kasteren dataset has been utilized, and discovered process models from the original and refined event logs have been presented. The obtained results show that the application of our label refinement method on a particular sensor types allows to discover more structured process models, which provide informative insights for the analysis of human behavior within the smart-home environment. In order to highlight the importance of the sensor event refinement at the event log level, our unsupervised approach has been compared to the data-aware process mining with transition guards method, which illustrated the limitations of the latter solution by considering the abstract sensor labels of the original (i.e unprocessed) smart-home event log. During the second case study, more sophisticated private Philips dataset has been utilized, following the same evaluation procedures in order to demonstrate the usefulness of our framework. Despite the increasing number of sensor events, we have still succeeded to show that it is possible to obtain more meaningful process models by pre-processing original event log using our label refinement method. Finally, using a new label refinement evaluation method that has been proposed by researchers of TU/e and Philips Research, we have shown that for every single sensor type, its refined label candidates may differ not only based on a timestamp, but also based on how significantly they maintain the ordering relations (e.g. directly follows, or eventually precedes) in respect to other available sensor types. This information is important for assessing the commonalities and discrepancies within a set of refined candidates (per sensor type), and their effect of the structure of a discovered process model.
Preface

This master thesis is the result of my graduation project for the Computer Science and Engineering master at Eindhoven University of Technology (TU/e), and has been carried out within Philips Research and Architecture of Information Systems (AIS) group of the Mathematics and Computer Science department of TU/e. This work concludes a wonderful scientific journey that has been experienced by me for the period of two years in one of the best European IT universities.

First of all, I would like express my sincere gratitude to my supervisor at TU/e, Dr. Natalia Sidorova for her thorough guidance, critical assessment, and constant encouragement throughout the whole duration of this thesis project. I would like to thank Dr. Sidorova for the amount of knowledge and experience that she has shared with me for the last six month.

Second, I would like to thank my supervisor at Philips Research, Dr. ir. Reinder Haakma for the created opportunity to pursue my thesis project in one of the smartest regions within the Europe. I am grateful for every productive meeting we had, which allowed me to sharpen my analytical thinking in a required direction for the successful completion of this work.

Third, I would like to thank my mentor, Niek Tax who is a PhD candidate within the AIS group at TU/e. From the very first day of this master project he shaped the course of my research domain, pointing out crucial aspects, and helping to define research priorities. Thank you for your sense of humor, availability and dedicated time. I wish every master student gets assigned a mentor like you.

Fourth, I would like to express my deepest gratitude to my family, for being patient and inspirational. Thank you for teaching me how to strive for a brighter future, by assimilating and analysing every day knowledge, and converting it into a valuable life experience.

I would like also to thank Prof. Dr. P.M.E. De Bra for joining my final assessment committee. Thank you for your time and interest.

Finally, I would like to thank The Netherlands, especially the city of Eindhoven for amazing adventures, kind people and obtained friends. Thank you, Berkay, Peter, Boris, Sakina, Ilaha, Shamil, Yashar, Michael, Jonas, Nan, Dato, Bin, Kirsten, Ahsan, Ehsan, Vladimir, Taha, Hilda, Khalid, Rudi!

Emin Alasgarov, July, 2016
Contents

List of Figures ix
List of Tables xiii

1 Introduction 1
1.1 Thesis Context 1
1.2 Problem Statement 2
1.3 Research Question 6
1.4 Research Method and Outline 8

2 Preliminaries 9
2.1 Process Mining 9
2.1.1 Event Log 10
2.1.2 Quality Metrics: Precision versus Generalization 12
2.1.3 Evaluation Method for Label Refinements 13
2.2 Literature Study - Label Refinement 15
2.3 Unsupervised versus Supervised Machine Learning 16
2.4 Clustering Method Selection 18
2.4.1 Probabilistic Mixture Models 18
2.4.2 Maximum Likelihood Estimate using EM Algorithm 18

3 Label Refinement Method based on a Timestamp 21
3.1 Preliminary Analysis of Smart-Home Sensor Data 21
3.2 Circular Nature of a Timestamp Attribute 25
3.2.1 Clustering ‘Duration’ of Smart-home Sensor Events 25
3.2.2 Clustering ‘Starting-Time’ of Smart-home Sensor Events 27
3.3 Clustering using von Mises Distribution 29

4 Implemented Framework 32
4.1 Data-Model Pre-Fitting Stage 33
4.1.1 Uniformity Check - Rao’s Spacing Test 34
4.1.2 Multimodality Check - Hartigan’s Dip Test 35
4.1.3 Number of Component Selection - Bayesian Information Criteria 36
4.2 Data-Model Fitting Stage 39
4.2.1 Data pre-processing for a circular data 39
4.2.2 Fitting a Mixture of von Mises Distributions 40
4.2.3 Resolution of an Ambiguous Sensor Events 42
4.3 Data-Model Post-Fitting Stage 46
4.3.1 Visual Assessment using P-P and Q-Q Plots 47
4.3.2 Watson $U^2$ Statistics 49

Log-based Generation of Event Labels using Event Attributes vii
## 5 Case Studies  
5.1 Datasets Used  
5.1.1 van Kasteren Dataset  
5.1.2 Philips Dataset  
5.2 Case 1 - van Kasteren Dataset  
5.2.1 Event log pre-processing  
5.2.2 Petri Net Mined from the Unrefined Event Log  
5.2.3 Application of the Label Refinement Method  
5.2.4 Petri Net Mined from the Refined Event Log  
5.2.5 Comparison to the Data-Aware Process Mining with Transition Guards  
5.3 Case 2 - Philips Dataset  
5.3.1 Event log pre-processing  
5.3.2 Petri Net Mined from the Unrefined Event Log  
5.3.3 Application of the Label Refinement Method  
5.3.4 Petri Net Mined from the Refined Event Log  

## 6 Conclusion  
6.1 Limitations and Future Work  
6.1.1 Concept Drift  
6.1.2 Combination of Sensor Event Types  
6.1.3 Multidimensional Feature Domain  
6.1.4 Iterative Refinement of Sensor Event Types  

Bibliography  
Appendix  
A Processing Philips Dataset  
B Statistical Testing Results
## List of Figures

1.1 Business Process Management System Event Log ............................................. 3
1.2 The process model discovered by applying process discovery algorithm on event log represented by Figure 1.1 ................................................................. 4
1.3 Smart Home Event Log ................................................................................... 4
1.4 The process model discovered by applying process discovery algorithm on event log represented by Figure 1.3 ................................................................. 5
1.5 The process model discovered after applying label refinement on the original smart-home event log ................................................................. 5
1.6 Original sensor event log (upper) versus refined sensor event log (lower) ............ 7

2.1 Process Mining application domain .................................................................. 10
2.2 Structure of event logs within Process Mining domain ...................................... 11
2.3 Two types of process models that could be discovered from the same event log .... 12
2.4 Representation of control-flow without (\(M_1\)) and with (\(M_2\)) transition guards (BPMN notation) ................................................................. 16

3.1 Histogram analysis of smart-home sensor events from the van Kasteren dataset (23 days) ........................................................................................................ 22
3.2 Van Kasteren dataset: standard occurrence of the 'Fridge' sensor event (23 days) 22
3.3 Van Kasteren dataset: standard occurrence of the 'Plates Cupboard' sensor event (23 days) ........................................................................................................ 23
3.4 High-level clustering approach towards the label refinement generation .......... 24
3.5 Philips dataset: standard duration of the 'TV' sensor event (85 days) ................. 24
3.6 Clustering 'TV' sensor event duration using Gaussian Mixture Model (3 components) ................................. 26
3.7 Clustering 'Fridge' sensor event using Gaussian Mixture Model (2 components) ........................................................................................................ 27
3.8 Clustering 'Plates Cupboard' sensor event using Gaussian Mixture Model (2 components) ........................................................................................................ 27
3.9 Circular representation of the 'Fridge' (left) and 'Plates Cupboard' (right) sensor event types throughout the day for the observed period of time (23 days) .......... 29
3.10 Density of the von Mises distribution \(M(\theta, \kappa)\) for \(\kappa = (0, 0.5, 1, 2, 4, 8)\) ........ 30

4.1 Clustering steps performed for the generation of a label refinement method ........ 32
4.2 Uniform, Unimodal and Multimodal distribution types around the circle ............ 33
4.3 Three major statistical steps required to perform before fitting sensor data to a mixture of von Mises components ................................................................. 34
4.4 Rao’s Spacing Test for the 'Fridge' sensor event type ....................................... 34
4.5 Visualization of empirical cumulative distribution function for the 'Fridge' sensor event ('diptest') ................................................................. 35
4.6 Hartigan’s Dip Test for the 'Fridge' sensor event type ....................................... 35
4.7 Visualization of empirical cumulative distribution function for the unimodal von Mises distribution ................................................................................. 36
4.8 Hartigan’s Dip Test for the unimodal von Mises distribution .............................. 36
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.9</td>
<td>BIC change for the increasing number of von Mises mixture components - 'Plates Cupboard' (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.10</td>
<td>BIC values for the number of components from 1 to 9 - 'Plates Cupboard' (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.11</td>
<td>BIC change for the increasing number of von Mises mixture components - 'Hall-Toilet door' (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.12</td>
<td>BIC values for the number of components from 1 to 9 - 'Hall-Toilet door' (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.13</td>
<td>Transformation from Polar to Cartesian coordinates, 'Plates Cupboard' sensor event, van Kasteren dataset</td>
</tr>
<tr>
<td>4.14</td>
<td>Clustering the 'Plates Cupboard' sensor events by fitting two von Mises distributions (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.15</td>
<td>Probability assignment to the 'Plates Cupboard' sensor events by fitting two von Mises distributions (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.16</td>
<td>Clustering the 'Hall-Toilet door' sensor events by fitting three von Mises distributions (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.17</td>
<td>Clustering the 'Refrigerator' sensor events by fitting three von Mises distributions (Philips dataset)</td>
</tr>
<tr>
<td>4.18</td>
<td>Supervised model using decision trees to address ambiguous points at the intersection of two von Mises distributions</td>
</tr>
<tr>
<td>4.19</td>
<td>Graphical representation of the cross validated error to prune the decision tree in Figure 4.20 (low cross validated error (xerror) is preferred)</td>
</tr>
<tr>
<td>4.20</td>
<td>Decision tree classifier obtained based on the 'preceding' and 'following' features of clusters 2 and 3 for the 'Refrigerator' sensor event type (Philips dataset)</td>
</tr>
<tr>
<td>4.21</td>
<td>Performance statistics of the decision tree classification model (Figure 4.20)</td>
</tr>
<tr>
<td>4.22</td>
<td>ROC curve and AUC value for the generated classification model</td>
</tr>
<tr>
<td>4.23</td>
<td>Clustering the 'Frontdoor' sensor events by fitting two von Mises distributions (van Kasteren dataset)</td>
</tr>
<tr>
<td>4.24</td>
<td>P-P (left) and Q-Q (right) plots for the Cluster 1 - 'Frontdoor' sensor event type</td>
</tr>
<tr>
<td>4.25</td>
<td>P-P (left) and Q-Q (right) plots for the Cluster 2 - 'Frontdoor' sensor event type</td>
</tr>
<tr>
<td>5.1</td>
<td>Sample snapshot, Philips Dataset</td>
</tr>
<tr>
<td>5.2</td>
<td>Transformation of raw smart-home event log (left) into XES format (right) for process discovery</td>
</tr>
<tr>
<td>5.3</td>
<td>Log inspector for van Kasteren dataset (ProM6)</td>
</tr>
<tr>
<td>5.4</td>
<td>Petri Net Discovered from the Unrefined Event Log (van Kasteren event log)</td>
</tr>
<tr>
<td>5.5</td>
<td>BIC change for the increasing number of von Mises mixture components - 'Hall-Bedroom door' (van Kasteren dataset)</td>
</tr>
<tr>
<td>5.6</td>
<td>BIC values for the number of components from 1 to 9 - 'Hall-Bedroom door' (van Kasteren dataset)</td>
</tr>
<tr>
<td>5.7</td>
<td>Clustering the 'Hall-Bedroom door' sensor events by fitting two von Mises distributions (van Kasteren dataset)</td>
</tr>
<tr>
<td>5.8</td>
<td>Transformation of sensor events '24' ('Hall-Bedroom door') after applying label refinement method</td>
</tr>
<tr>
<td>5.9</td>
<td>Petri Net Discovered from the Refined Event Log (Sensor '24')</td>
</tr>
<tr>
<td>5.10</td>
<td>Petri Net Discovered from the Refined Event Log (Sensor '12')</td>
</tr>
<tr>
<td>5.11</td>
<td>Sample 'flower-like' imprecise model discovered from an event log with four sensor labels, A, B, C, and D</td>
</tr>
<tr>
<td>5.12</td>
<td>Improved model discovered from a refined event log for two sensor types, A and B</td>
</tr>
<tr>
<td>5.13</td>
<td>Transition Guard Discovered for Sensor Event Types '24,1' and '12' (Refined Log for Sensor Event Type '24')</td>
</tr>
<tr>
<td>5.14</td>
<td>Decision Tree Model Discovered from the Original Event Log and Mined Petri Net</td>
</tr>
<tr>
<td>5.15</td>
<td>Decision Tree Model Discovered from the Refined Event Log and Mined Petri Net (Sensor '24')</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>5.16</td>
<td>Petri Net Discovered from the Unrefined Event Log (Philips Event Log)</td>
</tr>
<tr>
<td>5.17</td>
<td>BIC change for the increasing number of von Mises mixture components - 'Bed' (Philips dataset)</td>
</tr>
<tr>
<td>5.18</td>
<td>Clustering the 'Bed' sensor events by fitting three von Mises distributions (Philips dataset)</td>
</tr>
<tr>
<td>5.19</td>
<td>Petri Net Discovered from the Refined Event Log (Sensor '40')</td>
</tr>
<tr>
<td>5.20</td>
<td>Petri Net Discovered from the Refined Event Log (Sensor '22')</td>
</tr>
<tr>
<td>A.1</td>
<td>Transformation of raw smart-home event log data into XES format using RapidMiner</td>
</tr>
<tr>
<td>B.1</td>
<td>Statistical Testing Results among Refined Labels of Sensor '22'</td>
</tr>
</tbody>
</table>
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Contingency table of log statistics</td>
<td>14</td>
</tr>
<tr>
<td>3.1</td>
<td>Estimated parameters for the 'TV' duration using GMM clustering.</td>
<td>26</td>
</tr>
<tr>
<td>3.2</td>
<td>Estimated parameters for the 'Fridge' and 'Plates Cupboard' event types using GMM clustering.</td>
<td>28</td>
</tr>
<tr>
<td>4.1</td>
<td>Model selection criteria based on the change in BIC value</td>
<td>37</td>
</tr>
<tr>
<td>4.2</td>
<td>Estimated parameters for a mixture of von Mises distributions, 'Plates Cupboard', (van Kasteren dataset)</td>
<td>41</td>
</tr>
<tr>
<td>4.3</td>
<td>Estimated parameters for a mixture of von Mises distributions, 'Hall-Toilet door', (van Kasteren dataset)</td>
<td>42</td>
</tr>
<tr>
<td>4.4</td>
<td>Estimated parameters for a mixture of von Mises distributions, 'Frontdoor', (van Kasteren dataset)</td>
<td>47</td>
</tr>
<tr>
<td>4.5</td>
<td>Estimated ( U^2 ) statistics and p-value for the Cluster 1 and 2 of the 'Frontdoor' sensor event</td>
<td>49</td>
</tr>
<tr>
<td>5.1</td>
<td>Sample sensor events, van Kasteren dataset</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>Sample sensor events, Philips dataset</td>
<td>51</td>
</tr>
<tr>
<td>5.3</td>
<td>Sensor types and their value ranges, Philips Dataset</td>
<td>52</td>
</tr>
<tr>
<td>5.4</td>
<td>Estimated parameters for a mixture of von Mises distributions, sensor type 'Hall-Bedroom door', (van Kasteren dataset)</td>
<td>57</td>
</tr>
<tr>
<td>5.5</td>
<td>Change in ordering relation statistics of label '24' and its refined candidates '24_1' and '24_2' with respect to sensor event type '12'</td>
<td>58</td>
</tr>
<tr>
<td>5.6</td>
<td>The relative information gain for all refined sensor event types</td>
<td>58</td>
</tr>
<tr>
<td>5.7</td>
<td>Estimated parameters for a mixture of three von Mises distributions, sensor type 'Bed', (Philips dataset)</td>
<td>60</td>
</tr>
<tr>
<td>5.8</td>
<td>Split boundaries for sensor type '40' (Philips dataset)</td>
<td>68</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

This master thesis is a graduation project for the Computer Science and Engineering master at Eindhoven University of Technology (TU/e), and has been carried out within Philips Research and Architecture of Information Systems (AIS) group of the Mathematics and Computer Science department of TU/e.

Section 1.1 will introduce the thesis context by defining the motivation behind the notion of label refinement within the process mining domain. The problem statement will be formulated in Section 1.2 by emphasizing the difference of event log types produced by both smart-home environments and traditional workflow management systems. The research goal of this project will be then presented in Section 1.3, where we will discuss the importance of the timestamp attribute of a sensor events and its role in our method development. Section 1.4 will present the overall research method and the outline of the following chapters.

1.1 Thesis Context

In recent years, a tremendous increase in the number of sensor devices, and the progress in sensor data analytics have led to new opportunities as well as challenges [11]. One such challenge is to acquire and analyze life-log data of human activities. Rapid development of ubiquitous computing environments contributed to the recognition of the life-log concept, which addresses the tracking of behavioral data through sensors [61]. Many technologies exist these days that allow to generate and store life-logs produced by humans (e.g. wearable devices), and their primary goal is to generate continuous time-series type of sensor data (e.g. blood pressure, heart rate, etc.). Such real life data logs are then analyzed by researchers in order to produce informative and digitized representations of human life activities [34].

Analysis of life-log data within the smart-house context could be considered as one of the most principal research directions in the field of life-logging. Smart house projects, such as CASAS [12], PlaceLab [28], CARE [35] and the Aware Home [32] represent a new generation of intelligent environments, which help to monitor and record human behavior in order to facilitate better quality of living. Rapid development of smart house applications forces prompt adaptation of existing technologies in order to meet sophisticated user requirements (e.g. safety and security, installation cost, low usage complexity and etc.). From this perspective, welfare and independent life maintenance of an elderly people could be considered as one of the priority directions.

The ageing of society [73, 5] creates a new interest in methods and techniques that enable elderly longer independent living. An important barrier for independent living is the decrease in

\[\text{http://www.philips.com/a-w/research/locations/eindhoven.html}\]
medicine adherence and compliance with healthy eating and drinking habits. Memory disorders like Alzheimer's disease often cause elderly to forget eating, drinking, or taking their medication. A method to detect medicine non-adherence and non-compliance with healthy eating and drinking habits would enable extension of the period of independent living by signaling deviations to home care workers, family, or the elderly themselves. Smart home sensors enable monitoring and logging of in-house activity for the purpose of triggering on-time recommendations and identifying possible risk patterns.

Researchers from a range of computer science branches, such as activity recognition (AR) and pattern discovery have been extensively involved in studies with an objective of identifying actions of one or more smart-home agents by considering environmental conditions and empirical observations. Authors of [52, 47] utilize temporal information of sensor events in order to model an adaptive smart home systems where the application of machine learning algorithms allows to discover patterns in smart-home agents' daily activities. On the contrary, researchers in [36, 27] examine the usage of a fixed and a non-fixed length sliding window approach based on sensor events in order to extract appropriate features for the behavior analysis. Despite the fact that traditional activity recognition methods considerably contributed to the understanding and analysis of human behavior within the smart house environments, it is important to state that these solutions lack of representational power (e.g. visual assessment) when it comes to the descriptions of a real life end-to-end process models.

In the field of process mining, much work has been done on checking conformance of events that happened in the real world with its regular behavior or a normative set of rules. Originally these conformance checking techniques were developed in the context of business processes, where process are often reasonably structured. Structured process models are required to apply conformance checking techniques successfully. In the smart home sensor scenario, the output of the sensors can be translated into events in many ways and on many different abstraction levels. Activity can for example be regarded on a very high level with events like eating, sleeping, and showering, or on a very low level with events like open fridge, press bed, and open bathroom door. The 'structuredness' of the process models that can be discovered from the sensor outputs is highly dependent on how the outputs are translated into events. For example, when a sensor on the fridge senses the fridge being opened it can be an indication of different events, depending on context. In case activity has been observed that concerns for example an oven or a microwave, it is likely that food is being prepared, otherwise, it is more likely that a snack has been taken out of the fridge. Other contextual information, like the time of the events, might also play a role. Researchers of TU/e and Philips Research have recently developed a method to estimate the usefulness of a context-based event abstraction or refinement step without the need of computationally expensive process discovery. The goal of this final master project is to automatically generate context-based refinements or abstractions of a smart home event log, after which the recently developed method can be used to select the most useful refinement or abstraction from the set of ideas.

1.2 Problem Statement

Mining of meaningful and reliable process models could be considered relatively convenient when applied on event logs obtained from classical business management systems (e.g. work-flow management and transactions systems). Major reason behind this is a traditional event labeling (i.e. the labels of events in such systems are known/clear) used by organizations to differentiate business process activities. On the contrary, life-log events present a challenging application domain for the process discovery algorithms in the absence of a complementary contextual information (e.g. time difference between the events, location).

The problem is that labels of events are generally not known, and, depending on what additional information we take into account, every event is either unique (when we take into account all
contextual information, including sensor value), or, there is no structure in the models that can be discovered (when we take into account only sensor id). In particular, identification of a specific event types and assignment of appropriate labels (i.e. events that belong to the same type are assigned same label) to them are still considered to be a relatively complex problem. As an example, let us now examine Figures 1.1 and 1.3, which show two different types of event logs.

As can be seen, both event logs perfectly record the occurrence of events within the respective application domains. It would be appropriate to mention that in case of business process management system, every ‘Case ID’ represents one particular instance of a process, such as ’register request’ or ’make payment’. Each case is represented by a sequence of execution activities required to accomplish this case (sometimes called trace). On the contrary, a transformation of smart-home life-log into cases is performed by grouping together same day-part of the time-stamp, yet other grouping options are possible (e.g. address, room location). In both event logs ordering of events is based on time-stamp attribute and every single event has a unique ’Event ID’.

By comparing two event logs, one can easily derive that business process management systems operate with ‘finely-grained’ real-life activities. Application of the existing process discovery algorithms on this event log allows to discover a reliable and interpretable process model, like the one depicted in Figure 1.2.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Event ID</th>
<th>Timestamp</th>
<th>Activity</th>
<th>Resource</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>30-12-2010 11:02</td>
<td>Register Request</td>
<td>Peter</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>31-12-2010 10:06</td>
<td>Examine Thoroughly</td>
<td>Sue</td>
<td>...</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>05-01-2011 15:12</td>
<td>Check Ticket</td>
<td>Mike</td>
<td>...</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>06-01-2011 11:18</td>
<td>Decide</td>
<td>Sara</td>
<td>...</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>07-01-2011 14:24</td>
<td>Reject Request</td>
<td>Pete</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>30-12-2010 11:32</td>
<td>Register Request</td>
<td>Mike</td>
<td>...</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>30-12-2010 12:12</td>
<td>Check Ticket</td>
<td>Mike</td>
<td>...</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>30-12-2010 14:16</td>
<td>Examine Casually</td>
<td>Peter</td>
<td>...</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>05-01-2011 11:22</td>
<td>Decide</td>
<td>Sara</td>
<td>...</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>08-01-2011 12:05</td>
<td>Pay Compensation</td>
<td>Ellen</td>
<td>...</td>
</tr>
<tr>
<td>11</td>
<td>13</td>
<td>15-01-2011 13:34</td>
<td>Register Request</td>
<td>Peter</td>
<td>...</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>10-01-2011 10:26</td>
<td>Examine Thoroughly</td>
<td>Sara</td>
<td>...</td>
</tr>
<tr>
<td>13</td>
<td>15</td>
<td>06-01-2011 13:06</td>
<td>Reinitiate Request</td>
<td>Sara</td>
<td>...</td>
</tr>
<tr>
<td>14</td>
<td>16</td>
<td>06-01-2011 14:34</td>
<td>Examine Casually</td>
<td>Peter</td>
<td>...</td>
</tr>
<tr>
<td>15</td>
<td>17</td>
<td>09-01-2011 09:55</td>
<td>Decide</td>
<td>Sara</td>
<td>...</td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td>15-01-2011 10:45</td>
<td>Pay Compensation</td>
<td>Ellen</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 1.1: Business Process Management System Event Log

Log adopted from Wil M.P. van der Aalst Process Mining: Discovery, Conformance and Enhancement of Business Processes
In comparison, the event log produced by smart-home agent allows to discover the model represented by Figure 1.4. One of the major drawbacks of this process model is that it lacks of representational power (i.e. we know what kind of sensor has been triggered, but we do not know what kind of activity is possibly performed) that could be useful for the human activity analysis and corresponding smart-home procedures (e.g. recommend to take medicines on time). For instance, 'Kitchen' sensor event could be classified into a number of possible activity types, such as ‘cooking’ or ‘eating’.

In addition, process models that can be discovered based on a sensor event level described in Figure 1.4 are frequently unstructured (‘flower-like models’). This type of models (usually described in a form of Petri Net) allows to replay all traces that could be found within an original event log, but also traces dissimilar to the observed behavior [67]. In other words, ‘flower-like’ models are under-fitting the underlying event log, because they overgeneralize the observed behavior [66]. As a result, we obtain imprecise process models that do not allow to make any meaningful conclusions about the human behavior within a smart-home environment.

<table>
<thead>
<tr>
<th>Case ID</th>
<th>Event ID</th>
<th>Timestamp</th>
<th>Sensor Event</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>15-02-2014 09:02</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>15-02-2014 09:03</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>15-02-2014 09:05</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>15-02-2014 09:10</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>15-02-2014 09:14</td>
<td>Livingroom Motion</td>
<td>Livingroom</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>15-02-2014 09:20</td>
<td>Bathroom Motion</td>
<td>Bathroom</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>16-02-2014 08:53</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>16-02-2014 08:58</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>16-02-2014 09:02</td>
<td>Kitchen Motion</td>
<td>Kitchen</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>16-02-2014 09:07</td>
<td>Livingroom Motion</td>
<td>Livingroom</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>16-02-2014 09:10</td>
<td>Bathroom Motion</td>
<td>Bathroom</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>16-02-2014 09:17</td>
<td>Frontdoor</td>
<td>Hall</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Figure 1.3: Smart Home Event Log

---

3Model adopted from Wil M.P. van der Aalst Process Mining: Discovery, Conformance and Enhancement of Business Processes
CHAPTER 1. INTRODUCTION

Using additional contextual information (e.g. preceding and succeeding events, time-stamp, duration) it is actually possible to refine event label 'Kitchen' into either 'Cooking' or 'Eating'. This information might be useful in case an elderly inhabitant of a smart-home has been prescribed the 'before-meal' or 'after-meal' daily medicines and notification system is required to assure performing that action. From this perspective, a desired process model might look like the one described in Figure 1.5.

As can be seen from Figure 1.5, label refinement of the original smart-home event log allows for discovery of a more detailed human behavior model, which helps to identify the activities of interest. In this Petri Net, we can observe that the refinement of 'Kitchen' sensor label into 'Cooking' and 'Eating' sensor labels allows to notice that 'Frontdoor' event might happen only after the person is cooking and not eating. On the other hand, the eating activity is usually followed by visits to toilet or bathroom. It was not possible to derive these kind of conclusions, using the Petri Net from Figure 1.4.

A number of event relabeling options are applicable for this case, such as machine learning, data mining or even manual annotation by domain experts. For the latter case, home care employee might indicate that an elderly inhabitant cooks daily between 18:00 and 19:00 for a duration of 30 minutes. Using this information, label refinement could be achieved by relabeling all 'Kitchen' events that take place within that time interval (and with a similar duration) as a 'Cooking' activity. Otherwise a system might assume that the happening event is related to the 'Eating' activity. However, by considering a number of smart-home sensor generated events, manual refinement of event logs is not a feasible option, which requires the investigation of autonomous approaches.
Unfortunately, most of the existing studies fail to address the difficulty of identifying low-level event labels in the context of non-deterministic human behavior, which is observed within smart house environments.

Traditional activity recognition and pattern discovery methods incorporate a number of possible solutions, such as supervised learning, model-based learning and online learning. Specifically, in order to develop appropriate activity models, several probability-based algorithms, such as the Hidden Markov Model (HMM) and the Conditional Random Field (CRF) have been used extensively [33, 70]. Application of these models usually requires the discretization of sensor data into fixed-length time slices. These time slices are exposed to further transformations in order to obtain better feature collections for classification problems. Another popular approach, namely Bayesian Network (BN), have been examined in [69, 46]. Bayesian Network (as well as HMM and CRF) is a probabilistic graphical model described in a form of directed acyclic graph where nodes represent random variables and edges represent statistical dependencies between those variables. One of the major advantages of BNs for activity recognition is their ability to deliver the notion of causality by indicating the conditional probability distribution per node. The fundamental difference between such machine learning models (as HMM and CRF, also RNN) and process models is that the latter ones can be easily visualized/interpreted and can therefore be used to communicate between process stakeholders.

One common characteristic of these approaches is that they represent supervised learning techniques, which are based on explicit identification of target labels (i.e. human activity corresponding to the sensor observation). Learning from training data requires human intervention; domain experts, model developers or even smart house agents need to label both low-level sensor events and/or the activities associated with them [21]. By considering the number of possible sensor events and activity types within the modern smart house environments, manual annotation process may be cumbersome during the data collection phase. In addition, labeled training data sets might vary significantly for different smart house environments, which makes the creation of a common activity learning model (i.e. activity recognition model for different smart-home agents) troublesome. It is now time to define our problem statement:

**Problem Statement:** Given a smart-home sensor event log, how to apply unsupervised machine learning algorithms in order to generate a reliable label refinement method.

### 1.3 Research Question

It has been indicated that addressing label refinement problem from the process mining point of view is crucial for the purpose of discovering self-contained and reliable process models. The quality of these process models directly affects the analysis of human behavior by pointing out undesirable deviations and risk patterns within the context of elderly welfare. In order to discover an informative process model, proper label refinement method is required that will help a discovery algorithm to mine better process models. By considering the sensor event log produced by smart-home agents behavior (Figure 1.3), our high-level goal is illustrated by Figure 1.6, which represents the application of a label refinement method on 'Kitchen' sensor event type.

We have already discussed that (based on the expertise of a home-care employee) it is possible to refine motion within the kitchen into a variety of possible activities. In this particular case, we relabel every occurrence of 'Kitchen' sensor event around 18:00-19:00 into 'Cooking' event, which is On the contrary to 'Eating' event that happens in other time intervals. Although, this example demonstrates a simplified case that might be obtained from a real life scenario, it helps to point out the contextual information that we will utilize in our project, namely timestamp attribute.
CHAPTER 1. INTRODUCTION

For the purpose of generating a label refinement method, we would like to investigate two categories of a timestamp attribute: starting time and duration of every sensor type. The whole procedure is based on the hypothesis that:

**Hypothesis:** For every single sensor event type it is highly possible to cluster its aggregated occurrence instances around the day (i.e. based on its starting time derived from the timestamp attribute and duration) in order to discover interesting patterns of human behavior.

Our work might be compared to [47], where authors have developed an AR model based on the notion of the temporal association rules, which is an extension of the traditional association rules with a time dimension. According to authors, mining of temporal relations with the help of a timestamp attribute could be obtained in two major steps. First, they derive canonical representations of start times and durations for every activity instance within the existing dataset. Second, using the information from the previous step, for each activity type they determine the set of subsequent activities with a certain level of confidence. According to authors, obtained information might play an important role in various activity prediction scenarios and deviation detection procedures.

On the contrary, in our project we focus on the event level instead of activity level, which is crucial for the discovery of fine-grained process models and does not require pre-annotation of activities. Moreover, we do not extend our application with association rules, but rather check what type of events precede/follow our target events for every identified cluster. In addition, we will show that clustering method utilized by the authors (i.e. clustering using Gaussian Mixture Model) may not be suitable for clustering the data around the midnight due to the circular nature of the timestamp attribute used. As a result, we will present the notion of a circular data type and a complete set of statistical steps required for the successful implementation of circular data clustering. This procedure will be performed for every single sensor type and the obtained number of clusters will indicate a possible number of refinements that will be applied on each sensor within the original event log. We can now introduce the research question that will be explored in this thesis project:

**Log-based Generation of Event Labels using Event Attributes**
CHAPTER 1. INTRODUCTION

Research Question: Given an unprocessed (i.e. original) smart-home sensor event log, we would like to develop an event label refinement method based on the clustering of starting and/or duration times of sensor type instances around the day, which will help to obtain a more 'fine-grained' sensor event log for the application of process discovery algorithms.

Refined event labels are expected to serve as a better groundwork for the discovery of a more detailed and structured process models. However, an application of process discovery algorithms on a set of refined event logs could be considered as a costly procedure, which also requires model assessment along the traditional quality dimensions (i.e. precision, generalization, fitness, simplicity). One major problem with this approach is that indicated quality metrics could be compared only if process discovery algorithms are applied to the very same event log (i.e. same event labels). However, label refinement method is expected to generate event labels that will constitute distinct event logs in comparison to the original one. Authors of [17] have recently developed a solution to estimate the usefulness of a context-based event refinement step without the need of computationally expensive process discovery.

Suggested evaluation method is based on two major steps. First, it checks if label refinement method facilitates the log statistics to become more deterministic expressed in a form of entropy. Log statistics encapsulates a number of event ordering patterns that are crucial to majority of process discovery algorithms (e.g. direct successor, length-two loop, direct/indirect successor) [65, 68]. Second, statistical test estimates if label refinement generates considerable statistical difference to at least one of the log statistics. One of the major goals of this project is to present a concrete approach to automatically generate context-based (in our case starting time and duration of events) refinements or abstractions of a smart home event log, after which this statistical method can be used to select the most useful refinement [59].

1.4 Research Method and Outline

Following steps have been taken in order to solve research question and address problem statement:

- Examination of the existing literature on unsupervised learning methods and their application for our domain has been performed first. Detailed explanation could be found in Chapter 2. This chapter also contains the discussion about the existing process model quality metrics and which of them are targeted in our work.

- Chapter 3 presents our selection of a clustering approach, namely Gaussian Mixture Modeling and its application on sensor event duration. In this chapter we derive a canonical representation of every sensor duration (i.e. how long on average does it take for an event to happen) and show how well Gaussian Mixture Modeling clusters these 1-dimensional feature domain. We will introduce a limitation of applying GMM method in case we switch our feature domain from being linear (duration of sensor event) to circular one (starting time of sensor event).

- Chapter 4 describes the application of von Mises Mixture Modeling on starting time of sensor, while presenting a set of statistical steps required for the successful implementation of sensor event clustering. This chapter will introduce our final approach for the generation of a label refinement method.

- The validity of our approach will be evaluated in Chapter 5, where we will examine how our label refinement method allows to obtain more structured process models for two different smart-home environments.

- Conclusion of this thesis will be presented in Chapter 6 that will also contain limitations of our method and future work discussions.
Chapter 2

Preliminaries

As we have defined our problem statement and research question in the previous section, in this chapter we will introduce preliminary concepts that will be utilized throughout this thesis project.

In Section 2.1, we will briefly introduce certain objectives of the process mining research discipline, mostly focusing on the structure of event logs and traditional quality metrics, which are used for the assessment of process discovery. This section will also introduce a new evaluation method for the label refinement domain, which is computationally less expensive than the evaluation based on the application of process discovery algorithms. Literature study for the domain of label refinement will be presented in Section 2.2 covering several approaches proposed by fellow researchers. We will continue with Section 2.3 that will explain our motivation behind the unsupervised machine learning approach that we decided to utilize for the generation of label refinement method. Section 2.4 will describe our approach towards the selection of an appropriate clustering method, which is required for a smart-home sensor event attribute (i.e. timestamp) that have been used in our project.

2.1 Process Mining

Process mining is a comparatively young research field, which focuses on the discovery and analysis of process models using event data. Traditional data mining techniques, such as regression, classification or association rule learning do not provide complete analysis methodologies that will help to discover, present and perform a diagnosis of business process models. Process mining fills this gap by focusing on end-to-end real life processes, while incorporating various process discovery algorithms and conformance checking methods [64]. Another key point distinguishing process mining field from traditional data mining methodologies is that the former one deals with models that can be visually represented and have formal semantics.

As can be seen from Figure 2.1, event logs are aggregated by a variety of software systems that operate within the real world environments. Obtaining insights into processes based on event logs is then achieved with the process mining. There are three major techniques commonly used within this research field:

- **Process discovery** is a technique that allows for automatically discovery of process models using the recorded events within an event log. In recent years, a number of discovery algorithms has been proposed and the goal is to extract and present descriptive models without under-fitting (i.e. too much possible behavior allowed) or over-fitting (i.e. too less possible behavior allowed) event logs.
CHAPTER 2. PRELIMINARIES

- Model under (over)-fitting could be identified and evaluated by help of **Conformance checking** technique. Unlike process discovery, this technique utilizes both discovered (or manually drawn) process models and their event logs in order to quantify the commonalities and discrepancies between the modeled and mined behaviors. This technique helps to detect deviations from real life (process) scenarios that have been established by process owners within organizations.

- In addition, **Process enhancement** technique helps to extend/improve discovered process models by utilizing the information stored in event logs. For instance, usage of timestamp attribute might help to detect bottlenecks throughout the process instance execution phase (e.g. loan registration from bank). Process enhancement should not be confused with conformance checking. The latter technique assesses the alignment between models and reality, whereas the former one helps to improve/extend original process models [63].

![Figure 2.1: Process Mining application domain](image)

In our project, we will demonstrate how the performance of the first two techniques (i.e. process discovery and conformance checking) might be affected by the refined event log. In the following two subsections, we will talk about the structure of event logs and their role in the process discovery and conformance checking. We will also consider two quality metrics of process models, namely generalization and precision, which are directly affected by the quality of provided event logs and will demonstrate a new approach that has been developed by researchers at the TU/e and Philips Research in order to assess the quality of the label refinement method.

### 2.1.1 Event Log

In Chapter 1, we have indicated that smart-home environments equipped with a number of sensor do not generate event logs from which a meaningful process models can be discovered. In order to explain the problem behind this, let us now examine the graphical representation (Figure 2.2) of event logs structure as has been illustrated in the Process Mining book by W. M. P. van der Aalst [63]. In traditional business workflow managements systems, process owners indicate what kind of activities should be performed in order to achieve a particular business goal (e.g. provide a loan, register a house at municipality and etc.). For any business process, a case represents a process instance, which is defined by a sequential record of events, and every event represents a corresponding business activity. Events are order based on a timestamp attribute that allows to

---

1 see Wil M.P. van der Aalst "Process Mining: Discovery, Conformance and Enhancement of Business Processes"
CHAPTER 2. PRELIMINARIES

track down the occurrence sequence of activities. A variety of attributes could be associated with a particular event, such as resource (i.e. who has performed the activity), cost and etc. Successful completion of this sequential chain of events (i.e. activities) allows to achieve a required business goal. In addition, each event has a label, which explicitly indicates what kind of activity has been performed.

Figure 2.2: Structure of event logs within Process Mining domain

Event logs serve as an input for process discovery algorithms, which associate every unique event label with one and only one activity type. Activity represents an execution unit, and relates to a particular step required for a successful completion of a given real-life process. Traditional conventions of the process mining field threat ‘event labels’ and ‘activity’ as a same thing, by considering the fact an event label usually reflects the name of a performed activity. However, within the smart-home context, event labels represent only abstract sensor triggers, which can not be immediately associated with some activity type.

Thus, it is impossible to discover a process model that will represent a real life behavior if a supplied event log will contain different activities with a same label. This is what happens if we would consider original smart-home event logs, where instead of predefined activity labels we usually receive high-level abstract sensor labels. In this type of event logs, event labels indicate that a particular sensor has been triggered, and it is relatively hard to obtain any further insight about the type of activity that has been performed. Application of process discovery algorithms on this kind of event logs often results into under-fitting process models, which allow much more behavior that the original smart-home event log. A simple approach will be to assign a unique label to every single event, however then the discovered process model will over-fit the event log. Hence, it is important to develop a sophisticated method that will help to assign a same label to sensor events, which are similar based on some contextual information. In particular, we are required to develop a label refinement method that assigns a new label to events in the event log such that two events that were labeled the same before can now be labeled differently, while

\[\text{Log-based Generation of Event Labels using Event Attributes}\]
everything that was labeled differently before is also labeled differently after the refinement. We will show how timestamp of sensor events allows to group similar events, which are expected to represent different activity types. Our approach should be considered as a pre-processing step of an original sensor event log, and refined event labels will allow to discover more insightful process models of a human behavior.

2.1.2 Quality Metrics: Precision versus Generalization

Within the process mining domain, the quality of discovered models is usually assessed by balancing between four quality dimensions: precision, generalization, fitness and simplicity [63]. The simplicity dimensions could be compared to the Occam’s Razor principle, which states that the best model is the one that explains the underlying behavior within the log in the simplest way. On the other hand, fitness dimension helps to ensure that every single case (i.e. trace) that has been observed within the log can be replied on the (discovered) model. We should state that both simplicity and fitness does not depend on the quality of the label refinement method and directly related to the performance of process discovery algorithms.

On the contrary, precision and generalization dimension can be directly affected by the improved (i.e. refined) event log. Figure 2.3 illustrates types of model that we would like to avoid as our final process model output. Given a particular event log \( A \), neither model \( M_1 \), nor model \( M_2 \) represent a precise and sufficiently general process model. Precise models do not allow for behavior that could not be observed from a submitted event log. Model \( M_2 \) is a typical example of ‘flower-like’ model where we can fire any transition (i.e activity) in any order and generate a set of traces that will under-fit the original event log. On the other hand, model \( M_1 \) over-fits the event log by constructing a path for every single trace type that has been observed. This kind of model is capable of explaining the behavior of the current event log \( A \), however will fail to explain another sample event log that will represent the very same process.

![Figure 2.3: Two types of process models that could be discovered from the same event log](image)
For a majority of process discovery algorithms the ordering of activities observed throughout the event log is crucial. Within the process mining domain the ordering of activities helps to mine a control-flow perspective of a process model. This perspective provides an optimal characterization of all possible traces within the log, usually expressed in a form of Petri Net [63] (e.g. models $M_1$ and $M_2$). Execution of these activities might follow different patterns, such as sequential, concurrent or optional. In order to understand why ordering of an abstract smart-home sensor labels does not allow to discover a reliable process models, let us examine a small event log $L$, which contains only 4 traces: $L = \{(EBACD), (BEDCA), (DEBCA), (ABCDE)\}$. Every letter represents a distinct sensor event and ordering of events is based on their timestamp attribute. Let us now assume that there are two particular sensor events that we are interested in; $C$ - 'Frontdoor' sensor and $A$ - 'Bed' sensor. The triggering of 'Bed' sensor might be due to several reasons; either someone is tossing (i.e. still sleeping) in a bed or maybe waking-up. It would be logically to assume that 'Frontdoor' sensor event might happen only when person is awake and not still in a bed. 'Sleeping' and 'Waking-up' is exact type of event abstraction level that we are interested in, thus it would be useful to split the original sensor event $A$('Bed') into two different events, $A_1$ ('Waking-up') and $A_2$ ('Sleeping'). However, within the original event log $L$, both possible activity types are represented by an abstract sensor label $A$, which does not allow process discovery algorithms to mine a real order of events. Refinement of this log $L$ into a new log $L_1 = \{(EBA_1CD), (BEDCA_2), (DEBCA_2), (A_1BCDE)\}$ might help to resolve the existing problem, where it is clear that sensor event $C$ ('Frontdoor') always follows $A_1$ ('Waking-up') and always precedes $A_2$ ('Sleeping'). Appropriate label refinement method allows to restrict unseen behavior within the original smart-home log, which positively affects the precision and generalization metrics of the discovered human behavior process models. Contextual information in a form of event timestamp (in particular, starting time and duration) might help to generate a necessary label refinement method. For instance, we might discover that the events $A$ ('Bed') happening between 23:00 - 07:00 and between 07:30 - 10:00 are different from the perspective of following and preceding sensor events. In Chapters 3 and 4 we will discuss how it is possible to achieve an appropriate label refinement method by utilizing timestamp attribute of our sensor events.

### 2.1.3 Evaluation Method for Label Refinements

Up until now we have discussed only traditional quality metrics used to assess the performance of process discovery algorithms. For the label refinement task there are two major problems that require an alternative evaluation metric to be developed:

- First, we can apply process discovery algorithms on a refined event log and compare the structure of a discovered process model to the original one. However, by considering a potentially high number of label refinement options, this approach might be impractical, because process discovery is a computationally expensive task.

- Second, traditional quality metrics, such as precision and generalization defined in such a way that it is impossible to compare the performance of two relabeling methods against each other based on the numbers of these quality metrics. Major reason behind this is that these metrics require the same labeling language (i.e. label domain for all sensor events within the log) for process discovery algorithms, in order to be able to conduct a reliable comparison. In contrast, various relabeling methods produce logs with different event labels, which creates an obstacle for the evaluation of a usefulness of a suggested method.

In order to tackle these problems, researchers at the TU/e and Philips Research developed an evaluation method of label refinement, which incorporates entropy-based measure and a statistical testing. We have indicated the importance of event ordering within the log and their explanation could be found in [65, 68].

However, in order to be able to reason about the log-based ordering statistics of events, let us
formally define it as described in work by Tax et al. [59]. Let \( L \) be an event log, and let \( b \) and \( c \) be two events from that \( \log b, c \in A(L) \), where \( A(L) \subseteq A_1 \ldots A_n \) denotes the alphabet of event labels that exist in \( \log L \), and ordering relationship between \( b \) and \( c \) defined as:

- \( \#L^+ > (b, c) \) is the number of occurrences of \( b \) in the traces of \( L \) that are directly followed by \( c \), and \( \#L^-, > (b, c) \) is the number of occurrences of \( b \) which are not directly followed by \( c \) (directly successor pattern).

- \( \#L^+, >> (b, c) \) is the number of occurrences of \( b \) that are followed by \( c \), and \( \#L^-, >> (b, c) \) is the number of occurrences of \( b \) that are not followed by \( c \) (length-two loops).

- \( \#L^+, >>> (b, c) \) is the number of occurrences of \( b \) that are eventually followed by \( c \), and \( \#L^-, >>> (b, c) \) is the number of occurrences of \( b \) that are not eventually followed by \( c \) (direct or indirect successor).

Let us now assume that event label \( c \) has been refined into \( c_1 \) and \( c_2 \). Then ordering statistics of \( c_1 \) and \( c_2 \) within the event log in respect to event \( b \) can be expressed in a form of contingency table (Table 2.1).

<table>
<thead>
<tr>
<th></th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( + )</td>
<td>( #L^+_{1,x}(c_1, b) )</td>
<td>( #L^+_{2,x}(c_2, b) )</td>
<td>( #L^+_i(c_1, b) )</td>
</tr>
<tr>
<td>( - )</td>
<td>( #L^-_{1,x}(c_1, b) )</td>
<td>( #L^-_{2,x}(c_2, b) )</td>
<td>( #L^-_i(c_1, b) )</td>
</tr>
</tbody>
</table>

In the table above \( L_1 \) is an original event log and \( L_2 \) is a refined event log that contains refined labels \( c_1 \) and \( c_2 \). In addition, \( \#L^+_i(i, j) \) indicates the number of times when \( i \) satisfy relation \( x \) (directly follows, directly precedes, eventually follows, eventually precedes) with respect to \( j \), and \( \#L^-_i(i, j) \) tracks the number of times when \( i \) does not satisfy relation \( x \) with respect to \( j \).

To complete our discussion of the evaluation metrics proposed by the authors, we will briefly discuss two major components of their method: entropy-based information gain and statistical testing.

- **Information gain.** Authors have utilized the binary entropy function, expressed as \( H_b(p) = -p \log p - (1-p) \log (1-p) \), where \( \log 0 \) is equal \( 0 \). Application of the binary entropy function on log statistics allows them to calculate the level of non-determinism, which is important for process discovery algorithms. Log statistics with a low indicator of entropy level helps process algorithms to discover more structured process models. The information gain for any type of log statistic is calculated as total entropy of log statistic before the refinement minus conditional entropy after refinement. The conditional entropy is the weighted average of the entropy of refined labels. Finally relative information gain is calculated that describes how label refinement method helped (or not) to reduce bits of entropy in comparison to the original event log.

- **Statistical Testing.** Due to the fact that infrequent refined labels might increase the relative information gain by chance, a statistical testing is also required that will allow to calculate the actual information gain. In order to address this problem, authors have used Fisher’s exact test [18], which is a statistical significance test proposed to analyze contingency tables. \( P \)-value of this significance test is calculated based on the null hypothesis that refined events \( i_1, ..., i_n \) are equally satisfy log-ordering relation \( x \) with regard to some event \( j \).

According to the authors, the computational complexity of their evaluation method is \( \mathcal{O}(|S| \times |L| \times |splitSet|) \) (splitSet function is required to split the original label into a number of candidates), whereas many process discover algorithms increase their computational complexity exponentially by increasing number of label refinements. Thus, evaluation of our label refinement method will be performed based on this method and will be examined in Chapter 5.
2.2 Literature Study - Label Refinement

The problem of label refinement within the smart-home context could be considered a relatively new research direction for the process mining discipline. Before explaining our own approach, it would be appropriate to discuss related work that has been conducted by fellow researchers.

Despite the fact that several process discovery methods address the refinement problem to the certain extent [25, 9], their methods could not be easily applied on other process discovery algorithms. In other words, instead of resolving the label refinement problem at the process discovery algorithm level (i.e. each algorithm should resolve this problem differently) it would be more convenient to address this issue at the event log level. Control-flow patterns within an event log are what most process discovery algorithms agree upon, and this requires to come-up with a standard event log format for a majority of those algorithms.

Several methods exist that help to distinguish between different variants of the process by applying clustering techniques on event traces [20, 75]. According to authors, assignment of similar traces into groups and division of complete event-logs into smaller sub-logs allows to overcome the discovery of inaccurate and highly complex process models. One of the major drawbacks of these approaches is that they operate on the trace level and not capable of distinguishing between two different activities with a same label. On the contrary, our clustering approach is applied at the event level, which will help to detect and refine events accordingly.

In their work, Lu et al. [38] propose a general framework that helps to pre-process the original event log in order to handle duplicated activities. Introduced method consists of three major steps. First, in order to restrict the search scope, authors identify possible candidates for imprecise event labels using domain expert knowledge and automated method based on the properties of Inductive Miner [37]. Second and third steps in their approach allow to refine the imprecise labels horizontally (i.e. refine labels into different variants across the traces) and vertically (i.e. refine labels into various activities within a trace), respectively. In addition, intermediate step is taken by authors (after step one) in order to identify a particular similarities between activities across traces. This step is performed based on the trace matching technique, where optimal mapping of similar events is achieved based on the cost function (i.e. higher cost between two events results into higher chance to label them differently). Application of this method within the smart-home context could be considered problematic, because it does not take contextual information of an abstract sensor events into account (but rather tries to identify duplicated activities within an event log). Our machine learning approach deploys the timestamp attribute of sensor events, which is important from the perspective of human behavior analysis.

In addition, authors of [41, 14] propose data-aware process discovery and evaluation techniques that allow to add data-perspective analysis to the process model discovery in addition to the commonly used control-flow perspective. In order to do so, authors, first discover the control-flow of process models by utilizing one of the available process discovery algorithms. Next, they perform the alignment between the discovered control-flow and event log for the purpose of conformance checking without considering any event data attributes. Finally, they discover data-flow perspective using the notion of transition guards. These guards represent decision points, and according to authors can be seen as a classification problem (i.e. a particular trace is taken within a process model based on the values of variables). Figures 2.4 illustrates two models ($M_1$ and $M_2$) which describe the very same process of loan request handling, however using different perspectives. $M_1$ is a representation of control-flow perspective, which does not take data attributes of events into account. Adding data-flow perspective ($M_2$) allows to increase the model precision by restricting the behaviour that could not be observed from the event log. For instance, by mining the decision points, authors identified that the activity 'Simple Check' can only be performed when loan amount is less than 2000 ($M_2$). $M_1$ fails to address this issue, thus allowing for an additional behavior, which does not represent the underlying event log. Within the smart-home context, this method might be useful, if for instance, we would like to refine the occurrence of 'Bed' sensor
event into 'Waking-up' and 'Sleeping' labels, respectively. However, one of the major limitations of this approach is that during its initial phase, control-flow of the future process model is already discovered based on the original smart-home event log, thus leading to a 'flower-like' model. In Chapter 5, we will compare these two approaches, and present the importance of resolving label refinement problem at the level of event-log.

![Figure 2.4: Representation of control-flow without \((M_1)\) and with \((M_2)\) transition guards (BPMN notation)\(^3\)](image)

**2.3 Unsupervised versus Supervised Machine Learning**

Within the machine learning domain, classification and clustering represent the largest application approaches, being related to the instances of supervised and unsupervised learning methods, respectively. The former method addresses the problem of classifying newly arriving observations by considering response variables \(y = (y_1, y_2, ..., y_m)\) given a set of predictor variables \(x = (x_1, x_2, ..., x_m)\). Classification models build upon existing training sets, which contain observations that are already categorized according to the known labels (e.g. fraudulent versus non-fraudulent, malignant versus benign).

Existing supervised learning algorithms, such as support vector machine (SVM), decision trees (DT), artificial neural networks (ANN) utilize training sets in order to prepare models for future classification tasks. Performance and reliability of these classification models could be assessed by help of validation and test sets, where label column is skipped in order to evaluate how well a particular model can predict target class. From the perspective of smart-home domain, application of supervised learning methods requires generation of ground-truth activity labels that could be fit to the algorithms, such as HMM or CRF. We would like to indicate three major obstacles for a proper training set generation procedure.

\(^3\)adopted from Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, Wil M.P. van der Aalst "Measuring the Precision of Multi-perspective Process Models"
First of all, obtaining the training activity labels could be achieved in numerous of ways; manual annotation by smart-home agents indicating what activity is about to be performed [58, 49], or following a pre-defined activity protocol, which provides an agent with a sequence of activities to be performed [13, 22]. Practical nature of these approaches could be argued, by considering the fact that for elderly people, it is usually hard to memorize performed activities, not to mention accurate indication of when those activities have been performed.

Moreover, it would appropriate to state that individual behavior of different smart-home agents might vary significantly, which could drastically decrease the prediction power of a classification model. Interesting statement has been made by the authors of [53], who made a clear distinction between inter-subject and intra-subject variability in respect to every activity execution. The former concept indicates the fact that different smart-home agents can perform a given activity in many different ways, whereas the latter concept implies that the very same smart-home agent can perform the same activity differently. As a result, a reliable model per smart-home agent is required, which will allow to differentiate specific behavioral characteristics (habits) of a particular elderly person.

Finally, in order to assure the high accuracy of a classification model, a sufficient number of training sensor events is required that will constitute a solid ground truth for a future predictions. Labeling of such a large amount of sensor events is a relatively time-consuming procedure, which requires extra efforts in order to obtain a consistent training set. Evaluation of these difficulties, lead us to the implementation of an unsupervised label identification method that helps to overcome the discussed obstacles.

Clustering is one of the heavily used unsupervised machine learning algorithms, which allows to organize a set of observations into meaningful subsets (i.e. clusters), where the major objective is to preserve the notions of cohesion and coupling. Cohesion refers to the inter-component relation among the elements (i.e. observations) of the same cluster, whereas coupling helps to evaluate the intra-component relation among the elements of distinct clusters. A reliable clustering model should be able to maintain a high level of cohesion and low level of coupling values. From the perspective of statistics, clustering of observations could be related to the well know problem of density estimation, which is based upon an underlying probability density function:

\[
Pr[a \leq X \leq b] = \int_{a}^{b} f_{X}(x) dx
\]  

Major objective of clustering algorithms is to learn the underlying structure of data given a number of observations \((x_1, x_2, \ldots, x_n)\) and infer the properties of the probability density \(Pr(X)\) without any supervision [24]. Deployment of clustering algorithms on smart-home sensor events for the purpose of activity identification could be considered as one of the traditional and reliable approaches. In Chapter 1, we have already discussed several methods that have been used extensively by researchers from the fields of activity recognition and pattern discovery. On the contrary, for the field of process mining, analysis of smart-home human behavior by help of clustering algorithms is a relatively new application domain.

As it was already indicated, behavioral characteristics of smart-home agents incorporate certain level of randomness, and do not follow the exact execution procedures (unless explicitly required). As a result, application of process mining techniques (e.g. process discovery) on raw and non-processed sensor event logs does not allow to obtain meaningful insights. Using process mining terminology, it is complex to assure that non-processed sensor event logs can help us to discover human behavior models with a sufficient levels of precision and generalization. Thus, a comprehensive clustering methodology should be developed in order to process the original smart-home event log that will help to discover meaningful and interpretable process models.
2.4 Clustering Method Selection

In Chapter 1, we have indicated that human behavior is diverse, which requires appropriate clustering methodologies to be utilized for the label refinement. There are two major clustering types used extensively by researchers and practitioners in the machine learning field, namely hard and soft clustering. Hard clustering methods, such as traditional K-means [39] require grouping of a data in an exclusive way, which assigns every single data point to only one definite cluster. On the other hand, soft clustering methods allow to cluster data, such that every single data point may belong to several clusters with a certain membership degree between 0 and 1 (and sum of membership degrees of one data point over all clusters sum to 1) [7].

In our project, we would like to explore the possibility of applying soft clustering method to our life-log sensor data in order to generate a reliable label refinement method. This approach is justified by the work of Charniak and Goldman [10], who used to show that activity identification model could not be considered complete without incorporating a certain level of uncertainty reasoning. This reasoning could be also applied to the problem of label refinement by taking into account that it is related to the concept of activity recognition (though at different abstraction levels). We have briefly discussed several probabilistic models in Chapter 1 that are commonly applied and produce good results within the field of activity recognition. On the contrary, in our project, we will use probabilistic mixture models that will constitute the backbone of our label refinement method within the process mining domain. Detailed explanation of this method and major motivation behind it are presented below.

2.4.1 Probabilistic Mixture Models

Clustering by mixture modeling is based on the assumption that every single data point within a given population can be described by a probability density function from which it might be drawn. Mixture modeling uses a finite set of components, where every single component is a parametrized density function model. Following the definition from [24], mixture model has the form:

Definition 1. (adopted and modified from [24]) K number of density function components form a mixture model $f(x)$ which can be written as:

$$f(x) = \sum_{k=1}^{K} \alpha_k f_k(x)$$  \hspace{1cm} (2.2)

where the mixture weight for a given component $k$ defined by $\alpha_k$ and $\sum_{k=1}^{K} \alpha_k = 1$.

One of the major advantages of mixture modeling is that any type of distribution could be used based on the underlying nature of your population. By considering the fact that every distribution has its unique parameters, Equation 2.2 could be refined into:

$$f(x) = \sum_{k=1}^{K} \alpha_k f_k(x \mid \theta_k)$$  \hspace{1cm} (2.3)

where $\theta_k$ represents the parameters of component $k$ (i.e. density function).

2.4.2 Maximum Likelihood Estimate using EM Algorithm

Let us now assume that mixture model type has been selected with specified parameters and we are now in a position of estimating goodness-of-fit (GoF). In particular, we are interested in how well a selected mixture model fits our data. One of the most common approaches to evaluate GoF is to estimate parameters of the model that best fits the data [45]. Parameter estimation can
be achieved using two general methods; least-squares estimation (LSE) and maximum likelihood estimation (MLE). The former method is a standard approach in regression analysis and allows to estimate parameters by minimizing the sum of squared errors between the functional part of the model and the observed responses. LSE is useful for summarizing observed data by deriving a descriptive measure without making any kind of distributional assumptions for the population. However, this method does not provide any basis for a hypothesis testing, and have a strong assumption about the normal distribution of noise with equal variance [62].

On the contrary, MLE is a convenient statistical method that allows to estimate model parameters from the underlying population by optimizing the parameters values in order to increase the probability of obtaining the observed data. For the purpose of using maximum likelihood estimate, we should first specify the joint density function for all data points in a form:

\[ p(x_1, x_2, ..., x_n | \theta) = p(x_1 | \theta) * p(x_2 | \theta) * ... * p(x_n | \theta) \] (2.4)

We adjust Equation 2.4, so that we consider data points \( x_1, x_2, ..., x_n \) to be fixed and allow \( \theta \) variable of our function to vary:

\[ L(\theta | x_1, x_2, ..., x_n) = p(x_1, x_2, ..., x_n | \theta) = \prod_{i=1}^{n} f(x_i | \theta) \] (2.5)

Obtained new equation is called the likelihood function and the problem of mixture estimation from a given data \( x_1, x_2, ..., x_n \) can be defined as a task of calculating parameters \( \theta \), which provides the maximum likelihood estimate solution as following:

\[ \theta = \arg \max_{\theta} L(\theta | x_1, x_2, ..., x_n) \] (2.6)

Having identified our objection function (Equation 2.6) that we are trying to maximize, it is now time to explore the Expectation-Maximization (EM) algorithm, which is a commonly used method for calculating MLE with unknown model parameters [15].

The EM algorithm is an efficient iterative procedure, where each iteration defined by two major steps: the expectation (E-step) step and the maximization step (M-step). The E-step is responsible for the estimation of the missing data given the observed data as well as calculation of the current model parameters [8]. On the other hand, the M-step is maximizing likelihood function by re-estimating model parameters under the assumption that the missing data has been identified. Formally we can define EM algorithm as following:

**Definition 2.** (adopted and modified from Wikipedia) Given a statistical model with an underlying probability density function and a vector of unknown parameters \( \theta \) which generates a set \( X \) of observed data, and a set \( Z \) of unobserved latent data, the objective function in Equation 2.6 could be optimized by the marginal likelihood of the observed data represented as:

\[ L(\theta | X) = p(X | \theta) = \sum_{z} p(X, Z | \theta) \] (2.7)

The EM algorithm allows to maximize this marginal likelihood, iteratively applying E and M steps until convergence:

(E-step):

\[ Q(\theta | \theta^t) = E_{Z|X,\theta^t}[\log L(\theta | X, Z)] \] (2.8)

(M-step):

\[ \theta^{t+1} = \arg \max_{\theta} Q(\theta | \theta^t) \] (2.9)

\( ^4 \)adopted and modified from https://en.wikipedia.org/wiki/Expectation-maximization_algorithm
CHAPTER 2. PRELIMINARIES

We will avoid the derivation procedure of the EM algorithm as it goes beyond the scope of this thesis project, and interested reader is referred to [6] or [4]. However, it would be appropriate to make several notes:

- Initialization of the EM algorithm is a debatable topic among researchers of the machine learning field due to its tendency to converge to a local optima. We can either initialize this algorithm with a set of initial parameters and then continue with an E-step, or we can assume initial weights for every single data point following with a M-step. None of this approaches can guarantee a global optima solution that is why running the EM algorithm using multiple initial starting parameters helps to avoid the problem.

- Up until now we have not discussed a number of mixture model components and how to estimate them. Basically, we would like to determine an optimal number of clusters that we would like to fit with data using our mixture model approach. An optimal number of clusters for mixture modeling is a number of distribution function from which the underlying data could be drawn without under-fitting whole sample population. It turns out to be a highly important issue, because fitting our observations to the wrong number of mixture model components might produce misleading results. We address this problem in Chapters 3 and 4.

Having discussed major objection function for the probabilistic mixture models and the way of optimizing it, we are now in a position of applying this clustering methods on our sensor event data. In Section 3.2 we will discuss the circular nature of the timestamp attribute that we will utilize for the clustering purpose, followed by Section 3.3 where we will thoroughly examine von Mises mixture modeling.
Chapter 3

Label Refinement Method based on a Timestamp

Before explaining the technical implementation of our label refinement method, in this chapter we would like to describe major (preliminary) analysis steps that have been taken by us in order to understand the underlying structure the available sensor data. We will use van Kasteren and Philips datasets interchangeably, as they both represent the same data format of smart-home sensor data (Philips dataset being more large) in order to demonstrate major theoretical decisions made for the successful accomplishment of this project. In Chapters 1 and 2 we have already indicated that we will concentrate on the timestamp attribute of smart-home sensor events for generating label refinement method using unsupervised machine learning. In this chapter we will discuss the analysis of duration and starting-time features of sensor events that can be obtained from an available timestamp attribute within an event log.

In Section 3.1 we will talk about the histogram analysis performed for every single sensor type within the available datasets in order to understand whether the probabilistic mixture modeling is a right approach for clustering the data. Circular nature of a timestamp attribute has been explained in Section 3.2. In this section we will talk about the successful application of Gaussian Mixture Modeling on the duration feature domain of sensor events, which complies to the rules of linear statistics. In contrary, starting-time feature domain, which is circular in nature requires the transition to von Mises Mixture Modeling and will be covered in Section 3.3.

3.1 Preliminary Analysis of Smart-Home Sensor Data

To justify our intention to utilize a mixture model clustering, simple data analysis in a form of histogram modeling for a starting time of several sensor events has been performed by us. Figure 3.1 illustrates taken steps in order to perform a required preliminary analysis. In this particular case, we have utilized publicly available van Kasteren dataset [70], which contains only state based sensors (ON/OFF) that makes it easier to identify and display human behavior within the smart-home.

First, we retrieve all instance occurrences of every single sensor type within the dataset (in case of van Kasteren dataset the number of sensors is 14, each having multiply instances) and their starting times. Next, we construct a 1-dimensional histogram per sensor type, where x-axis indicates the starting time of every single sensor instance, and y-axis displays the occurrence frequency.
CHAPTER 3. LABEL REFINEMENT METHOD BASED ON A TIMESTAMP

Figure 3.1: Histogram analysis of smart-home sensor events from the van Kasteren dataset (23 days)

Figures 3.2 and 3.3 demonstrate the occurrence frequency of two event types 'Fridge' and 'Plates Cupboard', from the van Kasteren dataset during the day for a period of 23 days.

Figure 3.2: Van Kasteren dataset: standard occurrence of the 'Fridge' sensor event (23 days)
CHAPTER 3. LABEL REFINEMENT METHOD BASED ON A TIMESTAMP

From the figures above, one can immediately notice that the obtained distributions have at least two peaks (i.e. modes), which allows us to make an assumption about the multimodal nature of the underlying probability density function. Having this in mind, it would be appropriate to state that clustering a continuous multimodal density function could be achieved with the help of mixture modeling. Major idea behind this approach is that all observations of the same type come from the same distribution (i.e. belong to a certain sub-population) and any difference among them could be considered as a matter of chance [57]. Another noteworthy aspect of the probabilistic mixture modeling is that selected mixture of \( K \) component distributions (Equation 2.2) allows to conceptualize the trend within the population (i.e. what kind of distribution our data follows). Using statistical inference methods, we can indeed check if our old data follows the selected distribution type, and whether new data could be generated from it. Based on the obtained results we can either accept or reject a potential clustering model. From this perspective, an accurate selection of a probability density function should be considered as one of the top priorities.

In Chapter 1 we have stated that for the initial stage of this project we decided to concentrate on the timestamp attribute of our sensor events. In particular, we would like to cluster our sensor data points based on their time occurrence throughout the day, and Figures 3.2 and 3.3 support our intuition that we might obtain reliable clusters. For instance, within our smart-house environment we might have 10 different types of sensors (e.g. 'Microwave', 'Fridge', 'Toilet-Door') and for every single sensor type we would like to derive a canonical representation of its occurrences during the day. As a result, we might have a 'Microwave' sensor event type happening usually around 8:30 - 9:20 in the morning and 17:45 - 18:55 in the evening (for a given elderly person during the observation period). By considering the fact that in our work we are utilizing probabilistic mixture models, every single cluster will group time-based similar events and will be identified by the parametrized distribution function. Estimation of model parameters with the help of maximum likelihood estimate method will be explained below. Successful implementation of this clustering approach will constitute the backbone of our label refinement method and will help to decide how to refine a particular sensor type. Figure 3.4 illustrates our high level clustering approach towards the generation of sensor label refinement, and its implementation details will be covered in Chapter 4.
CHAPTER 3. LABEL REFINEMENT METHOD BASED ON A TIMESTAMP

In addition to the standard representation of sensor starting times, we have also performed a simple analysis of a duration for which a particular sensor type has been turned on. In order to do, we have analyzed a ‘TV’ sensor type from Philips dataset, which contains not only state based sensor, but also continuous sensor variables. Detailed explanation of both dataset could be found in Chapter 5. Figure 3.5 shows a histogram that represents how long on average does it take for a particular smart-home agent to watch a TV. In this figure, $x$-axis represents a duration of TV watching sessions in minutes and $y$-axis displays how frequently these durations can be observed.

From Figure 3.5, one might derive that there are different duration times that can be observed.
for a particular agent, while watching a TV. We might have an assumption that if an agent X watches a 'TV' for a period of 2-3 hours, it is very likely that he or she will go to sleep after (indication of 'Sleeping' activity). On the other hand, if an agent X watches a 'TV' for a period of 30-60 minutes, it is very likely that he or she will go out after (indication of 'Leaving the House' activity). This kind of behavioral assumptions should at least be supported by the preliminary data analysis, which will clearly indicate a structure for clustering (e.g. Figures 3.2, 3.3, and 3.5). Otherwise, a generation of label refinement method based on mixture modeling applied to unstructured feature domain might not help to produce a reliable result.

Duration analysis of sensor events from both van Kasteren and Philips datasets, allowed us to make several notes:

- Close examination of van Kasteren dataset showed that only few sensor, such as 'Frontdoor' and 'Hall-Toilet door' happen long enough to track down their duration (even for these sensor types a number of desirable events is not enough). Due to the fact that the majority of event types within this dataset are state-based, usually it takes only 0 to 1 seconds to change sensor states. Thus, it is impossible to obtain any insightful distributions, which makes the application of probabilistic mixture modeling irrelevant. Philips dataset does contain several sensor event types (e.g. 'TV' or 'Microwave'), which are identified by the continuous sensor values, and with a proper data pre-processing it is possible to derive a relatively long duration periods for such sensor types (e.g. Figure 3.5).

- Analysis of both starting times and durations of smart-home sensor events (for both dataset types) allows us to make an assumption about the feasibility of applying probabilistic mixture modeling for this kind of data. Our application domain will be a 1-dimensional feature domain, where we first examine the application of mixture modeling on duration attribute, followed by the starting time attribute.

3.2 Circular Nature of a Timestamp Attribute

Preliminary analysis of the underlying sensor event data showed that it is possible to utilize the concept of a mixture modeling. While application of standard mixture modeling approaches results into reliable model parameters for a linear data structures (e.g. duration of a sensor type events), transition to the circular feature domain (e.g starting time of events during a day) requires an appropriate distribution function that will help to fit the data around the clock circle. This section presents the required transformation of the probability density function that has been performed in our project.

3.2.1 Clustering 'Duration' of Smart-home Sensor Events

Before describing the notion of a circular data, it would useful to explore the application of Gaussian Mixture Model (GMM) clustering on the sensor event types represented by Figures 3.2, 3.3, and 3.5 in order to visualize the existing limitation. Before applying this type of mixture modeling one might have an assumption that the underlying nature of a feature domain (in our case 1-dimensional, duration and starting time attributes) is linear. In other words, the x-axis in Figures 3.2 and 3.3 that represents the standard occurrence of a particular sensor event throughout the day, and the x-axis in Figure 3.5 that represents the standard duration of sensor events are not continuous, and do not allow density functions to cross limit borders. While this is a valid assumption for a duration feature domain, in our following discussion we will show the limitation of this assumption for a starting time attribute that we would like to cluster.

Gaussian Mixture Modeling represents a common statistical choice for clustering multi-modal data populations as represented by Figures 3.2, 3.3, and 3.5. Gaussian (i.e. normal) distribution
could be denoted by $\mathcal{N}(\mu, \sigma^2)$, where a random variable $x$ is distributed normally with mean $\mu$ and variance $\sigma^2$, and the probability density function formulated by:

$$p(x \mid \mu, \sigma^2) = \frac{1}{\sigma \sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2}$$

By substituting the appropriate Gaussian distribution function within the Equation 2.3, mathematical representation of a mixture of $K$ Gaussian components defined as:

$$f(x) = \sum_{k=1}^{K} \alpha_k \mathcal{N}(X \mid \mu_k, \sigma_k^2)$$

where $X$ represents a dataset of $n$ elements $(x_1, ..., x_n)$ and $\alpha_k$, $\mu_k$ and $\sigma_k^2$ represent mixing weight, mean and variance of the $k^{th}$ Gaussian component, respectively.

Several R packages, such as ‘Mclust’ [19] or ‘Mixtools’ [3] exist that can help us to fit sensor event data into the required number of Gaussian components. Application of GMM on the duration of the ‘TV’ sensor type (Figure 3.5) has been illustrated by Figure 3.6. Fitting for 3 Gaussian mixture component has been performed with the help of `normalmixEM` function from the package ‘Mixtools’ in R, which is based on EM algorithm that we have discussed in Chapter 2. The number of mixture components has been determined by the Bayesian Information Criteria (BIC) and its detailed explanation can be found in Chapter 4. Estimated model parameters are presented in Table 3.1. Obtained results show that, usually, a particular elderly person watches a TV for a duration of 54, 121 and 225 (by considering the corresponding variances) minutes. Mixture modeling also indicates that watching a TV for almost 2 hours ($\mu_2$) represents the major tendency for the smart-home agent (highest $\alpha_2 = 0.53$) with a variance of 33 minutes ($\sigma_2^2$).

Table 3.1: Estimated parameters for the ‘TV’ duration using GMM clustering.

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\mu_3$</th>
<th>$\sigma_1^2$</th>
<th>$\sigma_2^2$</th>
<th>$\sigma_3^2$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>‘TV’</td>
<td>53.68</td>
<td>120.69</td>
<td>224.95</td>
<td>12.27</td>
<td>32.56</td>
<td>17.37</td>
<td>0.30</td>
<td>0.53</td>
<td>0.17</td>
</tr>
</tbody>
</table>

1https://en.wikipedia.org/wiki/Normal_distribution
3.2.2 Clustering 'Starting-Time’ of Smart-home Sensor Events

Application of GMM on the duration of 'TV' sensor from Philips dataset showed that it is possible to obtain a clustering structure of sensor population using probabilistic mixture modeling. Data points that belong to one of these clusters similar to each other based on the density distribution, and their further appearance within the event-log could be refined according to the cluster type they belong to.

However, after trying to apply the very same GMM method on the starting time of these events, we face the problem of wrong distribution type selection. In order to illustrate the problem, we will fit sensor events types 'Fridge' and 'Plates Cupboard' from van Kasteren dataset. Obtained results are described by Figures 3.7 and 3.8, and estimated model parameters for both sensor types can be found in Table 3.2.

![Figure 3.7: Clustering 'Fridge’ sensor event using Gaussian Mixture Model (2 components)](image1)

![Figure 3.8: Clustering 'Plates Cupboard’ sensor event using Gaussian Mixture Model (2 components)](image2)
Table 3.2: Estimated parameters for the 'Fridge' and 'Plates Cupboard' event types using GMM clustering.

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>$\mu_1$</th>
<th>$\mu_2$</th>
<th>$\sigma_1^2$</th>
<th>$\sigma_2^2$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>'Fridge'</td>
<td>9.05</td>
<td>20.1</td>
<td>3.01</td>
<td>1.64</td>
<td>0.41</td>
<td>0.59</td>
</tr>
<tr>
<td>'Plates Cupboard'</td>
<td>9.7</td>
<td>20.00</td>
<td>2.95</td>
<td>1.49</td>
<td>0.56</td>
<td>0.44</td>
</tr>
</tbody>
</table>

Despite the fact that Gaussian mixture modeling has successfully identified major density distributions for both "Fridge" and "Plates Cupboard" sensor event types, careful reader should notice the distorted variance of the red Gaussian distribution in Figures 3.7 and 3.8. Major reason behind this is that the underlying EM algorithm assumes the wrong distance between events that used to happen right before and after the midnight (almost 24 hours vs. couple of minutes/seconds). Sensor events that occurred right after the midnight are closer to the mean of the red component rather than blue component (they are only closer in terms of euclidean distance, they are not closer in terms of distance on the clock), if to consider the x-axis as a finite dimension from 00:00 to 23:59 (starting position of the Gaussian components does not affect the final outcome). The EM algorithm then re-estimates $\mu$ and $\sigma^2$ parameters of the red Gaussian component by including those misleading data points (probability of coming from red component is higher for those points).

We have already stated that utilization of an appropriate probability density function is crucial for a successful application of a mixture modeling. From this perspective, the nature of a feature domain that we are trying to cluster should be evaluated correctly.

Conducted research showed that for clustering starting times of sensor events we have to switch from the linear statistics to the circular one, which allows us to represent every single data point on the circumference of the unit circle. Circular data can arise in many different ways (e.g. compass direction or clock time), and every single data point can be defined by the angle from the initial direction to a point on a circle [43]. The notion of circularity allows us to overcome the problem introduced by the linear variable which is used to define a circular data point on the real line. As we move away from the origin along the real line, a linear variable increases, and as a result value of 355 is close to a value of 350, however considered far from the value of 0 (i.e. origin) [48].

In order to demonstrate what kind of problems may occur if we treat our circular data as being linear, let us assume that we have two 'Microwave' sensor events happening at 23:45 (hour-float 23.75) and 00:05 (hour-float 0.08). By using Equation 3.3, we can convert our time attributes to an angular measurement (356.25° and 1.2°, respectively) on a unit circle:

$$a = \frac{(360°) \times \left(\frac{X}{k}\right)}{k}$$

where $k$ is the number of units on the circular scale (24 for hours, 356 for days of a year) and $X$ is a hour-float. Normally, we would assume that these two sensor events should be grouped together due to their close occurrence times. However, if one will take the arithmetic mean of these angles (178.73°) the resultant vector direction will be in a totally opposite direction. This is a common mistake for a traditional clustering algorithms, such as GMM or K-means which assume a finite linearity of a feature domain. It turns out that the representation of a circular data in a form of histograms also contributes to the wrong interpretation of the underlying data type. Thus, in this report we will illustrate the distribution of sensor events around a natural circular plot, where 0° represents 00:00 and our day proceeds in a counter-clockwise direction (Figure 3.9).

As one might already understand, blue dots around the circle reflect the time-based distribution of the particular sensor events that have been converted to a corresponding angular measurements. On the other hand, dashed red line represents a kernel density estimates, which allows to estimate the underlying population density around the circle in a non-parametric fashion. [48]. It would
be appropriate to state that the representation of a circular data and other statistical analysis applied for this kind of data type has been achieved with the help of 'circular' package in R [1].

![Circular representation of sensor event types](image)

Figure 3.9: Circular representation of the 'Fridge' (left) and 'Plates Cupboard' (right) sensor event types throughout the day for the observed period of time (23 days)

### 3.3 Clustering using von Mises Distribution

In directional statistics, the von Mises distribution plays a role of a continuous distribution around the circle. The von Mises distribution is considered as a special case of the von Mises-Fisher (vMF) distribution that is a probability distribution on the (n - 1) dimensional sphere in $\mathbb{R}^n$. We can define the probability density function of the von Mises distribution for a given angle $\theta$ using the formulation from Mardia and Jupp [43]:

**Definition 3.** (adopted from Mardia and Jupp [43] and modified) Given a von Mises distribution $M(\mu, \kappa)$, where $\mu$ and $\kappa$ represent mean direction and concentration parameters, respectively, probability density function is defined as following:

$$g(\theta | \mu, \kappa) = \frac{1}{2\pi I_0(\kappa)} e^{\kappa \cos(\theta - \mu)}, \quad 0 \leq \theta \leq 2\pi, \quad 0 \leq \mu \leq 2\pi, \quad \kappa \geq 0$$

(3.4)

where $I_0$ represents the modified Bessel function of order 0 and can be defined as:

$$I_0(k) = \frac{1}{2\pi} \int_0^{2\pi} e^{\kappa \cos(\theta)} d\theta$$

(3.5)

The shape of the von Mises distribution is illustrated in Figure 3.10. It is important to state that:

- As $\kappa$ approaches to 0 value, the distribution becomes uniform around the circle.
- As $\kappa$ increases, the distribution becomes relatively concentrated around the angle direction $\mu$ and takes the form of a normal distribution.
Our intuition to apply exactly this clustering method is inspired by the work of Banerjee et al. [2] who proposed a generative mixture model approach to cluster data points that lie on the unit sphere by utilizing von Mises-Fisher distribution. The authors explored the possibility of applying the Expectation Maximization framework in order to estimate the concentration and mean parameters of a given mixture model. Their application domain consists of high-dimensional text and gene-expression data for which they have successfully estimated required model parameters given a finite number of vMF components. In our project, we will show that it is possible to apply their method on a data, which is distributed around the unit circle.

Before describing the application of this method to our project, it would be appropriate to quickly review the vMF distribution and MLE of it parameters as indicated in the paper by Banerjee et al [2]:

**Definition 4.** (adopted from Banerjee et al. [2] and modified) Given an arbitrary $d$-dimensional unit vector $x$, its probability density function for the $d$-variate vMF distribution can be described as following:

$$f(x \mid \mu, \kappa) = C_d(\kappa)e^{\kappa \mu^T x} \quad (3.6)$$

where $k \geq 0$, $d \geq 2$, $\mu^T \mu = 1$, and the normalizing constant $C_d(\kappa)$ defined as:

$$C_d(\kappa) = \frac{\kappa^{d/2 - 1}}{(2\pi)^{d/2}I_{d/2 - 1}(\kappa)} \quad (3.7)$$

where $I_a$ stands for the modified Bessel function of order $a$, and density function $f(x \mid \mu, \kappa)$ is parametrized by $\mu$ (mean direction) and $\kappa$ (concentration level around $\mu$).

Let us now assume that $X$ is a finite set of datapoint unit vectors that follow the vMF probability distribution function in Equation 3.6, our major goal is to find MLE of $\mu$ and $\kappa$ parameters for a finite number of components:

**Definition 5.** (adopted from Banerjee et al. [2] and modified) Given $X$ and by assuming that every single datapoint $x_i$ is independent and identically distributed, we can define our log-likelihood function for $X$ as:

$$\ln P(X \mid \mu, \kappa) = n * \ln C_d(\kappa) + \kappa \mu^T r \quad (3.8)$$

---

2 adopted from https://en.wikipedia.org/wiki/Von_Mises_distribution
where \( r = \sum_i x_i \) and MLE for \( \mu \) and \( \kappa \) can be calculated using:

\[
\hat{\mu} = \frac{r}{\|r\|} = \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} x_i} \quad \text{and} \quad (3.9)
\]

\[
\hat{r} = \frac{\|r\|}{n} = \frac{I_{d/2}(\hat{\kappa})}{I_{d/2-1}(\hat{\kappa})} \quad (3.10)
\]

Several notes should be made before proceeding further:

- Equations 3.8, 3.9, 3.10 are defined for a single vMF component and should be refined in order to cover the mixture of vMFs components.

- According to the authors of [2], it is not possible to obtain an analytic solution for the estimation of \( \kappa \) (Equation 3.10), and asymptotic methods should be explored in order to obtain an accurate approximation of this model parameter. In our project we will use the approximation method suggested by Dhillon and Sra [16] and Banerjee.

Having identified the objectives of the EM algorithm framework in Section 2.4 and by representing our mixture of \( K \) vMFs components in form:

\[
h(x | \theta) = \sum_{k=1}^{K} \alpha_k f(x | \theta_k) \quad (3.11)
\]

it is possible to fit our data to these finite number of components using the following E-M steps [26]:

- **E-step**: calculates a-posteriori probabilities of datapoint belonging to a certain mixture component:

  \[
p(k | x_i, \theta) \propto \alpha_k f(x_i | \theta_k) \quad (3.12)
  \]

- **M-step**: maximizes the expected complete-data log-likelihood by re-estimating parameters per every single vMF component:

  \[
  \hat{\alpha}_k = \frac{1}{n} \sum_{i=1}^{n} p(k | x_i, \theta) \quad (3.13)
  \]
  \[
  \hat{\mu}_k = \frac{\sum_{i=1}^{n} p(k | x_i, \theta)x_i}{\| \sum_{i=1}^{n} p(k | x_i, \theta)x_i \|} \quad (3.14)
  \]
  \[
  \frac{I_{d/2}(\hat{\kappa}_h)}{I_{d/2-1}(\hat{\kappa}_h)} = \frac{\| \sum_{i=1}^{n} p(k | x_i, \theta)x_i \|}{\sum_{i=1}^{n} p(k | x_i, \theta)} \quad (3.15)
  \]

- Repeat iteratively E and M steps until convergence criteria is met, which could be defined as an absence of a significant relative absolute change in the log-likelihood.

Implementation of the EM algorithm for a finite set of vMF mixture components has been presented in [26] and exists in a form of an R package 'movMF'.

Having briefly introduced the mathematical foundation behind the mixture modeling using vMFs, in Chapter 4 a complete clustering pipeline that represents a set of steps required for the successful generation of a reliable label refinement function will be presented.
Chapter 4

Implemented Framework

In this chapter we will talk about the clustering pipeline using von Mises Mixture Modeling in order to generate a label refinement method based on the notion of circular data. A starting-time feature derived from the timestamp attribute of smart-home sensor events will be utilized throughout this chapter that will help to demonstrate how to fit circular data around the clock to the mixture of von Mises components. Overall clustering methodology has been illustrated in Figure 4.1, and the detailed explanation of every component (rounded rectangle) and their intermediary steps (oval nodes) will be presented in the subsequent sections.

Figure 4.1: Clustering steps performed for the generation of a label refinement method

In order to have a complete statistical approach for the analysis of a sensor event data and to be able to obtain reliable clusters using the selected unsupervised approach the following steps have been defined by us:

- **Data-Model Pre-fitting Stage:** A big issue within the unsupervised machine learning
CHAPTER 4. IMPLEMENTED FRAMEWORK

branch is that all clustering algorithms will eventually return clusters despite the fact that there might not be any. A so called cluster tendency assessment should be performed in order to evaluate the underlying structure of a given data set. Many works ignore this stage by assuming its weak effect on a final result, however in our project we will tackle this issue by the help of statistical inference. Moreover, having identified that there is a potential tendency for clustering, the natural question is how many of them could be identified. Due to the fact that the performance of the mixture modeling is closely related to the number of components selected, this issue will be also addressed in this section. Section 4.1 will cover this part of the clustering methodology.

- **Data-Model Fitting Stage:** At this stage the actual clustering using vMFs mixture components will be performed. Maximum likelihood estimates will be obtained by the help of the EM algorithm in order to fit our sensor event data to a given number of vMFs distributions. Detailed explanation of this step can be found in Section 4.2.

- **Data-Model Post-Fitting Stage:** At this final stage of our clustering pipeline, the goodness-of-fit test will be performed in order to check whether the data could have been drawn from the fitted von Mises distributions. Visual assessment using P-P and Q-Q plots, as well as Watson $U^2$ statistics will be thoroughly examined at this particular stage. Evaluation of clustering results using both approaches have been explained in Section 4.3.

4.1 Data-Model Pre-Fitting Stage

In order to illustrate different distribution types around the circle, and how that may affect our clustering decisions, let us examine Figure 4.2. Due to the fact that probabilistic mixture modeling requires a precise specification of a number of von Mises components to fit the data, before applying this algorithm we should be able to differentiate between three distributions types.

![Figure 4.2: Uniform, Unimodal and Multimodal distribution types around the circle](image)

Uniform distribution simply means that our data points (in our case sensor events) are randomly distributed and there is no any meaning of applying any kind of clustering algorithm. In contrary, unimodal and multimodal distribution types signal for a possible cluster structure and more sophisticated statistical methods are required in order to determine a reliable number of components.

Data-Model Pre-fitting stage within our clustering pipeline can be divided into three major steps represented in Figure 4.3. Each of this steps will be discussed in detail and its effect on the successful clustering implementation will be thoroughly examined.
CHAPTER 4. IMPLEMENTED FRAMEWORK

4.1.1 Uniformity Check - Rao’s Spacing Test

Let us now start with a ‘Uniformity Check’ step, which will help us to estimate whether our sensor data is uniformly distributed around the circle. In order to do so, we will conduct Rao’s Spacing Test, which has been developed by J.S. Rao in his doctoral dissertation [51]. This statistical method is based on the idea that circular data is uniformly distributed around the circle, and N number of observations are separated from each other for a 360/N distance unit. In case our sensor data fails to pass this null hypothesis, sign for a particular directionality is generated for us:

**Definition 6.** (adopted from J.S. Rao [51]) Given a number of \( n \) observations, the test statistics \( U \) for Rao’s Spacing Test is defined as:

\[
U = \frac{1}{2} \sum_{i=1}^{n} |T_i - \lambda| \tag{4.1}
\]

where \( \lambda = \frac{360^\circ}{N} \), \( T_i = f_{i+1} - fi \) for \( 1 \leq i \leq n - 1 \) and \( T_n = (360^\circ - f_n) + f_1 \) for \( i = n \)

In order to run this statistical test, which is based on the null hypothesis assumption of uniformity, we will utilize `rao.spacing.test(x, alpha)` function from the ‘circular’ package in R. For this function, variable \( x \) specifies a vector that contains our circular datapoints and variable \( \alpha \) represents a numeric value that defines the significance level of the test (\( \alpha = 0.01 \) in our case). A range for the p-value, which is required for the indication of the significance test is determined using the table of simulated critical points [54]. It would be appropriate to indicate that for any statistical test used, our goal is not to prove the correctness of the alternative hypothesis, but rather indicate its plausibility in comparison to the null hypothesis.

As an example of the application of Rao’s Spacing Test on the ‘Fridge’ sensor event data, Figure 4.4 demonstrates the test result.

```r
> rao.spacing.test(antecedentEventStartHoursRadial, 0.01)
Rao's spacing test of uniformity
Test statistic = 242.616
Level 0.01 critical value = 148.84
Reject null hypothesis of uniformity
```

Figure 4.4: Rao’s Spacing Test for the ‘Fridge’ sensor event type
Chapter 4. Implemented Framework

Test statistic value of 242.616 is way above the critical value of 148.84 for the level 0.01, which basically allows to reject the null hypothesis of uniformity and proceed to the next step of our statistical data analysis.

4.1.2 Multimodality Check - Hartigan’s Dip Test

Having identified that our sensor event data is not uniformly distributed around the circle, it is now time to perform a bimodality test by the help of Hartigan’s Dip Test. This statistical test is based on the null hypothesis that our data is unimodal, and failing to pass this hypothesis allows us to assume that our data is at least bimodal (i.e. two distribution modes). According to Hartigan [23], dip statistic helps to measure the maximum difference between the empirical distribution function (EDF) and the unimodal distribution function (UDF), which tries to minimize the maximum difference, and this is done over all data points.

We will omit the mathematical foundation behind this statistical test and the interested reader is referred to the original paper by Hartigan. However, it will be useful to illustrate the application of this test on two different data distribution types; "Fridge" sensor event type from the van Kasteren dataset (expected to be at least bimodal by visual evaluation of Figure 3.2) and randomly generated unimodal von Mises distribution. Obtained results are shown in Figures 4.5 and 4.7 with corresponding statistical test results (Figures 4.6 and 4.8). It would be appropriate to state that the implementation of Hartigan’s Dip test can found a R package ‘diptest’ [40].

![Figure 4.5: Visualization of empirical cumulative distribution function for the 'Fridge' sensor event ('diptest')](image)

```
> dip.test(antecedentEventStartHoursRadial)
Hartigans' dip test for unimodality / multimodality
data: antecedentEventStartHoursRadial
D = 0.11467, p-value < 0.00000000000000022
alternative hypothesis: non-unimodal, i.e., at least bimodal
```

Figure 4.6: Hartigan’s Dip Test for the 'Fridge' sensor event type
CHAPTER 4. IMPLEMENTED FRAMEWORK

Figure 4.7: Visualization of empirical cumulative distribution function for the unimodal von Mises distribution

Figure 4.8: Hartigan’s Dip Test for the unimodal von Mises distribution

'diptest' package allows to evaluate the underlying nature of our observations both visually and statistically. Figures 4.5 and 4.7 illustrate the main difference between unimodal and bimodal (at least) distribution types by representing the empirical cumulative distribution function (y-axis) over the whole sample space (x-axis: time of event occurrence converted to radians). Steep sloped regions that can be observed on both figures indicate a dense distribution of samples, whereas shallow sloped regions is a sign of a sparse distribution.

One can easily see that ‘Fridge’ sensor event type shows a dense aggregation of samples around two periods, which is also supported by a significance test where p-value is small enough to reject the null hypothesis of unimodality. In contrary, randomly generate von Mises distribution (\(\mu = 1, \kappa = 10\)) shows a dense aggregation of points only around one period, which is also reflected in its high p-value that does not allow to reject the null hypothesis.

4.1.3 Number of Component Selection - Bayesian Information Criteria

Rao’s Spacing Test and Hartigan’s Dip Test allowed to filter out sensor event types that do not represent any particular clustering structure. As a result, further analysis of these event types will have no or very little impact on the pattern discovery of a smart-home agent.

Having identified that certain event types represent at least bimodal distribution (i.e. at least
two clusters might be identified), one now in a position to estimate a reliable number of von Mises mixture components that will fit our data.

Number of components (i.e. cluster) selection is one the most debatable topics for the unsupervised branch of machine learning. A variety of direct (e.g. average silhouette width, elbow) and testing (e.g. gap statistic) methods exist that allow to determine an optimal number of clusters. However, their application on a circular type of data has been explored extensively. We should also state that as for any other unsupervised machine learning approach, optimal clustering identification problem is a relatively subjective task, and no global solution has been found so far [55].

In our project, we decided to utilize the Bayesian information criterion (BIC), which is statistical model selection method among a finite number of models. Having in mind that our primary objection for the probabilistic mixture modeling is to increase the log-likelihood, there is always a possibility to over-fit the data by adding more parametrized components to our mixture. Bayesian information criterion allows to overcome this problem by introducing a penalty for an increasing number of parameters for our model.

The BIC can be interpreted as a formal implementation of principle of Occam’s Razor, which basically requires to select the simplest model that allows to adequately describe the underlying data. This information criterion was developed by Schwarz in 1978 [56] and can be formally defined as (using the formulation from [76]):

\[
BIC = -2 \times \ln \hat{L} + k \times \ln(n) \tag{4.2}
\]

where \(\hat{L}\) is a maximized value for the likelihood function that can be described as \(f(y | \hat{\theta}_k)\), \(n\) is a sample size, and \(k\) is the number of parameters to be estimated. In order to decided which model to select (i.e. how many von Mises mixture components to fit), the one with the lowest information criterion should be considered first. Authors of [30] suggest to accept or reject the evidence against the model with the higher BIC according to Table 4.1. In our project, we will accept \(K\) components with a lower value of BIC if \(\Delta BIC < 10\).

<table>
<thead>
<tr>
<th>(\Delta BIC)</th>
<th>Evidence against higher BIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 2</td>
<td>Not worth more than a bare mention</td>
</tr>
<tr>
<td>2 - 6</td>
<td>Positive</td>
</tr>
<tr>
<td>6 - 10</td>
<td>Strong</td>
</tr>
<tr>
<td>&gt; 10</td>
<td>Very strong</td>
</tr>
</tbody>
</table>

We should indicate that in case of von Mises mixture models, complexity of the clustering model is increased by the number of component parameters, which are the mean direction (\(\mu\)), the concentration around the mean (\(\kappa\)) and the mixture weight (\(\alpha\)).

In order to illustrate the role of the Bayesian information criterion in our project, let us examine Figures 4.9 and 4.11. Change in Bayesian information criterion for sensor event types ‘Plates Cupboard’ and ‘Hall-Toilet door’ from the van Kasteren dataset has been presented in both figures.

Utilization of the Bayesian information criterion allowed us to determine that it is plausible to fit two components for the ‘Plates Cupboard’, and three components for the ‘Hall-Toilet door’, respectively (based on the \(\Delta BIC\) value). This procedure is performed for all sensor events types that have successfully passed uniformity and unimodality test, and now in a position of estimating a reliable number of possible clusters. BIC does not provide any information about the estimated model parameters, and we will continue this discussion in Section 4.2.
CHAPTER 4. IMPLEMENTED FRAMEWORK

Figure 4.9: BIC change for the increasing number of von Mises mixture components - 'Plates Cupboard' (van Kasteren dataset)

```
> set.seed(111)
> bic.mixvmf(wait, A = 9)
$sbic
  1  2  3  4  5  6  7  8  9
234.3570 155.7783 151.6620 168.3131 202.8424 175.5488 220.0570 205.6331 218.8555
```

Figure 4.10: BIC values for the number of components from 1 to 9 - 'Plates Cupboard' (van Kasteren dataset)

Figure 4.11: BIC change for the increasing number of von Mises mixture components - 'Hall-Toilet door' (van Kasteren dataset)

```
> set.seed(111)
> bic.mixvmf(wait, A = 9)
$sbic
  1  2  3  4  5  6  7  8  9
700.1211 578.0433 560.9719 581.2001 586.1773 606.3349 635.9041 693.9748 666.4859
```

Figure 4.12: BIC values for the number of components from 1 to 9 - 'Hall-Toilet door' (van Kasteren dataset)
Implementation of BIC for the mixture of von Mises-Fisher distributions (special case, the mixture of von Mises distributions) could be found in the R package 'Directional' in a form of \( \text{bic.mixvmf}(x,n) \) function. For both sensor event types, we iteratively estimated BIC values by increasing the number of mixture components, and selected the model with \( K \) components where no more significant change in \( \Delta BIC \) can be observed.

### 4.2 Data-Model Fitting Stage

In the previous section we have thoroughly examined the way of exploring the underlying nature of our sensor events types. As it has been stated earlier, the major objective of the 'Data-Model Pre-fitting' stage is to discard sensor events that do not provide any meaningful insights for the human behavior recognition. This procedure is performed based on the statistical inference methods that have been developed for a circular data by considering only event time occurrence distribution throughout the day.

In this chapter, we will concentrate on the sensor events that have successfully passed the 'Data-Model Pre-fitting' stage and we show how to estimate the parameters of a von Mises mixture model. In order to do so, we will utilize an R package 'movMF' [26], which allows to fit a mixture of von Mises-Fisher distributions (unit sphere), and as a special case a mixture of von Mises distributions (unit circle). The number of components to be fitted is determined by the Bayesian information criterion that we have already discussed in the previous section.

#### 4.2.1 Data pre-processing for a circular data

Small data pre-processing is required in order to be able to apply the 'movMF' package on our sensor data. Providing a one-dimensional vector of angles that represents a corresponding event occurrence time around the clock is not sufficient for the application of 'movMF', which requires at least two-dimensional data representation.

In order to tackle this problem, the angles must be converted to points on the unit circle (from a contact with Kurt Hornik and Bettina Gruen, the authors of the movMF package). Basically, we need to convert Polar coordinates (radius, angle) into Cartesian coordinates (x-axis, y-axis) using equations:

\[
\begin{align*}
x &= \text{radius} \times \cos(\text{angle}) \\
y &= \text{radius} \times \sin(\text{angle})
\end{align*}
\]

where radius = 1 due to the fact that we are fitting the sensor data around the unit circle.

Figure 4.13 illustrates the data transformation for a small portion of the 'Plates Cupboard' event type from van Kasteren dataset.

![Figure 4.13: Transformation from Polar to Cartesian coordinates, 'Plates Cupboard' sensor event, van Kasteren dataset](image)
It would be appropriate to state that the application of the 'movMF' package will be performed on the last two columns, which allows to represent every single data point around the circle using its Cartesian coordinates.

4.2.2 Fitting a Mixture of von Mises Distributions

After performing a required data transformation, it is now time to apply our unsupervised method in order to fit a circular sensor data to a mixture of von Mises distributions. First, we apply clustering approach on the 'Plates Cupboard' sensor event type, and our intention is to fit two von Mises distributions as has been suggested by Figures 4.9 and 4.10. Figure 4.14 illustrates the obtained result.

Figure 4.14: Clustering the 'Plates Cupboard' sensor events by fitting two von Mises distributions (van Kasteren dataset)

As can be seen from the Figure 4.14, points around the midnight are clustered together, which is one the most important characteristics we are striving for. Every single data point belongs to one of the clusters to the certain degree of probability (Figure 4.15, rows - datapoints, columns - clusters).

Figure 4.15: Probability assignment to the 'Plates Cupboard' sensor events by fitting two von Mises distributions (van Kasteren dataset)

Convergence of the EM algorithm has been achieved in 10 iterations with a final log-likelihood
equal to 31.58. Estimated model parameters are shown in Table 4.2.

Table 4.2: Estimated parameters for a mixture of von Mises distributions, 'Plates Cupboard', (van Kasteren dataset)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>$\alpha$</th>
<th>$\mu$ (radii)</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>0.51</td>
<td>5.24</td>
<td>2.44</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>0.49</td>
<td>2.57</td>
<td>3.91</td>
</tr>
</tbody>
</table>

Obtained results show that Cluster 1 (red) covers the 51% of the sensor event data, having mean direction value of 5.24 (300.23 in angles, 20.02 in hour-float) and a relatively less concentration value of 2.44 around the mean. In contrary, Cluster 2 (blue) covers the 49% of the whole population, and has a mean direction value of 2.57 (147.25 in angles, 9.82 in hour-float) with a concentration value of 3.91. Lower $\kappa$ value indicates that a dispersion of sensor events around the mean for the Cluster 1 is higher in comparison to the Cluster 2.

It is important to provide some trigonometric interpretation of the $\mu$ value and be able to differentiate between mean resultant vector and mean direction.

**Definition 7.** (adopted and modified from [48]) Given a sample of circular observations $n$ in a form of unit vectors $a_1, a_2, ..., a_n$ and corresponding angle values $\theta_1, \theta_2, ..., \theta_n$, constants $\hat{C}$ and $\hat{S}$ can be defined as:

$$\hat{C} = \frac{1}{n} \sum_{i=1}^{n} \cos \theta_i \quad \text{and} \quad \hat{S} = \frac{1}{n} \sum_{i=1}^{n} \sin \theta_i$$

and mean resultant vector $\hat{R}$ is defined as:

$$\hat{R} = \sqrt{\hat{C}^2 + \hat{S}^2}$$

and mean direction $\theta$ is defined as:

$$\hat{\theta} = \text{atan2}(\hat{S}, \hat{C})$$

We should indicate that the $\text{atan2}$ function returns values in a range $(-\pi, \pi]$ and in order to map this values to $[0, 2\pi)$ it is necessary to add $2\pi$ to the negative values [48]. Mean direction $\theta$ is a vector direction that we are interested in and we would like to estimate it with the help of 'movMF()' function. In addition, we should state that the approximation value of a concentration parameter $\kappa$ has been obtained with the help of methodology suggested by Banerjee et al [2], however 'movMF' package allows to use different approximations techniques. Interested reader is referred to [26], Section 2.2.

Having successfully identified (this claim is made without a goodness-of-fit tests that will be described in the following section) a mixture of two von Mises distributions for the 'Plates Cupboard' sensor event type and its corresponding parameters, let us now examine a mixture of three von Mises components Figure 4.16.

Table 4.3 illustrates the mixture model parameters estimated after fitting our 'Hall-Toilet door' sensor data to three von Mises distributions. We will omit the detailed explanation as we have already discussed the meaning of model parameters for a mixture of two von Mises distributions. Instead, it would be appropriate to remind to our reader that a combination of obtained clusters (components) defines a canonical representation of a particular sensor event type throughout the day and for an observed period of time. This clustering approach allows us to define a number of possible splits per sensor event type (e.g. 'Hall-Toilet door' into three, 'Plates Cupboard' into two) and we consider these splits to represent a different human behavior that will help us to obtain more fine-grained process models.
4.2.3 Resolution of an Ambiguous Sensor Events

In order to fully complete our clustering approach, we should tackle one of the existing limitations. To illustrate this limitation, let us now have a look on Figure 4.17. As we are utilizing a probabilistic clustering approach, points which rely within the black circles (i.e. ‘Problematic Regions’) depicted in the figure show a tendency for both clusters. In other words, so called ambiguous sensor events types lie on the intersection of two von Mises distributions and it is not reliable to assign those kind of data point to one of clusters due to a small difference in their probability values (To illustrate this issue we have switched from van Kasteren dataset to Philips dataset due to the fact that the latter one is larger and has more dense distribution of sensor events for clustering).

Refined event logs that will be generated by applying our clustering approach will be fitted to process mining algorithms in order to obtain reliable and meaningful process models. Those algorithms do not consider a probabilistic nature of the underlying sensor events and only concerned about their occurrence frequency throughout an event log (original and refined).

In order to tackle this problem, we decided to use a supervised machine learning and train our potential classifier based on the features on sensor events that fall within two standard deviations (95.45%) range of obtained von Mises distributions (we have already stated that with increasing value of $\kappa$ parameter, von Mises distributions approaches to normal distribution around the circle). Trained classifier will help to predict ambiguous points at tails. Overall process can be observed from Figure 4.18, which requires several notes to be made:
As we have already pointed out, only sensor events with a solid probability assignment (> 95% membership value) participate in the classification model building process.

Set of solid clusters members then further divided into training and test sets having 70% and 30% division rate (increasing number of test sets allows to evaluate model performance on unseen data).

In order to maintain the fairness of this process, we eliminate our initial time attribute that we have used for the clustering and build a supervised model based on the new feature domain. For now, we decided to concentrate on the sensor events that precede and follow our major sensor event that we have clustered.

We strive for a high accuracy (prediction power) of our classification model, and in order to do so, prune our tree based on the cross-validation [44] method to avoid over-fitting on test data. The idea is to generate a small tree (i.e. interpretable) with a low value of cross validated error (Figure 4.19). Implementation of the decision tree used can be found in an R package ‘rpart’ [60] (in particular functions rpart(), printcp(), plotcp()).

Obtained classification model is used for the ambiguous sensor events (with a membership value ≤ 95%) in order to obtain their final cluster assignment. This information is used to adjust cluster memberships and finalizes our probabilistic mixture modeling.
CHAPTER 4. IMPLEMENTED FRAMEWORK

Figure 4.18: Supervised model using decision trees to address ambiguous points at the intersection of two von Mises distributions

Figure 4.19: Graphical representation of the cross validated error to prune the decision tree in Figure 4.20 (low cross validated error (xerror) is preferred)
Figure 4.20 illustrates a decision tree that has been obtained based on the features 'precedingEvent' and 'followingEvent' of the 'Refrigerator' sensor event type for the cluster 2 (green) and 3 (blue) 4.17. As can be derived from their names, these two features describe the events that directly precede and directly follow the 'Refrigerator' sensor event within each cluster. One of the major advantages of a decision tree classification is that an obtained model is self-explanatory. From the figure below, one can easily derive that if 'Refrigerator' sensor event in cluster 3 is not followed by events '11' (Water Cooker) and '21' (Cutlery) and not preceded by event '11' (Water Cooker) there is a possibility of 78% that it is a member of cluster 3.

Moreover, it would be appropriate to present the model performance statistics for the decision tree classification model shown above. In order to do so, we have build a confusion matrix that allows to calculate accuracy of model prediction and other useful statistics (e.g. kappa, sensitivity, specificity) by applying our classification model on the test set. Obtained results are shown in Figure 4.21.

- First, the accuracy (Accuracy = (TN + TP) / (TN+TP+FN+FP)) of our prediction model for these two clusters (2 and 3) of the 'Refrigerator' sensor event type is sufficient enough (0.8082) to be utilized in the further classification of the ambiguous sensor events (those
which rely between green and blue clusters, Figure 4.17). (note: assignment of clusters to the ‘positive’ or ‘negative’ membership can be interchanged)

• Sensitivity = TP / (TP + FN) = (Number of true positive assessment)/(Number of all positive assessment). In our case the sensitivity value is equal to 0.6606 (72 + / ( 72 + 37)).

• Specificity = TN / (TN + FP) = (Number of true negative assessment)/(Number of all negative assessment). In our case the specificity value is equal to almost 0.8962.

• Kappa statistics, which allows to compare an observed accuracy with an expected accuracy (a matter of a random chance) is equal to almost 0.576 and can be considered moderate [29].

To complete our evaluation of the classification model, let us finally present receiver operating characteristic (ROC) curve, which plots the true positive rate (TPR) against the false positive rate (FPR). ROC analysis is useful for the evaluation of the trade-off between sensitivity and specificity. Figure 4.22 shows the obtained ROC curve for our classification model with a corresponding area under curve (AUC) value. Being a quantitative interpretation of a ROC curve, AUC value allows to estimate whether the classifier will rank a randomly chosen positive instance higher than a randomly chosen negative instance [17].

![ROC curve and AUC value for the generated classification model](image)

Figure 4.22: ROC curve and AUC value for the generated classification model

Newly generated decision tree model can be now used to classify an ambiguous sensor events that lie on the intersection of two von Mises distributions (clusters 2 and 3). Cluster memberships of those data points are adjusted accordingly, and we are now ready to assess the goodness-of-fit our mixture model. We should also state that this supervised learning approach is required when there is no clear separation of clusters. For instance, from Figure 5.7 one can easily derive two distinct clusters with a clear separation among the cluster members.

4.3 Data-Model Post-Fitting Stage

The goodness-of-fit is a commonly used approach in statistical modeling that allows to measure how well do the observed data can be described by the fitted model. After fitting our sensor events to a mixture of von Mises distributions, a natural question would be is whether the sensor data could have actually be drawn from those distributions. In our project, we decided to use two well approaches in a field of circular statistics: graphical representation (P-P and Q-Q plots) and hypothesis testing (Watson $U^2$ statistics). In this section, we will continue use the 'Frontdoor'
sensor event type from the van Kasteren dataset, and we will illustrate an application of both goodness-of-fit types on both clusters.

4.3.1 Visual Assessment using P-P and Q-Q Plots

A Probability-Probability (P-P) plot is obtained by plotting the empirical cumulative distribution function (ECDF) for every single observation against the theoretical cumulative distribution function (TCDF), which is in our case a von Mises distribution function. On the other hand, a Quantile-Quantile (Q-Q) plot allows to compare the empirical quantile function against the von Mises quantile function. In other words, observed data samples are plotted against the quantiles that have been obtained after the distribution fitting. The data-model fitness is evaluated based on how plotted points are organized along the diagonal line, which connects the points (0,0) and (1,1) for a P-P plot, and (0,0) and (2\pi, 2\pi) points for a Q-Q plot [48].

To illustrate a working principle of a P-P and Q-Q plots, let us examine their application for the 'Frontdoor' sensor event type from the van Kasteren dataset. The Bayesian information criterion indicates that two von Mises distribution should be fitted, which is illustrated in Figure 4.23. Estimated model parameters can be found in Table 4.4.

![Figure 4.23: Clustering the 'Frontdoor' sensor events by fitting two von Mises distributions (van Kasteren dataset)](image)

Table 4.4: Estimated parameters for a mixture of von Mises distributions, 'Frontdoor', (van Kasteren dataset)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>( \mu ) (radii)</th>
<th>( \kappa )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>2.6</td>
<td>4.32</td>
<td>0.39</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>5.1</td>
<td>2.38</td>
<td>0.61</td>
</tr>
</tbody>
</table>

Having identified cluster memberships, we can now have a look at Figures 4.24 and 4.25, which describe the data-model fit of the Clusters 1 and 2 from the 'Frontdoor' sensor event, respectively.
Chapter 4. Implemented Framework

Figure 4.24: P-P (left) and Q-Q (right) plots for the Clusters 1 - 'Frontdoor' sensor event type

Figure 4.25: P-P (left) and Q-Q (right) plots for the Clusters 2 - 'Frontdoor' sensor event type

Plotting of graphs (Figures 4.24 and 4.25) is based on the estimated model parameters that could be found in Table 4.4. Basically, we are trying to check if the obtained mixture model parameters allow us to generate a data that follows a von Mises distribution. Despite the fact that most of the sensor events lie along the diagonals for both clusters, some deviation could be also noticed. Overall, it is possible to conclude that our mixture of von Mises distributions has sufficiently fitted our sensor data. However, visual assessment of the goodness-of-fit, and interpretation of these plots might be considered subjective (how big the deviation is from the diagonal line), and we need a formal method for our hypothesis testing.
4.3.2 Watson $U^2$ Statistics

Alternatively to the graphical representation of a goodness-of-fit, we can apply a commonly used hypothesis test on our circular data, namely Watson’s $U^2$ statistics \[74\]. This statistical approach is designed for a circle and location invariant, which basically allows not to be dependant on the starting point for an angle $\theta$.

**Definition 8.** Given a random sample $\theta_1, \theta_2, ..., \theta_n$ drawn from some population. In order to measure the discrepancy between the cumulative distribution function $F(\theta)$ and the empirical distribution function $F_n(\theta)$, Watson $U^2$ statistics for a sample size $n$ is presented as follows (by making some notational changes as described in \[50\]):

$$U^2 = n \int_0^{2\pi} \left[ F_n(\theta) - F(\theta) - \int_0^{2\pi} \{ F_n(\phi) - F(\phi) \} dF(\phi) \right]^2 dF(\theta) \quad (4.7)$$

The goodness-of-fit for a one von Mises distribution can be performed by replacing the $F(\theta)$ by:

$$F(\theta; \mu, \kappa) = \int_0^\theta f(\phi; \mu, \kappa) d\phi \quad (4.8)$$

In order to test the null hypothesis $H_0$, that the random sample $\theta_1, \theta_2, ..., \theta_n$ does come from the distribution $F(\theta)$, Lockhart and Stephens in their paper \[50\] suggest the following 4 steps to calculate $U^2$ statistics:

- estimate unknown parameters using maximum likelihood estimate approach
- calculate $z_i = F(\theta_i)$ for each $i$ in 1, 2, ..., $n$ and if necessary replace unknown parameters by their MLE
- put the $z_i$ in ascending order to obtain $z_1 < z_2 < ... < z_n$
- calculate the $U^2$ statistics as:

$$U^2 = \sum_{i=1}^n \left[ z_i - \left( \frac{2i - 1}{2n} \right) \right]^2 - n(\bar{z} - \frac{1}{2})^2 + \frac{1}{12n} \quad (4.9)$$

where $\bar{z} = \frac{\sum_{i=1}^n z_i}{n}$ is the sample average of the $z$ values.

The R package 'circular' \[1\] allows to implement this version of the hypothesis test with the help of a `watson.test` function (when argument `dist` equals to ‘vonmises’), the $H_0$ is 'Data follows the von Mises distribution'. In our project, we apply this function on every single cluster that has been obtained after fitting the sensor data to the mixture of von Mises distributions.

Obtained results allow to quantitatively assess how well every single von Mises distribution (i.e. cluster) represents the underlying sensor data. Table 4.5 below illustrates the estimated $U^2$ statistics and p-value for both clusters of the ‘Frontdoor’ sensor event.

<table>
<thead>
<tr>
<th>Cluster</th>
<th>$U^2$</th>
<th>P-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>0.0523</td>
<td>&gt; 0.10</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>0.0845</td>
<td>&gt; 0.10</td>
</tr>
</tbody>
</table>

As can be seen from the Table 4.5, $U^2$ statistic values of 0.0523 (Cluster 1) and 0.0845 (Cluster 2) are way below the critical value of 0.164 (Lockhart and Stephens \[50\], Table 1) for the level 0.01, which does not allow to reject the null hypothesis of following the von Mises distribution.
Chapter 5

Case Studies

To test the validity of our label refinement framework, two case studies have been performed by us. In order to verify the reliability of our approach, for both case studies real-life smart-home event log data has been examined and results are presented in the following sections. The development of the label refinement framework has been performed in R Studio, and the verification of this method within the process discovery domain has been achieved with the help of ProM6 [72] toolkit.

In Section 5.1 we will introduce smart-home data sources that contain publicly available dataset of Tim van Kasteren [70] and private dataset that has been provided by Philips Research. We will continue with Section 5.2, where we first discuss the smart-home event log pre-processing as a preparation step for the further analysis. Next, we present the Petri net mined from the unrefined event log in order to illustrate the existing limitations, followed by the discussion of our label refinement method applied on one of the sensor types. We conclude this section by presenting the Petri net mined from the refined event log, and comparing our approach to one of the existing methods within the process mining research discipline. Section 5.3 follows the same discussion procedure, however covering more sophisticated smart-home event log data.

5.1 Datasets Used

In this section, the data format of two datasets (i.e. van Kasteren and Philips) will be discussed by indicating existing commonalities and discrepancies that should be taken into account before applying the label refinement framework.

5.1.1 van Kasteren Dataset

This dataset contains the recorded sensor event information about a 26-year old male inhabitant for the period of 28 days. For this time duration 2570 sensor events and 245 agent annotated activities have been collected. Annotation of activities has been performed with the help of Bluetooth headset device using the predefined set of commands and rely upon the well-known Katz Activities of Daily Living (ADL) index [31]. As in our project we are concentrated on the unsupervised learning, we will not consider annotated activity level of Kasteren dataset (as a good ground truth for supervised learning) and will focus on sensor event level. Sensor network formed out of 14 different state-change sensors (their ID’s indicated in parentheses); 'Microwave’ (1), 'Hall-Toilet door’ (5), 'Hall-Bathroom door’ (6), ‘Cups cupboard’ (7), 'Fridge’ (8), 'Plates cupboard' (9), 'Front-door’ (12), 'Dishwasher’ (13), 'Toilet-Flush’ (14), 'Freezer’ (17), 'Pans Cupboard’ (18),...
'Washing-machine' (20), 'Groceries Cupboard' (23), 'Hall-Bedroom door' (24). Every event represents the state change of a particular sensor type. Table 5.1 presents a small sub-set of Kasteren dataset.

Table 5.1: Sample sensor events, van Kasteren dataset

<table>
<thead>
<tr>
<th>Start Time</th>
<th>End Time</th>
<th>Sensor ID</th>
<th>Value (always 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25-Feb-2008 00:20:14</td>
<td>25-Feb-2008 00:22:57</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>25-Feb-2008 09:37:58</td>
<td>25-Feb-2008 09:38:01</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>25-Feb-2008 09:37:51</td>
<td>25-Feb-2008 09:37:52</td>
<td>14</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

5.1.2 Philips Dataset

In comparison to van Kasteren dataset, Philips dataset is more comprehensive from the perspective of a number of available event attributes, sensor types and observation period (almost 3 month). Table 5.2 illustrates small subset of this particular dataset.

Table 5.2: Sample sensor events, Philips dataset

<table>
<thead>
<tr>
<th>SensorID</th>
<th>SensorType</th>
<th>SpaceID</th>
<th>ApplianceID</th>
<th>Key</th>
<th>Value</th>
<th>Timestamp</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>PowerSensor</td>
<td>Livingroom</td>
<td>TV</td>
<td>Power</td>
<td>82</td>
<td>02-18-2015 15:03:42</td>
</tr>
<tr>
<td>10</td>
<td>PowerSensor</td>
<td>Livingroom</td>
<td>TV</td>
<td>Power</td>
<td>80</td>
<td>02-18-2015 15:04:02</td>
</tr>
<tr>
<td>34</td>
<td>MotionSensor</td>
<td>Livingroom</td>
<td>Movement</td>
<td>Motion</td>
<td>1</td>
<td>02-18-2015 15:04:17</td>
</tr>
<tr>
<td>31</td>
<td>MotionSensor</td>
<td>Hallway</td>
<td>Movement</td>
<td>Motion</td>
<td>1</td>
<td>02-18-2015 15:04:21</td>
</tr>
<tr>
<td>20</td>
<td>OpenCloseSensor</td>
<td>Hallway</td>
<td>Frontdoor</td>
<td>Open</td>
<td>1</td>
<td>02-18-2015 15:04:25</td>
</tr>
<tr>
<td>34</td>
<td>MotionSensor</td>
<td>Livingroom</td>
<td>Movement</td>
<td>Motion</td>
<td>0</td>
<td>02-18-2015 15:04:33</td>
</tr>
<tr>
<td>20</td>
<td>OpenCloseSensor</td>
<td>Hallway</td>
<td>Frontdoor</td>
<td>Open</td>
<td>0</td>
<td>02-18-2015 15:04:36</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

This dataset contains observations of 20 different elderly people (stored in 20 different files) with different diagnosis, such as memory problem or dementia. Sensor types are; 'CO2Sensor', 'HumiditySensor', 'MotionSensor', 'OpenCloseSensor', 'PowerSensor', 'PressureSensor', 'ProximitySensor', and 'TemperatureSensor'. In comparison to the van Kasteren dataset, Philips dataset contains sensor types which do not only operate based on state (0 or 1). Sensor types, such as 'CO2Sensor' or 'HumiditySensor' measure the corresponding values (CO$_2$ concentration and humidity) within the given measure range. Moreover, sensor events of type 'PowerSensor' indicate the periodic power consumption throughout the working stage. In addition, sensors of type 'OpenCloseSensor' or 'PressureSensor' record the data based on state (1 - open, 0 - close). A detailed representation of sensors and their types could be found in Table 5.3

In our project, we will not utilize sensors of type 'CO2Sensor', 'HumiditySensor', and 'TemperatureSensor' due to their different data nature (i.e. they are not directly affected by human behavior). Indeed, one might argue that opening a window (which is performed by human) could affect the temperature, humidity or carbon-dioxide concentration within the room. However, this argument cannot be checked, thus we consider to avoid this kind of vague assumptions to be logical. Instead, we would concentrate on state-based sensors, such as 'Frontdoor' or 'Bed', which directly indicate human behavior within the smart-home environment. In addition, due to the continuous value nature of sensors, such as 'TV' (value range: 0 - 157), 'Microwave' (value range: 1 - 1340), and 'Water Cooker' (value range: 0 - 1885), they will be converted to the state-based
sensor types by identifying their ‘ON’ and ‘OFF’ ranges. The importance of this transformation is illustrated by Figure 5.1

![Figure 5.1: Sample snapshot, Philips Dataset](image)

As can be seen from the figure above, ‘TV’ sensor event is happening continuously, every 10-20 seconds. By considering the fact that in our project, we are going to cluster starting times of sensor events (e.g., we would like to know when TV has been turned on/off) this kind of continuous timestamp update might significantly affect the validity of our method. Thus, we would like to transform our data in a form where it would be possible to track down the state change of a particular sensor within a smart-home. In this particular case, final data format should contain starting and ending times of ‘TV’ sensor and not its intermediate power consumption updates. After describing both datasets in a more detailed form, in the following Sections 5.2 and 5.3 we will present two conducted case studies, in order to verify and validate our label refinement approach using the von Mises Mixture Modeling.
5.2 Case 1 - van Kasteren Dataset

This section provides a detailed explanation of the suggested label refinement method for the analysis of the van Kasteren dataset. Major focus is made on the process discovery results before and after the refinement of particular sensor events.

5.2.1 Event log pre-processing

Existing smart-home event log should first be transformed into an appropriate data format for the application of process discovery algorithms. Within the process mining research discipline, the eXtensible Event Stream [71] (XES)\(^1\) format is a standard solution for this purpose, which is an event logging format based on the Extensible Markup Language (XML)\(^2\). Transformation of the original smart-home event log long into XES formatted event log is illustrated by Figure 5.2 and achieved with the help of ProM6 plugin ‘Parse van Kasteren Data Set’.

![Figure 5.2: Transformation of raw smart-home event log (left) into XES format (right) for process discovery](image)

After transforming raw smart-home event log into the appropriate data format, additional steps should be performed before the application of process discovery algorithm. These steps are executed using several ProM6 plugins and applied to the XES event log successively: first, ‘Resort Log Based on Time’ should be performed to order events according to their timestamp, followed by ‘Apply Day Case Notion’\(^3\) that will help to generate process cases (i.e. traces) based on day component of the timestamp attribute, and finalized by ‘Filter Events’ plugin to leave only

\(^1\)http://www.xes-standard.org/
\(^2\)https://www.w3.org/XML/
\(^3\)https://svn.win.tue.nl/repos/prom/Packages/NiekTax/Trunk/
complete states of sensor events (represents lifecycle transition attribute of a particular event). These transformation steps generate an unrefined (i.e. original) sensor event log that represents a human behavior within a smart-home environment, and now ready for the application of one of the available process discovery algorithms. Obtained result is shown in Figure 5.3.

5.2.2 Petri Net Mined from the Unrefined Event Log

In order to mine a process model from the unrefined event log, we apply the Infrequent variant of Inductive Miner with a noise threshold 0.2 (i.e. 20% of infrequent log behavior will be filtered out) available within the ProM6 toolkit. Obtained result is shown in Figure 5.4, where big rectangular transitions are associated with an occurred sensor event, and small rectangular transitions (i.e. invisible transitions) are important from the routing perspective and their executions is not reflected in the original sensor event log. The discovered Petri Net is a traditional example of the 'flower-like' process model, which allows for much more behaviour that can be observed from the original event log. In this particular case, execution of sensors ['1' ('Microwave'), '17' ('Freezer'), '18' ('Pans Cupboard'), '7' ('Cups Cupboard'), '8' ('Fridge'), '9' ('Plates Cupboard')], and sensors ['5' ('Toilet Door'), '6' ('Bathroom Door'), '12' ('Frontdoor'), '24' ('Bedroom Door'), '14' ('Toilet Flush')] are allowed to happen in any order and for an infinite number of times, because of the backloop in the Petri net. This kind of process models are considered to be imprecise, as they struggle to restrict a behavioural patterns that can never be observed from the original event log. From the perspective of human behavior, it is hard to understand the ordering of the sensor events, which does not allow to obtain any meaningful insights. In order to tackle this problem, let us now apply our label refinement method on starting-time (expressed in a form of hour-float) attribute of several sensor types, and evaluate what kind of structural and statistical changes have been introduced in comparison to the original process model (Figure 5.4).
Figure 5.4: Petri Net Discovered from the Unrefined Event Log (van Kasteren event log)
5.2.3 Application of the Label Refinement Method

Among the 14 different possible sensor types, let us start by applying our label refinement method to the '24' ('Hall-Bedroom door') sensor, and our intention is to fit two von Mises distributions as has been suggested by Figures 5.5 and 5.6.

![Image of BIC change for the increasing number of von Mises mixture components - 'Hall-Bedroom door' (van Kasteren dataset)](image)

Figure 5.5: BIC change for the increasing number of von Mises mixture components - 'Hall-Bedroom door' (van Kasteren dataset)

```
set.seed(111)
bic.mixvmf(wait, A = 0)

1  2  3  4  5  6  7  8  9
357.1792 314.5310 402.1478 383.4238 347.0940 338.8200 338.3385 345.7131 397.7003
```

Figure 5.6: BIC values for the number of components from 1 to 9 - 'Hall-Bedroom door' (van Kasteren dataset)

Having identified a number of possible von Mises components, we can now fit the 'Hall-Bedroom door' circular data to these mixture components. Figure 5.7 illustrates the obtained result.

![Image of clustering the 'Hall-Bedroom door' sensor events by fitting two von Mises distributions (van Kasteren dataset)](image)

Figure 5.7: Clustering the 'Hall-Bedroom door' sensor events by fitting two von Mises distributions (van Kasteren dataset)
Convergence of the EM algorithm has been achieved in 22 iterations with a final log-likelihood equal to 53.63. Estimated model parameters are shown in Table 5.4, where $\alpha$, $\mu$ and $\kappa$ represent weight, mean direction in radii, and concentration around the mean for both clusters, respectively.

Table 5.4: Estimated parameters for a mixture of von Mises distributions, sensor type ‘Hall-Bedroom door’, (van Kasteren dataset)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>$\alpha$</th>
<th>$\mu$ (radii)</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>0.76</td>
<td>2.05</td>
<td>3.85</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>0.24</td>
<td>5.94</td>
<td>1.56</td>
</tr>
</tbody>
</table>

Based on the estimated parameters, Cluster 1 (red) covers the 76% of the sensor event data having mean direction value of 2.05 (117.28 in angles, 7.82 in hour-float) and a relatively dense concentration value of 3.85 around the mean. In contrary, Cluster 2 (blue) covers only 24% of the whole population, and has a mean direction value of 5.94 (340.15 in angles, 22.67 in hour-float) and a concentration value of 1.56. Low $\kappa$ value indicates that dispersion of sensor events for the Cluster 2 is higher in comparison to the Cluster 1.

Analysis of both clusters for the ‘Hall-Bedroom door’ sensor type (which accounts for 116 observations within the van Kasteren dataset) showed that Cluster 1 (red) contains 88 observations and covers events with occurrence time from 3.08 till 10.44 (expressed in hour-floats) and Cluster 2 (blue) contains 28 observations and covers events with occurrence time from 17.06 till 0.88 (expressed in hour-floats). These two clusters are utilized by us in order to generate label refinement candidates, namely ‘Hall-Bedroom door 1’ (Cluster 1) and ‘Hall-Bedroom door 2’ (Cluster 2). Obtained time intervals are used to refine every single occurrence of ‘24’ (‘Hall-Bedroom door’) within [3.08 - 10.44) and [17.06 - 0.88) into ‘24_1’ (‘Hall-Bedroom door_1’) and ‘24_2’ (‘Hall-Bedroom door_2’), respectively. This is achieved with the help of ‘Manual Daytime Attribute-based Refinement’ plugin developed by Tax et al. [59], available within the ProM6 toolkit. Figure 5.8 illustrates a small trace sample from the van Kasteren dataset before and after the refinement of sensor ‘24’ (‘Hall-Bedroom door’) has been performed.

![Figure 5.8: Transformation of sensor events '24' ('Hall-Bedroom door') after applying label refinement method](image.png)
CHAPTER 5. CASE STUDIES

Taken snapshot represents a sequence of occurred sensor events grouped in a trace and ordered according to their timestamps. Application of our label refinement method substitutes every single occurrence of the sensor '24' into '24_1' and '24_2', as has been suggested by the obtained clusters.

5.2.4 Petri Net Mined from the Refined Event Log

Having refined our original event log, we now apply the Infrequent variant of the Inductive Miner with a noise threshold 0.2 to newly generated refined event log, and check what kind of structural changes have been introduced (Figure 5.9). Red rectangle indicates the region where the 'structuredness' of the process model has been improved.

In comparison to the process model illustrated by Figure 5.4, where it is possible to execute sensor events of type '24' ('Hall-Bedroom door') and '12' ('Frontdoor') in parallel, newly obtained process model (from the refined log for the sensor type '24') does not allow this. As a result, we obtain a more restricted (i.e. precise) behavior from the event log, and this is one of the major objectives that we were striving for in this project. Generated process model shows that the '12' sensor event type, should happen only after the '24_1' refined candidate of the '24' sensor event type. By considering the fact that the label '24_1' is assigned to all sensor events of type '24', happening within [3.08 - 10.44] interval (i.e. morning period), interpretation of this label refinement also fits to our common intuition. According to this intuition, a front-door event happening after a bedroom door event indicates that a smart-home agent is awake and about to leave the house. On the other hand, the '24_2' refined candidate covers the evening range of events of the '24' sensor event type (i.e. [17.06 - 0.88]) and could be followed by many other sensor event types within the smart-home, and never be directly followed by the sensor type '12'.

Besides being able to visually assess the usefulness of our label refinement method, evaluation based on the statistical test introduced in Section 2.1 has been also performed. Obtained results show that for the performed label refinement the relative information gain is equal to 2.13%, which indicates the decreased entropy that helps to discover more structured process model. This number might not seem high, however following the explanation by authors of [59], we should state that the relative information gain is calculated over all log statistics, and might be large for a particular type of log statistic. Table 5.5 illustrates the calculated log statistics (for 3 types of ordering relation) for sensor event type '24' before and after refinement in respect to sensor event type '12'. Number relation in parentheses indicate the occurrence frequency, such as for the 'Eventually Follows' ordering relation, within the original event log among the 116 observations of sensor type '24', only 36 have been eventually followed by the sensor type '12', resulting into entropy value of 103.65. In contrary, the refined event log which contains labels '24_1' and '24_2' allows to re-calculate the entropy per label candidate (66.23 and 27.06, respectively) resulting into the lower entropy value of 93.29. Explained procedure is conducted between sensor type '24' and every other sensor type within the original event log. Same procedure is then repeated for the refined event log, which now contains refined labels of the original sensor event type. The relative information gain is then calculated that indicates whether the generated label candidates help to improve the structure of the discovered process model.

Table 5.5: Change in ordering relation statistics of label '24' and its refined candidates '24_1' and '24_2' with respect to sensor event type '12'

<table>
<thead>
<tr>
<th>Relation</th>
<th>Label '24'</th>
<th>Label '24_1'</th>
<th>Label '24_2'</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eventually Precedes</td>
<td>81.28 (22/116)</td>
<td>71.39 (22/88)</td>
<td>25.36 (9/28)</td>
<td>96.75</td>
</tr>
<tr>
<td>Eventually Follows</td>
<td>103.65 (36/116)</td>
<td>66.23 (19/88)</td>
<td>27.06 (17/28)</td>
<td>93.29</td>
</tr>
<tr>
<td>Directly Follows</td>
<td>14.57 (2/116)</td>
<td>13.77 (2/88)</td>
<td>0.0 (0/28)</td>
<td>13.77</td>
</tr>
</tbody>
</table>

58 Log-based Generation of Event Labels using Event Attributes
CHAPTER 5. CASE STUDIES

Figure 5.9: Petri Net Discovered from the Refined Event Log (Sensor '24')
CHAPTER 5. CASE STUDIES

Until now, we have discussed the application of our label refinement method on the sensor type '24'. The van Kasteren dataset contains 14 sensor types and Table 5.6 contains the relative information gain values after applying our label refinement method on some of these sensor types. Important to state that for each sensor type, we generate a new refined event log, where the original sensor type is substituted by its identified refined labels. The relative information gain is then calculated for every refined event log, and the Inductive Miner is applied on each of them in order to assess the structure of the process model. In such a way, it is possible to identify which label refinement candidates affect the quality of the original (i.e. 'flower-like') process model the most. For instance, from Table 5.6 one can easily derive that refining sensor event type '17' ('Freezer') into two allows for a highest information, whereas for other sensor types this number varies between 1.1% and 3.3%.

Table 5.6: The relative information gain for all refined sensor event types

<table>
<thead>
<tr>
<th>Sensor Type</th>
<th>Number of Candidates</th>
<th>Split Intervals</th>
<th>The Relative Information Gain (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>(5.5 - 13.8), (14.2 - 19.8), (20.0 - 4.0)</td>
<td>-8.2</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>(3.3 - 14.6), (15.0 - 3.3)</td>
<td>1.1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>(8.3 - 14.0), (15.0 - 2.0)</td>
<td>3.3</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>(8.0 - 14.1), (15.0 - 21.8), (22.9 - 3.0)</td>
<td>2.4</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>(8.0 - 14.0), (15.1 - 1.0)</td>
<td>2.8</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>(6.3 - 13.2), (15.3 - 2)</td>
<td>-12.0</td>
</tr>
<tr>
<td>17</td>
<td>2</td>
<td>(8.3 - 14.0), (16.0 - 1)</td>
<td>6.4</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>(8.0 - 14.5), (15.0 - 1)</td>
<td>1.1</td>
</tr>
<tr>
<td>24</td>
<td>2</td>
<td>(3.0 - 11.0), (17.0 - 1)</td>
<td>2.1</td>
</tr>
</tbody>
</table>

In addition, for two sensor types, '5' and '12', negative relative information gain has been obtained, which basically means that more entropy bits are required for encoding the refined event log in comparison to the original one. The negative relative information gain is a product of log statistics calculated for the 'Eventually Follows' and 'Eventually Precedes' ordering relations when obtained entropy value for refined labels is higher than the entropy of the original label in respect to other sensors. When difference for these two ordering relation types is high (for original log versus refined log), it affects whole log statistics resulting into negative relative information gain. For the 'Directly Follows' and 'Directly Precedes' ordering relations this is not the case and has not be observed in practice.

Figure 5.10 illustrates the process model discovered from the refined event log for sensor event type '12'. Refined labels '12_1' and '12_2' are responsible for starting times of 'Frontdoor' (i.e. '12') sensor type within the time intervals [6.3 - 13] and [15.3 - 2], respectively. Overall model structure appears to be more complicated in comparison to the original process model described by Figure 5.4. In this particular case, execution of sensors ['23' ('Groceries Cupboard'), '17' ('Freezer'), '14' ('Toilet Flush'), '7' ('Cups Cupboard'), '8' ('Fridge'), '9' ('Plates Cupboard')] happens without any restriction on ordering and number of executions. Refined labels '12_1' and '12_2' are less useful in improving the structuredness of the original process model and should not be considered as reliable as refined candidates '24_1' and '24_2' for sensor type '24'. To justify this statement, we can think of simplicity evaluation metric used for the assessment of discovered process models. In Section 2.1 we have already stated that if a simple model can explain the underlying event log better, or at least as good as a more complex model, selection should be directed towards the former one. From this perspective, despite that the refinement of sensor type '12' restricts certain behavior in comparison to the original event log, it is less helpful for the human activity analysis.
Figure 5.10: Petri Net Discovered from the Refined Event Log (Sensor '12')
5.2.5 Comparison to the Data-Aware Process Mining with Transition Guards

Before comparing our method to the data-aware process mining with transition guards, it would be appropriate to provide a small example that will help to show the limitation of the latter approach. Let us have a look on Figure 5.11, which describes a simple 'flower-like' model discovered from an arbitrary event log with four sensor types, and for the further examination consider sensor types $A$ and $B$.

![Image](image1.png)

**Figure 5.11:** Sample 'flower-like' imprecise model discovered from an event log with four sensor labels, A, B, C, and D

As can be seen from Figure 5.11, sensor types $A$ and $B$ can happen in any order by utilizing the initial token from place $p_1$. This is not a desirable representation of a human behavior with the help of process model, by considering the fact that we do not know the ordering relation of events. Let us also assume that there exist particular ordering relations between these two sensor triggers. Such as, it might be the case that in the morning $A$ happens before $B$, during the afternoon there is a possibility that either $A$ or $B$ might happen, and during the evening $A$ happens after $B$. One of the major goals of the process mining is to discover these ordering relations between events, and the identification of transition guards does not allow to progress in that when applied on the original smart-home event log. Original event log might contain events that relate to different human activities, however labeled the same (e.g. triggering 'Bed' sensor might indicate both 'Tossing' or 'Waking-up' activity types). Method that has been proposed by Massimiliano de Leoni and Wil M. P. van der Aalst in their work [14], is not capable of differentiating between this kind of abstraction levels, and mines a 'flower-like' model if the event log does not contain clear and fine-grained event labels. Thus, pre-processing of the original event log with the help of our label refinement method is required, which might help to find ordering relations described in Figure 5.12.

![Image](image2.png)

**Figure 5.12:** Improved model discovered from a refined event log for two sensor types, A and B

In this particular case, the refinement of sensor labels $A$ and $B$ into $[A_m, A_a, A_e]$ and $[B_m, B_a, B_e]$, respectively (based on the previously made assumptions of event occurrence during the day) allows to explore the ordering relations with the help of more precise process model (clouds indicate some other sensor activity happening).
We have briefly discussed the methodology that has been developed by the authors, and its implementation is achieved by applying 'Discovery of the Process Data-Flow (Decision-Tree Miner') plugin from the ProM6 toolkit. This plugin requires two inputs to be provided: original smart-home event log and discovered Petri net that has been mined by applying one of the process discovery algorithms on the original smart-home event log (for the consistency purpose of this work, we apply the Infrequent variant of the Inductive Miner with 0.2 noise threshold). In addition, we have to specify the event attribute that we would like to operate on in order to identify possible decision points for transition guards. We have used the *hour-float* attribute that indicates the starting times of events, as this a type of attribute utilized within the label refinement framework. Obtained process model is shown in Figure 5.14.

As can be seen from Figure 5.14, proposed approach helps to locate a particular transition guards, which represent decision points for a given sensor types. Despite the fact that this transition guards indicate which path should be taken based on the attribute value of hour-float, it does not allow to improve the original process model (for instance, sensor event types '12' and '24' are still happening in parallel). Major reason behind this, is that during the initial stage of the proposed methodology (i.e. discovery of Petri Net from a given event log), we already discover an imprecise (i.e. 'flower-like') process model based on the pure original event log. Application of this method on the unprocessed smart-home event log once again emphasizes the necessity to resolve the problem of imprecise labels at the event log level.

Application of the very same approach at our refined event log (i.e. sensor label '24' into '24_1' and '24_2') and mined from it process model results into different decision tree model discovery illustrated by Figure 5.15. We can observe that the discovered decision tree reflects the introduced improvement as described by Figure 5.9, where sensor event type '12' ('Frontdoor') can only happen after the refined candidate '24_1'. In addition, data-aware process discovery approach introduces a transition guard (Figure 5.13) that indicates the possibility to execute sensor of type '12' only if hour-float attribute is > than 8.72. This decision point indicates the fact that our smart-home agent uses the front-door after the bedroom door that happens in the morning (i.e. '24_1' which is a label for the morning bedroom door events), which supports our intuition about the fact the agent is awake and about to leave the house. Evening events of bedroom door (i.e. labels '24_2') are never followed directly by sensor event type '12', which highly likely indicates that the agent is at home (possibly about to go to sleep).
CHAPTER 5. CASE STUDIES

Figure 5.14: Decision Tree Model Discovered from the Original Event Log and Mined Petri Net

Log-based Generation of Event Labels using Event Attributes
Figure 5.15: Decision Tree Model Discovered from the Refined Event Log and Mined Petri Net (Sensor '24')
CHAPTER 5. CASE STUDIES

5.3 Case 2 - Philips Dataset

Having examined the application of our label refinement method on the van Kasteren dataset, in this section we will explore the Philips dataset and present obtained results.

5.3.1 Event log pre-processing

For the Philips dataset conversion of raw smart-home event log data into XES format has been achieved with the help of RapidMiner\(^4\) extended with a RapidProM [42] plugin. Process which contains a workflow of specific operators available within the RapidMiner framework is illustrated in Appendix A. We have analyzed sensor events produced by the smart-home agent D19 for the observation period of 85 days. For consistency reason, as for the case of van Kasteren dataset, we utilized state-based (i.e. 1/0) sensors events triggered by human activity within the smart-home environment. For our experiment, 6 different sensor types (i.e. ’20’ (’Frontdoor’), ’21’ (’Cutlery’), ’22’ (’Refrigerator’), ’23’ (’Potpincabinet’), ’40’ (’Bed’), and ’50’ (’Key chain’)) have been utilized, which generated 5034 events grouped in 85 traces (i.e cases) based on the day component of a timestamp attribute.

5.3.2 Petri Net Mined from the Unrefined Event Log

To mine a process model from the unrefined event log, we apply the Infrequent variant of Inductive Miner with a noise threshold 0.2 following the procedure of Section 5.2. Obtained process model can be found in Figure 5.16.

\(^4\)https://rapidminer.com/
As can be seen, obtained model is imprecise allowing 6 sensor event types to happen in parallel and in any order. Following discussion will illustrate the application of our label refinement method on a particular sensor types, and its effect on the structure of a discovered process models.

5.3.3 Application of the Label Refinement Method

Our exploration of refined label candidates per sensor type and their effect on the relative information gain has identified the most significant label refinement by splitting ‘40’ (‘Bed’) sensor type into ‘40_1’, ‘40_2’ and ‘40_3’. We first examine the sensor type ’40’ (‘Bed’), which is a state-based sensor type and triggered when the smart-home agent interacts with bed. The number of observations that has been recorded for this particular sensor type is equal to 678. The Bayesian information criteria suggests to fit three von Mises distributions described in Figure 5.17.

Figure 5.17: BIC change for the increasing number of von Mises mixture components - ‘Bed’ (Philips dataset)

Having identified a number of possible von Mises components, we can now fit the ‘Bed’ circular data to this mixture model. Figure 5.18 illustrates the obtained result.

Figure 5.18: Clustering the ‘Bed’ sensor events by fitting three von Mises distributions (Philips dataset)
Convergence of the EM algorithm has been achieved in 48 iterations with a final log-likelihood equal to 434.80, and the estimated model parameters can be found in Table 5.7, where $\alpha$, $\mu$ and $\kappa$ represent weight, mean direction in radii, and concentration around the mean for all three clusters, respectively.

Table 5.7: Estimated parameters for a mixture of three von Mises distributions, sensor type 'Bed', (Philips dataset)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>$\alpha$</th>
<th>$\mu$ (radii)</th>
<th>$\kappa$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>0.18</td>
<td>0.85</td>
<td>9.98</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>0.36</td>
<td>6.12</td>
<td>8.44</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>0.46</td>
<td>2.09</td>
<td>8.96</td>
</tr>
</tbody>
</table>

Obtained results show that Cluster 1 (red) covers the 18% of the sensor event data having mean direction value of 0.85 (48.70 in angles, 3.24 in hour-float) and a relatively dense concentration value of 9.98 around the mean. In contrary, Cluster 2 (green) covers 36% of the whole population, and has a mean direction value of 6.12 (350.65 in angles, 23.37 in hour-float) and concentration value of 8.44, whereas Cluster 3 (blue) is responsible for 46% of sensor events with a mean direction value of 2.09 (119.74 in angles, 7.98 in hour-float) and a concentration indicator of 8.96. Obtained cluster members allow to estimate the upper and lower boundaries for time intervals used for splitting, and can be found in Table 5.8.

Table 5.8: Split boundaries for sensor type '40' (Philips dataset)

<table>
<thead>
<tr>
<th>Cluster</th>
<th>Number of Observation</th>
<th>Split Intervals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster 1</td>
<td>121</td>
<td>[1.6 - 5.35)</td>
</tr>
<tr>
<td>Cluster 2</td>
<td>251</td>
<td>[17.6 - 1.58)</td>
</tr>
<tr>
<td>Cluster 3</td>
<td>306</td>
<td>[5.39 - 14.9)</td>
</tr>
</tbody>
</table>

5.3.4 Petri Net Mined from the Refined Event Log

Time intervals that have been obtained using our label refinement method are used to refine every single occurrence of '40' ('Bed') within [1.6 - 5.35), [17.6 - 1.58), and [5.39 - 14.9) into '40_1', '40_3' and '40_2', respectively. Once again, this operation is performed with the help of 'Manual Daytime Attribute-based Refinement' plugin available from the ProM6 toolkit. Figure 5.19 illustrates the process model that has been discovered from the refined smart-home event log. The structure of the original process model (Figure 5.16) has been improved by considering the fact that sensor event type '23' ('Potpancabinet') can now only happen after the refined label candidate '40_2'.

By considering the fact that the refined label '40_2' is responsible for the interval [5.39 - 14.9), being followed by the sensor type '23' is highly likely an indication that the person is awake and about to prepare a meal. Same conclusion can not be made about the other two refined label candidates, which cover the night and evening occurrences of the 'Bed' sensor event type. Close examination of the refined event log shows that the relabeled candidates '40_1' and '40_3' are usually followed by the '40_2', and in some rare cases might be followed by other sensor types, such as '22' ('Refrigerator') or '50' ('Key chain'). This behavior is also logical by considering the possibility of performing certain activities within the kitchen or any other locations of a smart-home during the sleeping hours. The relative information gain is equal to 3.81%, which reflects the fact of lower entropy in the log statistics, and as a result increased predictability of a process model from the refined log for sensor type '40'.

Log-based Generation of Event Labels using Event Attributes
Figure 5.19: Petri Net Discovered from the Refined Event Log (Sensor '40')
CHAPTER 5. CASE STUDIES

As it was indicated for the van Kasteren dataset, the negative relative information gain might be observed for some sensor types after applying our label refinement method, which reflects the fact that more entropy bits are required for encoding the refined event log in comparison to the original one. Figure 5.20 illustrates the label refinement of sensor type '22' into three possible candidate splits (identified by our label refinement framework).

Figure 5.20: Petri Net Discovered from the Refined Event Log (Sensor '22')
As can be seen from the Figure 5.20, structural changes have been introduced in comparison to the original 'flower-like' process model illustrated by Figure 5.16. Label refinement candidates '22.1', '22.2', and '22.3' are responsible for the occurrence time of the '22' (i.e. 'Refrigerator') sensor type within the [6.9 - 13.96), [13.96 - 20.29), and [20.3 - 0.3) time intervals, respectively. Interestingly enough, despite that the relative information gain is equal to -11\%, a certain improvement in model structure can be still observed. For instance, sensor events of type '40' ('Bed') or '21' ('Cutlery') might only happen after the refined candidate '22.3' (responsible for the evening interval [20.3 - 0.3), which for instance indicates the fact that a particular smart-home agent usually goes to bed after the evening occurrence of a refrigerator sensor events in comparison to the morning or afternoon ones. Obtained results show that the negative relative information gain might be a clear indicator of increasing entropy for information encoding while comparing original and refined log. However, it is still possible to obtain a process model that restricts a certain behavior available within an original process model, considering an ordering relations between smart-home sensor events. Nonetheless, as we have discussed in Section 5.2, based on the simplicity evaluation criteria for discovered process models, selection of simpler models that explain the underlying sensor event log better is preferable.

Besides being able to evaluate the relative information gain using the 'Manual Daytime Attribute-based Refinement', we can also check how refined candidates vary from each other in terms of their ordering relation with other available sensor type events. Appendix B describes the comparison between refined labels '22.1', '22.2', and '22.3' in such a way that we can examine significant statistical differences with respect to particular sensor types. For instance, obtained labels '22.3' and '22.2' are statistically significantly different from each other by considering their 'Directly Precedes' and 'Eventually Precedes' relations with '40' ('Bed') (p-values: 7.79*10^{-16} and 1.09*10^{-11}, respectively). On the other hand, labels '22.3' and '22.1' are statistically significantly different from each other by evaluating their 'Eventually Follows' relation with '50' ('Key chain') (p-value: 7.58*10^{-18}) and 'Directly Precedes' relation with '40' ('Bed') (p-value: 3.13*10^{-10}). Calculation procedure of a p-value and its threshold estimation can be found in work by Tax et al. [59]. This statistical information allows us to evaluate whether the obtained refined candidates are different from each other behaviourally (i.e. based on the ordering relation with respect to other sensor event types).
Chapter 6

Conclusion

The study was set out to explore the autonomous generation of context-based event label refinements of life-logs within the process mining research discipline. Before introducing major findings and contributions of this work, it would be appropriate to recall our main research question that we have defined as follows:

**Research Question:** Given an unprocessed (i.e. original) smart-home sensor event log, we would like to develop an event label refinement method based on the clustering of starting and/or duration times of sensor event type instances around the day, which will help to obtain a more ‘fine-grained’ sensor event log for the application of process discovery algorithms.

Prior to investigating the problem in more detail, we have discussed the conventional activity recognition and pattern discovery approaches, and indicated the advantages that process mining research field may introduce in comparison to those traditional methodologies. Our discussion of process mining field has been enriched with an explanation of certain fundamental concepts, such as event log structure, working principle of existing process discovery algorithms, and classical quality metrics used for the assessment of discovered process models. We have then identified the concept of label refinement as being a crucial raw smart-home event log pre-processing step, which helps to tackle the problem of abstract sensor events that does not allow to mine a reliable and insightful process models of human behavior. In addition, we have discussed a new evaluation technique developed by researchers of TU/e and Philips Research that facilitates the evaluation of usefulness of our label refinement method without the need of computationally expensive process discovery.

The first contribution we made with this work was a conducted data analysis of a smart-home sensor events, and an identification of an appropriate unsupervised machine learning technique, which constitutes the backbone of our label refinement method. We have shown that starting and duration times of smart-home sensor events derived from their timestamp attribute may establish a proper feature domain for the application of unsupervised machine learning algorithms in order to identify repetitive patterns of human behavior in autonomous and personalized manner. Our selection of clustering technique using the probabilistic mixture modeling has been motivated by the underlying data distribution of starting and duration times of smart-home sensor events. This clustering approach is based on the assumption that every single data point (in our case expressed in form of starting and duration times) within a given population can be described by a parametrized probability density function from which it might be drawn. In this work we have shown the importance of selecting a correct density function, by starting from Gaussian mixture modeling for linear data (i.e. duration times of sensor events) and making a transition towards von Mises mixture modeling for circular data (i.e. starting times of sensor events). Further development of this project and the validation of obtained results have been performed for the...
Our major contribution is a developed and presented label refinement framework based on the circular feature domain, which incorporates a set of statistical inference methods required to perform a reliable sensor event analysis, and produces a set of possible refinement candidates per sensor event type. Implemented framework consists of three major stages; data-model pre-fitting, data-model fitting, and data-model post-fitting. During the data-model pre-fitting stage, a so-called cluster tendency assessment is performed, which helps to evaluate the underlying structure of a given sensor event types and their potential for being clustered in order to discover insightful characteristics of human behavior. The number of possible mixture components (i.e. clusters) for every single smart-home sensor event type is also identified at this stage with the help of the Bayesian information criteria. We continue with data-model fitting stage, during which the fitting of sensor data to the mixture of von Mises components is performed with the help of Expectation Maximization algorithm based on the maximum likelihood estimate. At this stage we obtain a necessary clusters (i.e. mixture components) that contain distributions of starting times of instances for a given sensor event type, and the number of these clusters define a possible set of refinement candidates. This procedure is performed for every reliable sensor event type, and the number of possible label refinement candidates can vary based on the obtained number of clusters. During the last stage of our implemented framework, namely data-model post-fitting the goodness-of-fit is performed, which allows to check whether the data could have been drawn from the fitted von Mises distributions.

In order to demonstrate the applicability of our label refinement framework, two case studies have been performed by using the smart-home event log data from two different sources. During the first case study, publicly available van Kasteren dataset has been utilized, and discovered process models from the original and refined event logs have been presented. The obtained results show that the application of our label refinement method on a particular sensor event types allows to discover more structured process models, which provide informative insights for the analysis of human behavior within the smart-home environment. In order to highlight the importance of the sensor event refinement at the event log level, our unsupervised approach has been compared to the data-aware process mining with transition guards method, which illustrated the limitations of the latter solution by considering the abstract sensor labels of the original (i.e unprocessed) smart-home event log. During the second case study, more sophisticated private Philips dataset has been utilized, following the same evaluation procedures in order to demonstrate the usefulness of our framework. Despite the increasing number of sensor events, we still could show that it is possible to obtain more meaningful process models by pre-processing original event log using our label refinement method. Finally, using the new label refinement evaluation method that has been proposed by researchers of TU/e and Philips Research, we have shown that for every single sensor event type, its refined label candidates may differ not only based on a timestamp, but also based on how significantly they maintain the ordering relations (see Section 2.1) in respect to other available sensor event types. This information is important for assessing the commonalities and discrepancies within a set of refined candidates (per sensor event type), and their effect of the structure of a discovered process model.

To conclude, in this project we have investigated and presented a label refinement method for pre-processing event logs obtained from smart-home environments that constitute a relatively new research domain for the process mining field. We believe that we have developed a reliable framework, which generates refined events logs that serve as a better data source format for traditional process discovery algorithms, and facilitates the mining of more reliable and insightful process models of human behavior.
CHAPTER 6. CONCLUSION

6.1 Limitations and Future Work

In this section, we would like to emphasize the observed limitations of our implemented framework, and indicate the possibilities for the future research within the label refinement study domain. Four major directions have been identified by us and presented below.

6.1.1 Concept Drift

First limitation of our label refinement method is related to the notion of concept drift. The concept drift can be caused by changes in human behaviour that usually occur throughout the life-time period. A restricted number of logged sensor events in real-life may be insufficient to detect two possible types of a concept drift; lifecycle (i.e. global) or seasonal (i.e. local) behavioural changes.

For the former type, the evaluated number of sensor events in both case studies that have been considered in this thesis, does not allow to track down a particular behavioural transitions throughout the life-time of a particular smart-home agent. Utilized event logs represent only a small portion of a possible smart-home agent behavior, and by considering the fact that human habits/preferences/life-styles might change due to certain circumstances (i.e. amelioration or deterioration of a particular illness), more observation are required in order to identify possible behavioural transitions.

The latter type of a concept drift is closely related to the former one, however reflects the seasonal changes in an overall human behavior. For instance, a particular human behavior (i.e. sleeping or cooking) during the winter period might be different from the one that can be observed throughout the summer period. Aggregated partial data may also lie at the transition point between these two seasonal periods that can introduce an alternative challenge for a reliable human behavior analysis. It would be interesting (and important) to apply our label refinement approach on a larger set of sensor event logs, and compare obtained process models with the ones that can be discovered from their sub-sets in order to assess the notion of concept drift.

6.1.2 Combination of Sensor Event Types

Second important aspect that should be investigated during the following studies is a set of possible sensor event types that can be used for the analysis of human behavior within the process mining domain. Considering that process discovery algorithms are sensitive towards the occurrence frequency of sensor events, mined process models may suffer from the selection of wrong sensor event type combinations. By considering a variety of possible sensor event types, and their occurrence frequency within an event log, a wrong combination of these sensors may result into misleading behavioural observations. For instance, during our second case study, we have noticed that the co-existence of a more frequent sensor event type '30' ('Bedroom') and less frequent sensor event type '40' ('Bed') within the same sensor network does not allow to explore the full capability of our label refinement method and illustrate it with the help of the discovered process model. Same issue can be related to the combination of sensor event type '32' ('Kitchen') and sensor event types '22' ('Refrigerator') or '23' ('Potpncabinet').

In our opinion, the major reason behind this problem is that we utilize sensor event logs that have been produced by sensors at different abstraction levels, such as occurrence of less abstract sensor event type '40' ('Bed') is not possible without a preceding and succeeding occurrence of more abstract sensor event type '30' ('Bedroom'). On the other hand, it is also not possible to trigger any kitchen appliance object (e.g. '22' ('Refrigerator')) without triggering more abstract sensor event type '32' ('Kitchen'). Considering the fact that within the Philips dataset the occurrence frequency of more abstract sensor event types significantly exceeds the number of less abstract
sensor event types, the application of our label refinement method on the latter ones may not reflect the introduced improvement for the human behaviour analysis. Taking into account that ordering relations discussed in Section 2.1 play a crucial role within the process discovery domain, 'noisy' coverage of more frequent sensor events may create an obstacle for the performance of the label refinement method.

### 6.1.3 Multidimensional Feature Domain

Third aspect of future work is related to the possibility of enriching the feature domain for the application of the clustering algorithm that has been introduced within this study. As it was indicated in Chapters 1 and 2, major focus in this project was directed towards the exploration of a timestamp attribute of sensor events. 1-dimensional feature domains (i.e. starting times and duration times of sensor events) have been explored by us that have proven to be a reliable sources for the generation of a label refinement framework. However, we believe that it is possible to enrich the initial feature domain that might create a new foundation for the analysis of possible refinement methods. In particular, features such as preceding and following events, combination of events that have been selected using a sliding/static window size may allow acquire more information with respect to human behavior analysis.

In our study, we have excluded sensors that measure humidity, temperature or \( \text{CO}_2 \) concentration due to the absence of a required domain knowledge (i.e. we can only guess about the human behavior with the help of these sensor outputs). However, future study may cover the possibility of incorporating measures from these sensor event types (first, pre-processing their output values) by considering that certain insights might be still derived from them (e.g. sharp decrease in a temperature level might be a sign of a window being opened or an air-conditioner being turned on). Enrichment of the initial feature domain is supported by the conceptual characteristics of the von Mises mixture modeling, which can be expanded from the application on data points that have been previously defined by a 1-dimensional circular space, and can now be identified by a multi-dimensional spherical space.

### 6.1.4 Iterative Refinement of Sensor Event Types

For our last improvement direction, we would like to emphasize the necessity to explore a particular refinement strategy that will help to apply our method iteratively on an original event log. In the current approach, we refine every possible smart-home sensor event type separately, while producing a new refined log for every single refinement. For instance, within our smart-home sensor network we might have 15 different sensor event types, and 10 of them might be identified as suitable candidates for the application of our label refinement method. In this particular case, 10 refined event logs will be produced, and the application of process discovery algorithms on each of them may reveal best refined sensor event types by comparing discovered process models. In addition, a new evaluation metric that has been introduced in Section 2.1.3 allows us to evaluate the relative information gain of a performed refinement operation.

However, we believe that it is important to come up with a solution that will help to proceed with refinements until no further improvement can be achieved, possibly using only one refined log, and improving it after every new iteration. One of the major challenges for this task is that one should consider an ordering of refinements, as a refinement of the best sensor label candidates may decrease their explanatory power over the process model if applied after not so promising label refinements and other way around.
Bibliography

[1] C. Agostinelli and U. Lund. *R package circular: Circular Statistics (version 0.4-7)*. CA: Department of Environmental Sciences, Informatics and Statistics, Ca’ Foscari University, Venice, Italy. UL: Department of Statistics, California Polytechnic State University, San Luis Obispo, California, USA, 2013. 29, 49


[19] Chris Fraley, Adrian E. Raftery, Thomas Brendan Murphy, and Luca Scrucca. mclust Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification, and Density Estimation, 2012. 26


[53] Parisa Rashidi, Diane J. Cook, Lawrence B. Holder, and Maureen Schmitter-Edgecombe. Discovering activities to recognize and track in a smart environment. IEEE Transactions on Knowledge and Data Engineering, 23(4):527–539, April 2011. 17
[57] Cosma Rohilla Shalizi. Advanced data analysis from an elementary point of view. Citeseer, 2013. 23
[65] Wil M. P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering, 16(9):1128–1142, September 2004. 8, 13


[73] G Vincent and V Velko. The next four decades - the older population in the united states: 2010-2050. 2010. 1


Log-based Generation of Event Labels using Event Attributes 81
Appendix A

Processing Philips Dataset

This appendix shows the RapidMiner work-flow operators used to process Philips dataset.

Figure A.1: Transformation of raw smart-home event log data into XES format using RapidMiner
Appendix B

Statistical Testing Results

Calculated p-values for the null hypothesis that indicates if refined labels \( X_i \) and \( X_j \) are equally likely to hold any log relation \( R \) (i.e. directly follows, directly precedes, eventually follows, and eventually precedes) with respect to some sensor event \( Y \).

Figure B.1: Statistical Testing Results among Refined Labels of Sensor ‘22’