

# Addressing Realisms in Activity Recognition for Smart Home Deployments

---

A Dissertation

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

---

In Partial Fulfillment

of the requirements for the Degree

Doctor of Philosophy (Computer Science)

by

Enamul Hoque

May 2014

## Approval Sheet

This dissertation is submitted in partial fulfillment of the requirements for the degree of  
Doctor of Philosophy (Computer Science)

Enamul Hoque

---

Enamul Hoque

This dissertation has been read and approved by the Examining Committee:

John A. Stankovic

---

John A. Stankovic, Advisor

Kamin Whitehouse

---

Kamin Whitehouse, Committee Chair

Alfred Weaver

---

Alfred Weaver

John Lach

---

John Lach, Minor Representative

Stephen Patek

---

Stephen Patek

Accepted for the School of Engineering and Applied Science:



---

James H. Aylor, Dean, School of Engineering and Applied Science

May 2014

# Abstract

Research in wireless sensor networks has been very successful in creating test beds and short-term deployments for many application areas (e.g, home health care, saving energy in buildings, security systems) that depend on accurate activity recognition. The utility of these activity recognition systems often depends on recognizing anomalies from typical behaviors learned based on the activities. However, for many real home situations these activity recognition and anomaly detection solutions are not robust enough due to many realities.

In this dissertation, we have designed, implemented and evaluated a novel activity recognition system named *AALO*, a comprehensive anomaly detection system in daily activities named *Holmes*, and a novel ground truth collection system named *Vocal-Diary* that can be used to evaluate both *AALO* and *Holmes*. *AALO* is an active learning based activity recognition system that applies machine learning and data mining techniques to address some of the realities of deployments including difficulty in obtaining labeled ground truth for training, individuals performing overlapping activities, and generalizability in varying smart home environments. *Holmes* is a comprehensive anomaly detection system for daily in-home activities that learns normal variability in daily activities based on specific days of the week, combine activity instances of a day / multiple days together to find the features that best represent regularity, and detect temporal and causal correlations among multiple activities. In addition, *Holmes* uses semantic rules learned based on resident and expert feedback that explain specific variations in daily activities in specific scenarios. Finally, *Vocal-Diary* is a

voice command based ground truth collection system where residents log activities by specific voice commands. To ensure robustness in the presence of different environmental noise in the home, *Vocal-Diary* integrates two-way acknowledgements and speaker recognition in the framework.

We evaluate the contributions with publicly available datasets and our own deployments. Results show that *Holmes* and *Vocal-Diary* performs better than state of the art systems and *Vocal-Diary* performs as good as the state of the art supervised activity recognition systems without requiring the large amount of ground truth that they need.

# Acknowledgements

**To my parents:** First and foremost, I would like to acknowledge my parents for always encouraging me to study hard and be competitive. They always ensured that I receive the best level of education. Before fulfilling any of their needs, they always made sure that my siblings and I are happy. I dedicate my thesis to them, the best parents in the world.

**To my other family members:** I would also like to acknowledge my siblings and other family members for always believing in my ability. Their love and care are precious.

**To my teachers in Bangladesh:** I would like to acknowledge all my teachers in Bangladesh who prepared me since my childhood. Teachers in Bangladesh work with limited resources. Still their dedication and hard work propel students like me to compete with the very best internationally.

**To my advisor, Jack:** I have learned so much from Jack. He always encourages to think of new ideas, pushes for excellence, values students' opinions, and corrects our mistakes patiently in a way that we would not make the same mistakes again. All the ideas in this thesis are a result of countless discussions with him. Above all, he always started our one-to-one meetings by asking whether I was doing fine or not. I was fortunate to get a visionary like him as my advisor and he will always be my role model.

**To my committee members, Kamin Whitehouse, John Lach, Stephen Patek, and Alfred Weaver:** I would like to thank the committee members of my dissertation for their valuable feedback in defining the scope of the work, and in formulating a proper evaluation

plan. They have been very responsive and helpful. I have definitely learned a lot from them over the course of the years.

**To University of Virginia:** I would like to thank University of Virginia for giving me the chance for graduate studies. Specially, the staffs and teachers in Computer Science department are very friendly and welcoming. I would also like to acknowledge fellow graduate students in our research lab, specially Rob who has introduced us to so many helpful tools and whose collaboration was very significant in my research. Overall, it was a pleasure studying in UVA.

**To my dear friends:** I am so lucky to be blessed with so many dear friends. Their role in my life is beyond words. I do not want to name anyone, they know who they are.

**To my wife, Tanima:** And last but not the least, I would like to acknowledge my wife. Tanima has made me a better person since the day I met her. During my graduate student life, there have been many ups and downs. She was with me at every moment to make this journey enjoyable. I am more confident with her by my side. I look forward to all the future chapters in life with her.

# Contents

<b>Contents</b>	<b>g</b>
List of Tables . . . . .	i
List of Figures . . . . .	j
<b>1 Introduction</b>	<b>1</b>
1.1 Thesis Statement . . . . .	6
1.2 Contributions . . . . .	7
1.3 Dissertation Organization . . . . .	9
<b>2 State of the Art</b>	<b>10</b>
2.1 Ground Truth Collection . . . . .	10
2.2 Activity Recognition . . . . .	12
2.3 Anomaly Detection . . . . .	14
2.4 Home Monitoring Systems . . . . .	16
<b>3 Data Collection</b>	<b>18</b>
3.1 Datasets . . . . .	18
3.2 Vocal-Diary . . . . .	23
3.3 Conclusions . . . . .	36
<b>4 AALO: Activity Recognition using Active Learning</b>	<b>37</b>
4.1 Motivation . . . . .	38
4.2 Contributions . . . . .	39
4.3 Framework For Training . . . . .	40
4.4 Activity Recognition . . . . .	48
4.5 Evaluation . . . . .	49
4.6 Conclusion . . . . .	69
<b>5 Semantic Anomaly Detection</b>	<b>70</b>
5.1 Motivation . . . . .	71
5.2 Contributions . . . . .	72
5.3 Types of Anomalies . . . . .	74
5.4 System Description . . . . .	78
5.5 Evaluation . . . . .	93
5.6 Conclusions . . . . .	109

<b>6</b>	<b>Conclusions</b>	<b>110</b>
6.1	Key Contributions towards Smart Home Deployments . . . . .	110
6.2	Future Improvements . . . . .	113
	<b>Appendices</b>	<b>115</b>
<b>A</b>	<b>Online Survey to Select Participants for Providing Daily Activity Details</b>	<b>116</b>
A.1	Pre-screening Survey Sent to Potential Participants . . . . .	116
A.2	General Consent Form Sent to Potential Participants . . . . .	120
<b>B</b>	<b>Form for Providing Daily Activity Details</b>	<b>123</b>
B.1	Google Form Sent out Daily for Two Months . . . . .	123
<b>C</b>	<b>IRB Approval to Deploy Sensors and Microphones</b>	<b>128</b>
C.1	General Consent Form Sent to Potential Participants . . . . .	128
	<b>Bibliography</b>	<b>132</b>

# List of Tables

3.1	List of Sensors in Our Deployment. . . . .	22
4.1	List of Sensors in the Kasteren Dataset. . . . .	50
4.2	Number of Clusters for Each Room in the Kasteren Dataset. . . . .	50
4.3	Instances per Activity in the Kasteren Dataset. . . . .	51
4.4	Set of Clusters for Kitchen for the Kasteren Dataset. . . . .	52
4.5	Confusion Matrix for all activities (percentage values) for the Kasteren dataset. The rows represent actual activities and columns represent the predicted activities. An entry in row $x$ and column $y$ represents percentage of time activity $x$ was recognized as activity $y$ . . . . .	56
4.6	List of Sensors in Our Deployment. . . . .	60
4.7	Number of Clusters for Each Room for the Data from Our Deployment. . . .	60
4.8	Set of Clusters for Living Room for the Data from Our Deployment. . . . .	61
4.9	Number of clusters for different activities for ‘Casas Dataset 1’. . . . .	66
5.1	List of models based on 3 months of training data for all activities . . . . .	96
5.2	A subset of frequent patterns generated from 3 months of training. If there is no other activity between the start and end of an activity, it is represented just by its name . . . . .	96
5.3	Average detected anomalies per month (rounded) across the 4-fold cross- validation for <i>BeClose</i> data set. <i>Holmes</i> generates the minimum number of alarms for each activity. . . . .	109

# List of Figures

1.1	End-to-End System for Smart Home Applications . . . . .	2
3.1	<i>Voice-Diary</i> End-to-End System. . . . .	27
3.2	Precision values for single-resident home 1. Adding speaker recognition increases the precision for all activities, and adding two-way acknowledgement increases it more. . . . .	32
3.3	Recall values for single-resident home 1. <i>Vocal-Diary</i> achieves 100% recall for all activities with the help of speaker recognition and two-way acknowledgement features. . . . .	33
3.4	Average precision values over all activities for three homes. Home 3 has two residents; Home 1 and 2 have single resident. . . . .	34
3.5	Average recall values over all activities for three homes. Home 3 has two residents; Home 1 and 2 have single resident. . . . .	35
4.1	Block Diagram of the Training Framework for textitAALO. . . . .	41
4.2	Using <i>AALO</i> for Activity Recognition . . . . .	48
4.3	Comparison of different instances of the cluster corresponding to ‘Sleep’ with the actual instances of ‘Sleep’ as recorded in the ground truth in the Kasteren dataset. . . . .	53
4.4	Training time slice error for each activity in the Kasteren dataset. . . . .	54
4.5	Training activity instance error for each activity in the Kasteren dataset. . .	55
4.6	Comparison of Time Slice Error for Activity Recognition among different classifiers for the Kasteren dataset. . . . .	57
4.7	Average time slice error for all activities in the Kasteren dataset over all the folds of leave-one-day-out cross-validation for <i>AALO</i> and semi-supervised algorithm with different bag sizes (i.e., different amount of labeled ground truth.)	58
4.8	Average time slice error for all activities in the data from our deployment over all the folds of cross-validation for normal clustering and for clustering with successive occupancy episode merging ( <i>AALO</i> ). . . . .	62
4.9	Average time slice error for all activities for the data from our deployment over all the folds of cross-validation for the data collected from our deployment. .	63
4.10	Number of clusters for different activities with varying training period for the data from our deployment. . . . .	64
4.11	Effect of retraining on training time slice error for different activities in the data from our deployment. . . . .	65

4.12	Comparison of time slice error of <i>AALO</i> with miSVM having different bag sizes for the data from our deployment. . . . .	66
4.13	Comparison of average time slice errors over all the folds of cross-validation of <i>AALO</i> with HMM and HSMM for different activities for ‘Casas Dataset 1’. .	67
4.14	Average time slice errors over all activities over all the folds of cross-validation for <i>AALO</i> and miSVM with different bag sizes for ‘Casas Dataset 1’. . . . .	68
5.1	Different types of anomalies that can be detected by <i>Holmes</i> . . . . .	75
5.2	<i>Holmes</i> Framework for Anomaly Detection . . . . .	80
5.3	Per-activity Context-aware Hierarchical Clustering . . . . .	83
5.4	Sample scatter plot of instances of a specific activity for 2 different cases showing need of context-aware clustering . . . . .	84
5.5	Average number of false positives generated by different algorithms across the cross-validation. <i>Holmes</i> generates the least number of false positives for every activity and reduces number of false positives by at least 46% . . . . .	97
5.6	Average number of false negatives generated by different algorithms across the cross-validation. <i>Holmes</i> generates the least number of false negatives for each activity and reduces number of false negatives by at least 27% . . . . .	99
5.7	Average precision values for different algorithms across the cross-validation for different activities. <i>Holmes</i> has the maximum precision for each activity and increases precision by at least 17% . . . . .	100
5.8	Average recall values for different algorithms across the cross-validation for different activities. <i>Holmes</i> has the maximum recall for each activity and increases recall by at least 6% . . . . .	101
5.9	Effect of <i>MERGE_THRESHOLD</i> on precision and recall values. We used the value 80%. . . . .	102
5.10	Comparison of precision values when <i>Holmes</i> is trained with actual activity labels collected by <i>Vocal-Diary</i> and when it is trained with activity labels produced by <i>AALO</i> . . . . .	103
5.11	Number of days each survey participant filled out the daily survey. . . . .	104
5.12	Total number of clusters for all activities with and without context-aware hierarchical clustering for the survey participants. . . . .	105
5.13	Average detected anomalies across all the folds of cross-validation for ‘Casas Dataset 1’. <i>Holmes</i> generates the least alarms for every activity. . . . .	106
5.14	Average detected anomalies per month (rounded) across the 4-fold cross-validation for ‘Casas Dataset 2’. <i>Holmes</i> generates the least alarms for every activity but showering. . . . .	107
5.15	Average detected anomalies per month (rounded) across the 4-fold cross-validation for ‘Casas Dataset 3’. <i>Holmes</i> generates the least number of alarms for every activity. . . . .	108

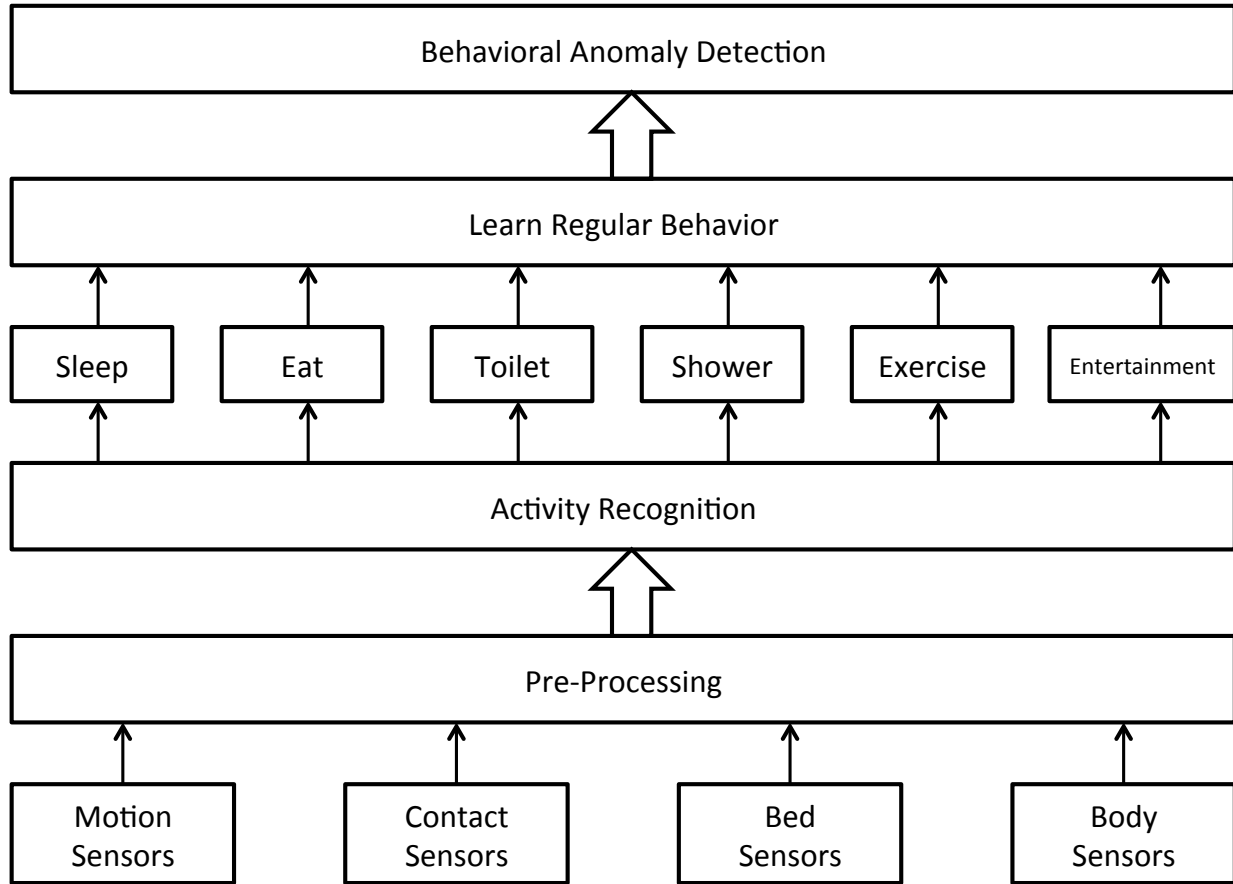


# Chapter 1

## Introduction

Due to the increasing number of elderly people living alone, wide-spread deployment of sensors has become prominent in home environments for monitoring daily activities and assessing behavioral changes. The ongoing emergence of the internet of things has facilitated making our homes smart. In such smart homes, living spaces and objects used for daily activities are instrumented with passive sensors. When a resident moves from one room to another or uses different objects that have attached sensors, a series of sensor firings with corresponding timestamps are generated that allow automatically detecting which activity the resident is currently performing, its duration and what objects are used for this activity. Home is where people spend most of their time in a day and these smart homes promise significant improvement in the quality of time spent in home in terms of comfort, safety, security, cost, and entertainment.

According to a recent analysis [1], the number of smart home installations is expected to reach more than 12 million by the end of 2015. A variety of applications run in these smart homes that include home healthcare, energy monitoring and control (e.g., HVAC, lighting), security, and safety [2, 3, 4]. All new homes will have some kind of sensing modalities installed as well as existing homes will be instrumented with state of the art sensors. It is expected that more and more innovative applications will be developed that will use these different



**Figure 1.1: End-to-End System for Smart Home Applications**

sensors in homes. Our vision is that smart homes will become the new hub for applications just as smart phones and smart cars are at present.

At the heart of all these applications exist the need of systems for accurate detection and summarization of daily activities, modeling regular behavior of the residents based on these daily activities, and long-term monitoring to detect deviation from regular behavior, i.e., anomalies. Due to the variability in home environments and difference in characteristics among different people, proper training is necessary for the systems listed above. The key to ensure that residents embrace different smart home applications, which clearly will improve their quality of life, is to keep the training process as less obtrusive for the residents as possible, and ensuring the reliability and robustness of the applications in home environments.

In this dissertation, we focus on building an end-to-end system for different applications

in single-resident smart homes. Figure 1.1 shows the end-to-end system to collect data from different sensors, infer daily activities of a resident based on the sensor firings, model a resident's regular behavior and monitor for any deviation in the learned behavior. The system in Figure 1.1 is a standard one for any home monitoring application. However, the main distinction of our approach from existing systems is in the way we ensure minimum user involvement in the training process, how we address different realistic scenarios in home environments, and how comprehensively we learn regular behavior in different contexts. Also, a necessity of an end-to-end system like the one shown in Figure 1.1 is ground truth for activities which we need to evaluate the performance of any new activity recognition system. We carefully address the challenges of ground truth collection that include ensuring comfort and privacy of residents, maintaining accuracy in ground truth, considering different real home scenarios that may cause errors in activity labeling, and addressing the reality that residents may forget to log activities.

The overarching novelty of this dissertation is that we address different challenges that may arise due to the usage of these applications in real homes. The challenges of smart home deployments that we address in this dissertation include obtaining labeled ground truth for training activity recognition systems, reducing the amount of user-labeled data necessary for training and retraining, solving the practical problems caused by overlapped activities in any clustering based activity recognition system, the inherent complexity in human behavior, and ensuring reliability of an anomaly detection system by reducing false positives and negatives. To address the challenge of obtaining labeled ground truth from residents, the activity recognition algorithms need to be designed so that they require as less labeled data for training as possible while maintaining acceptable accuracy. Also, comfortable, privacy-aware and robust ground truth collection systems need to be developed to collect whatever small amount of labeled data is necessary. In addition, activity recognition algorithms have to consider the fact that due to overlapped activities object uses for multiple activities may be interleaved. To address the challenge of complex human behavior, multiple models have to

be considered to model their regular behavior under different contexts, appropriate features for each activity have to be extracted, and temporal and causal correlations among different activities have to be learned. Existing systems in this domain do not address all these challenges in a holistic way (discussed below).

Activity recognition is at the center of an end-to-end system represented by Figure 1.1. Most existing activity recognition algorithms suffer from practical problems that may arise in real smart home deployments. Many ([5, 6, 7, 8, 9]) are based on supervised learning where classifiers are first trained with training data and then used for activity recognition. The training data needs ground truth i.e., accurate labeling of the activity a resident is performing at a particular time. To ensure high accuracy, the classifiers need to be trained with long traces of data that may range from months to years. However, collecting ground truth for such a long period is often not practical, very difficult and requires too much effort. Either the resident has to keep a record of all the activities or we need to use cameras and label each activity manually. The first approach is not comfortable for the resident; the second approach violates privacy and manual labeling from long traces of video data may not be practical.

There are some existing unsupervised activity recognition algorithms ([10, 11, 12]) that do not need ground truth. They require domain knowledge about activities and the environment (e.g., which objects are used for performing an activity). However, such systems cannot easily be generalized to wide variety of home environments and deployments. The other core components in Figure 1.1 are learning regular behaviors and detecting anomalies based on those. Existing anomaly detection systems suffer from generating numerous false positives and negatives that make them unreliable. The main reasons include treating each activity instance and day independently without considering the correlation among them ([13, 14, 15, 16, 17, 18]), and ignoring the dependence of behavior on different contexts. Another reason is the lack of semantic rules to filter out false positives as a logical deviation from normal behavior.

Existing ground truth collection systems suffer from different shortcomings such as use of cameras [19, 20] which is a privacy concern and very tedious, real-time logging by the user [6, 4] which is not suitable in real home settings for sufficiently long training duration, manual annotation of activity labels from sensor data based on some rules [21, 22] which is time-consuming and may not be always accurate, and necessity of wearing a microphone that may be uncomfortable for users [5, 8]. Also, none of the existing ground truth collection systems address the fact that residents may forget to log the begin and / or end of activities occasionally.

To address the challenges, first, we have designed a novel active learning based activity recognition system *AALO* that applies data mining techniques to cluster in-home sensor firings so that each cluster represents instances of the same activity. Users only need to label each cluster as an activity as opposed to labeling all instances of all activities; this significantly reduces user-labeled data for training. Once the clusters are associated to their corresponding activities, our system then uses this information to recognize future activities. *AALO* addresses one practical problem associated with such clustering based activity recognition approach: overlapped activities. This is accomplished by developing novel pre-processing techniques for sensor streams. Overall, our activity recognition techniques ensure minimal user involvement i.e., data labeling for re-training.

To evaluate *AALO*, we have developed a novel ground truth collection system named *Vocal-Diary* which is voice command based where residents log activities by specific voice commands. *Vocal-Diary* is privacy-aware, and robust to different environmental noise in the home and day-to-day conversations among the residents. To increase robustness, *Vocal-Diary* uses a speaker recognition system that is trained with the voice segments of all the residents in a home to ensure that the voice commands are spoken by the residents. Also, *Vocal-Diary* utilizes the sensor data produced by the underlying activity recognition system to query a resident periodically to check if he / she forgot to log any activity.

Based on the daily activities, we have developed *Holmes*, a comprehensive anomaly

detection system for daily in-home activities. *Holmes* learns a resident’s normal behavior in terms of in-home activities from training data based on *AALO*. Note that human behavior is complex. Therefore, a single per activity is not adequate in a real world to capture all the normal variations in behavior. Based on the training data, each activity may have multiple models as part of normal behavior (e.g., weekdays, weekends, Fridays). *Holmes* automatically learns these multiple models for each activity using a novel context-aware (i.e., day of week) hierarchical clustering algorithm. For example, there are a total of 15 – 35 models for all activities in our evaluation (details in Section 5.5). *Holmes* also learns temporal (e.g., if two activities often happen concurrently, if one activity often happens before another) and causal (e.g., if duration of one activity depends on another activity) relationships among multiple activities using sequential pattern mining and itemset mining algorithms. A highly accurate normal behavior assessment system forms the critical base upon which to detect anomalies.

*Holmes* is also designed to use semantic rules that define logical deviations from regular behavior. *Holmes* starts with an initial set of predefined rules defined from domain knowledge. As the system runs, newly detected anomalies are verified by the resident / experts to be included as new rules if appropriate. If there is repeated occurrence of one specific scenario (i.e., semantic rule), *Holmes* trains models for that particular scenario for future use; thus minimizing or even eliminating unnecessary false alarms.

## 1.1 Thesis Statement

By exploiting regularities in daily activities and semantic knowledge, our active learning based activity recognition algorithm performs on a par with existing supervised and semi-supervised activity recognition algorithms using less training data, and our semantic anomaly detection system produces less false positives and negatives in anomaly detection than purely data driven (statistical or machine learning based) anomaly detection systems.

## 1.2 Contributions

The main contributions of this thesis are as follows:

- A novel activity recognition system based on active learning named *AALO* that applies segmentation, itemset mining and clustering of low level sensor events during training, and automatically recognizes new room-level occupancy episodes as members of one of the clusters (i.e., activities) constructed during training. The novelty in the segmentation step is in the way *AALO* preprocesses raw sensor data by identifying overlapping activities across multiple occupancy episodes which is a concern for any clustering based activity recognition approach. The novelty in the itemset mining and the clustering steps is in the way *AALO* captures the temporal, spatial and object use regularities in the activities of daily living to represent them in terms of these regularities. Users only need to label each cluster as one activity after training which ensures feasibility of long term training.
- A comprehensive and multi-model system named *Holmes* for modeling regular resident behaviors carefully integrated with a sophisticated multi-level and semantic anomaly detection system. The novelty in modeling regular behavior includes the collection of following features: hierarchically merging clusters from different days of the week ensuring that the merged clusters do not become too generalized compared to the original ones, preservation of noise points found in training for future use during retraining, the use of a combination of features extracted from both individual activity instances and group of activity instances combined over a specific period, and use of sequential pattern mining and itemset mining algorithms to learn groups and sequences of activities that are part of a resident's regular behavior (i.e., temporal and causal correlations among activities). The novelty in the anomaly detection arises due to the combination of features that include: combining point, collective and context anomalies to ensure reliability, use of semantic rules that represent logical deviation from regular

behavior to reduce false alarms, and learning new semantic rules based on resident / expert feedback.

- A novel ground truth collection system named *Vocal-Diary* to collect accurate activity labels by listening to specific voice commands from residents with help of one / more microphone(s) in the home. The novelty of *Vocal-Diary* includes the use of two-way acknowledgement for listening to voice commands from a resident and integration of speaker identification in the pipeline for robustness. Another novel feature of *Vocal-Diary* is to query residents periodically to check if they forgot to log any activity by voice commands with the help of the sensor data produced by the underlying activity recognition system. The utility of *Vocal-Diary* is in evaluating the accuracy of *AALO* or training any other existing activity recognition system, and in using *Holmes* to model regular behavior and anomaly detection without deploying sensors in a home, i.e., only based on the activity labels collected by *Vocal-Diary*.
- Evaluation results based on: 1) six months of data collected from one single-resident home instrumented with sensors; 2) publicly available data from four single-resident homes that have one to six months of user-labeled in-home activity data along with data from corresponding in-home sensors; and 3) one data set from *BeClose* (a commercial senior safety system provider [23]) that contains four months of activity data from deployment of the *BeClose* system in a single-resident home.
- Evaluation results demonstrate that:
  - *Vocal-Diary* increases precision in accurate ground truth collection by at least 40% and recall by at least 10% compared to a system that uses voice command recognition without any acknowledgement and speaker recognition.
  - *AALO* performs as good as the state of the art supervised activity recognition algorithms (e.g., HSMM) even with significantly less amount of ground truth.

*AALO* also performs better than the multi-instance based semi-supervised activity recognition algorithm (with settings that are practical).

- *Holmes* reduces false positives in anomaly detection by at least 46% and false negatives by at least 27% compared to state of the art systems.

## 1.3 Dissertation Organization

The rest of the dissertation is organized as follows:

- Chapter 2 introduces the state of the art research related to ground truth collection, activity recognition, anomaly detection, and home monitoring systems.
- Chapter 3 details different data sets that we use to evaluate the technical contributions of this thesis. Then we present our novel ground truth collection system *Vocal-Diary* and its evaluation.
- Chapter 4 presents the details of different components of *AALO* which is our novel activity recognition system and its evaluation.
- Chapter 5 describes *Holmes*, a multi-model system for modeling regular resident behaviors carefully integrated with a sophisticated multi-level and semantic anomaly detection system. This is followed by evaluation of *Holmes*.
- Chapter 6 concludes this dissertation with a summary of the contributions, discusses the limitations of this work, and provides a number of possible directions for future work.

# Chapter 2

## State of the Art

In this chapter, first we discuss existing ground truth collection systems for activity labels and their limitations. Second, we describe state of the art activity recognition algorithms and argue why most are not suitable for long term deployments. Next, we present a summary of existing anomaly detection techniques and their shortcomings. Finally, we list the existing end-to-end systems for home monitoring.

### 2.1 Ground Truth Collection

A widely used method for collecting ground truth is using cameras and annotating activity episodes from the recording [11, 19, 20]. Cameras and computer vision algorithms are also used in [24] to help detect everyday actions such as stirring in a bowl or cooking. However, cameras suffers from privacy concerns, and it is very time consuming to go through the recordings to identify and annotate activities. Another common approach [22, 21, 25] is to go through the sensor firings during the training period and annotate activity labels based on some rules. This approach is also very time consuming and there is a magnitude of error which is not acceptable in ground truth. Authors in [6, 4] collect ground truth in real time as participants do different activities in a controlled lab setting. However, doing so in real deployments for a sufficiently long training period is impractical.

Logan et al. [26] use an experience sampling method to collect ground truth. In this method, periodic queries are sent to the smart phone of residents asking what activity they are doing. Authors in [27] use a combination of three methods that include experience sampling, a hand-written log where the resident records different activity labels at different times, and a collection of snap shots that the resident takes using the camera of smart phone. All these methods require the resident to actively interact with a smart phone / laptop / diary to log. Such high level interaction in a real home for a sufficiently long period is uncomfortable for residents.

We believe that the most comfortable way to collect ground truth is through voice commands. Interacting by voice has been proved acceptable in many applications (e.g., voice-based search, navigation, email). In the system developed by Kasteren et al. [5, 8], a resident wears a bluetooth headset and presses a button to give specific voice commands (implemented using Microsoft Speech API (SAPI [28])). However, this system has the discomfort that residents have to wear a headset before giving voice commands. Also, they may forget to wear the device or press the switch. From our experiments, we find that if we always keep the microphone on and / or if the microphone is far from the user, SAPI erroneously records many voice commands when the user did not speak. This is due to noise in the environment or other sounds (e.g., TV, day-to-day conversation). An ideal system should perform accurately even in such scenarios without a resident needing to turn on / off a switch. This is what *Vocal-Diary* accomplishes.

Also, none of the above systems address the issue that residents may occasionally forget to log start and / or end of activities. This may result in incomplete ground truth which can affect the training of activity recognition systems.

## 2.2 Activity Recognition

Many research groups [2, 29, 30, 31, 32, 33] have been investigating how to construct smart-living environments that target medical care to the individual. Emplaced wireless sensor networks (WSN) and body networks [34, 35, 36, 37, 38] are emerging technologies that promise to significantly enhance medical care for seniors living at home, in assisted living facilities, and in continuous care retirement communities (CCRCs). All of these require activity recognition.

Recognizing daily activities in complex home settings using in-home sensors is a well researched problem. Many existing solutions use simple sensors that detect movements of the resident from one room to another (i.e., motion sensors in the doorway) or changes in state of objects and devices (i.e., contact sensors). Kasteren et al. use temporal probabilistic models in [5] to recognize activities from sensor readings. They do not consider how long the resident has been performing an activity. Later in [8], the same authors use hidden semi markov models that consider duration of an activity to improve accuracy of activity recognition.

Tapia et al. apply a naive Bayes classifier for activity recognition in [7]. They propose to learn different time slice durations for different activities from the training data and use these durations for building models for different activities. Logan et al. use both naive Bayes and C4.5 decision tree models for activity recognition in [26]. Albinali et al. [39] apply Bayesian networks to detect activities with various optimization techniques that include eliminating redundant sensing data, bootstrapping to generate larger training sets and finding the best Bayesian network using a heuristic search.

Hu et al. [40] use skip-chain conditional random fields to model concurrent and interleaved activities. They do not need individuals performing concurrent or interleaved activities for training; they only need to perform each activity in isolation to train their system. Modayil et al. [41] model interleaved activities with interleaved hidden markov models. Their approach is also supervised and evaluation is done with controlled experiments. Kitani et al. [42] also recognize overlapped human activities; however, they need to collect ground truth with video

camera. Gu et al. [43] propose an emerging pattern based approach to recognize sequential, interleaved and concurrent activities. For training, they only need sequential activity trace. Helaoui et al. [44] markov logic networks to recognize concurrent and interleaved activities by combining training data and background knowledge.

One common problem associated with all the works discussed so far is that they require accurate labeling of activities (either by the resident or by manual annotation after viewing the data) during training which may be difficult to obtain for a long period (as discussed in [45]). Kasteren et al. present a technique in [46] to use the ground truth collected in one house to train activity recognition systems in other houses. However, details of activities may vary significantly from person to person and from home to home in which case this technique may not perform well. Our pre-processing techniques to detect overlapped activities and missing sensor events are novel considering the fact that we do not need ground truth for each activity which the state of the art algorithms listed above need.

By clustering sensor firings, Srinivasan et al. show in [47] that sensor firings have temporal and spatial regularity. Barger et al. present an unsupervised technique in [48] that clusters the sensor firings using mixture models. Each cluster has distinct time of occurrence, duration and rate of sensor firings. However, it does not consider the group of sensors being fired for clustering. Zheng et al. [9] also use clustering by a self-adaptive neural network to summarize the timing of sensor firings for each activity. Gu et al. [25] present an unsupervised approach for activity recognition based on object-use fingerprints to recognize daily activities without human labeling. This is done by first mining a set of object terms for each activity class from the web, and then mining contrast patterns among object terms based on emerging patterns. Similarly, Emmanuel et al. [11] extract activity models from text corpora such as the web and uses them to automatically produce labeled segmentations of activity data.

Philipose et al. present another similar approach in [10] to learn activity models from the web. However, the list of objects used for different activities may not be always extracted from web and mapping them to the actual deployed sensors is also complicated. Dimitrov et

al. [12] propose another unsupervised activity recognition approach that utilizes background domain knowledge about user activities and environment such as which objects are used for an activity. Unfortunately, such background knowledge may not be available. Wu et al. [19] recognize activities by learning object use sequence for different activities with the help of both web definitions and video of household activities. However, collecting video data has privacy concerns. Also, none of the unsupervised solutions mentioned so far address overlapping activities which may degrade the performance.

As a compromise between the supervised and unsupervised techniques, there have been previous works that require only a subset of training data to be labeled by users so that annotation effort is reduced. Stikic et al. [49] apply multi-instance learning for activity recognition from sparsely labeled data. Users are prompted after a predefined time interval (which is varied from 10 to 180 minutes) for labeling some activities. Therefore, users need to provide feedback multiple times in a day as opposed to our approach of labeling the clusters offline after training. Wu et al. [50] present a semi-automatic life log summarization system for elderly care. Similarly, Longstaff et al. [51] use active learning for activity recognition. However, both these systems require the users to always carry cell phones (having embedded sensors e.g., accelerometer, GPS, microphone) which may not be comfortable.

## 2.3 Anomaly Detection

Many current solutions use training data as a baseline and use statistical or clustering based anomaly detection approaches to find point anomalies. For example, Virone et al. [2] monitored 22 patients in an assisted living facility for two weeks that was treated as a baseline behavior. Then the baseline was used for the next six months to look for behavioral changes in their circadian rhythms. For changes of one order of magnitude a warning was signaled, and for changes of two orders of magnitude an alarm was signaled to a caregiver. In [52, 16, 53], authors learn which rooms the resident is in during different times of day and

monitor anomalies in room occupancy. However, circadian rhythms and room locations are very limited features to represent regular behavior.

Han et al. [13] use the mean of different features for different activities to define regular behavior and look for point anomalies based on predefined thresholds of deviation. Clustering based techniques are used in [14] to detect anomalies in timings and durations of different activities. All the above anomaly detection systems often suffer from generating numerous false positives that makes them unreliable. One reason for the false positives is treating each activity instance and day independently ignoring the correlation among them. Moreover, as they do not investigate the effect of day of the week on daily activities, this may also introduce a significant number of false alarms. In addition, they do not consider correlations among different activities.

There are existing systems that consider correlations among activities to detect collective anomalies. Anderson et al. [54] take an automata based approach to define sequence of activities as behaviors and learn those behaviors. They also support combining multiple days of activities to detect anomalies that occur over the time. However, they do not consider durations of each of the activities or the intervals among activities. These features are very useful for many health care applications. Jakkula et al. [17] use temporal mining to learn different temporal relations among different activities. In [18], authors use support vector machines to detect anomalies in sequences of activities. Authors use unsupervised pattern clustering techniques [55] to identify behavior model of the resident. However, none of these collective anomaly detection techniques considers differences in daily routines in different days of the week and specific features related to individual or group of activity instances which may cause both false positives and false negatives.

Authors present a survey of anomaly detection techniques in [56] in different domains and explains how context plays an important role in many domains. None of existing anomaly detection systems for in-home activities addresses the effect of context (e.g., day of the week, weather) on daily activities. *Holmes* is novel compared to existing systems in this respect.

Another shortcoming of the above techniques is the lack of semantic rules to filter out false positives as logical deviation from normal behavior. Raz et al. [57] propose anomaly detection using semantics in the field of software engineering to infer invariants about the normal behavior of dynamic data feeds. We argue that semantic rules are also important for anomaly detection systems for in-home activities.

## 2.4 Home Monitoring Systems

There are many existing home monitoring systems for specific applications that are summarized well in a survey paper [58]. Georgia Tech's *AwareHome* [59] combined context-aware and ubiquitous sensing, computer vision-based monitoring, and acoustic tracking of people to monitor health. The *Gator Tech* Smart House at the University of Florida was a laboratory-house created to assist older adults in maximizing their independence and maintaining a higher quality of life [60]. Harvard's CodeBlue[61] was designed to provide routing, naming, discovery, and security for various sensors including a portable 2-lead ECG, pulse oximeter, wearable *Pluto* mote with accelerometers, gyroscope, and electromyogram sensor for stroke monitoring. *AlarmNet* is an assisted living and residential monitoring system for pervasive and adaptive healthcare based on an extensible, heterogeneous network architecture targeting ad-hoc, wide-scale deployments [62]. Jiakang et al [63] propose a smart thermostat system by learning occupancy patterns of residents from motion sensors.

To date, there have been a few companies that have begun to sell their systems for home monitoring. There are many similarities in companies that provide home healthcare surveillance and for home security systems, and they often use similar sensors and network infrastructure. The WellAware [64] system provides commodity sensors to track sleep quality, activity levels, bathroom visits, and basic physiological information. *BeClose* [65] is another home monitoring system designed especially for the elderly. The system consists of a number of motion sensors as well as a bed pressure pad as well as a panic button that notifies authorities

and kin if there is something wrong. The user interface is built on the web platform and it presents a dashboard showing caregivers their patient's sleep patterns, movement, and weight. If the patient's behavior is anomalous, such as if they are not getting out of bed after a certain time or whether they are leaving the house too little or too much, a concerned relative can check on them.

Various large companies have also developed home monitoring systems for medical purposes which can help patients get proper medical help from home. PHILIPS provides Lifeline [66] with Auto Alert for elderly people. It is a help button that automatically places a call for help if it detects a fall. Intel-GE Care Innovations has developed Care Innovations QuietCare [67] that uses advanced motion sensor technology that learns the daily activity patterns of residents and sends alerts to help caregivers respond to potentially urgent situations and major routine changes. However, their system is limited as it only monitors the residents' room / zone level occupancy patterns. Cisco Expert on demand solution [68] improves patient care by using audio and video conferencing to support instant communication between clinicians, first responders, and other health experts. Cisco HealthPresence [69] can improve healthcare between patients, clinicians, and specialists located in distant places.

All of the above end-to-end systems for home monitoring are promising. However they do not consider the complexities in different daily activities (e.g., variation in home environments, overlapped activities) and the complexities in human behavior. We believe that addressing these challenges will enable these systems to fulfill all the benefits that they promise.

# Chapter 3

## Data Collection

In this chapter, we detail the different datasets that we use to evaluate the technical contributions of this thesis. The datasets include two publicly available datasets, one dataset from a senior safety system provider company named BeClose who deployed their system in an elderly person’s (living alone) home to collect data, data we collected by online survey for two months from 11 persons, and data we collected by deploying our end-to-end system in single-resident home for six months.

Also, we present our novel ground truth collection system *Vocal-Diary* and its evaluation in Section 3.2.

### 3.1 Datasets

#### 3.1.1 Public Datasets

We use data from two publicly available datasets for evaluation. Here we describe these two datasets.

### Kasteren Dataset

We use a publicly available dataset ([5]), in which a wireless sensor network was used in a single-resident home to observe the resident’s behavior and annotation was done by the resident using a bluetooth headset. The resident was a graduate student. It has 26 days of data and includes variety of sensors in different rooms. The dataset contains a total of 16 activities. They are: ‘Sleep’, ‘Eating’, ‘Prepare Breakfast’, ‘Prepare Dinner’, ‘Get Drink’, ‘Get Snack’, ‘Load Dishwasher’, ‘Unload Dishwasher’, ‘Load Washing machine’, ‘Unload Washing machine’, ‘Store Groceries’, ‘Use Toilet’, ‘Take Shower’, ‘Brush Teeth’, ‘Leave House’, and ‘Receive Guest’. Two of these activities (‘Eating’ and ‘Store Groceries’) have only one occurrence in the 26 days, so we do not consider these two activities. There are four rooms: *Bedroom*, *Kitchen*, *Toilet*, and *Shower*. We consider out of home as another room named *Outside* where the only sensor is the ‘Frontdoor’ sensor. All the sensors are binary sensors that include reed switches to measure open-close states of doors and cupboards, and float sensors to measure the toilet being flushed.

### Casas Dataset

We use data from three apartments from the publicly available CASAS smart home data set [22]. One of the apartments is instrumented with different sensors including motion sensors, door open / close sensors, and temperature sensors. We do not use the temperature sensors from the dataset as they just represent the temperature in the room and therefore do not depend on different activities. A single resident, who is a volunteer female adult, lived in the apartment, and we have six months of data from that apartment. Based on the sensor readings, the research team in [22] label activity episodes based on manual off line processing which we consider as ground truth for activity labels to evaluate both *AALO* and *Holmes*. We will refer to this dataset as ‘Casas Dataset 1’.

For the other two apartments, we only have daily activity data logged by the residents for four months. We will refer to these two datasets as ‘Casas Dataset 2’ and ‘Casas Dataset

3'. These datasets will be used to evaluate *Holmes*. The set of activities in each data set includes sleeping, meal preparation, working, watching television, showering, leaving and entering home, and dish washing, house cleaning and using the toilet. The data sets do not provide information on whether any of the activity instances is anomalous.

### 3.1.2 BeClose Data

BeClose is a commercially available home monitoring technology that promotes safety and wellness for aged and disabled individuals [23]. The BeClose technology platform consists of wireless, battery operated ambient sensors such as passive infrared motion sensors, door and cabinet sensors, bed and chair occupancy sensors, emergency alert buttons, and floor presence mats. The sensors communicate with a base station connected via cellular link (i.e. GSM) to a remote monitoring data center in the cloud. Such ambient sensor data is then contextualized to human behavior and activities via hierarchical clustering and heuristics based approaches. Activities include sleep, sedentary behavior, entries and exits, ambulatory activity, and kitchen use when available for the given resident. The BeClose system was deployed in the home of one individual over the age of 70. Data was collected and pre-processed over a period of four months. Data and activities were then retrieved from the BeClose cloud and incorporated into this study via a public API.

### 3.1.3 Data Collected by Online Survey

The ideal way to evaluate our end-to-end system is to deploy sensors in different homes, recognize daily activities from the sensor readings, and model daily behavior in terms of the daily activities to detect anomalies. However, it is difficult to get volunteers who would allow us to instrument their homes with sensors to test our system. To increase the variety in datasets for evaluation, we conduct online surveys to gather information about participants' daily life in terms of daily activities. The information includes the time of day when the participants do different activities of daily living. Although, these activity details are not

inferred from the sensors in their homes, they still allow us to evaluate our algorithms for modeling regular behavior.

Participants were selected on a volunteer basis based on pre-screening surveys. Initially, we sent out a pre-screening survey (Appendix A.1) by email to potential participants who included graduate students and friends (ages between 25-35). In this survey, we asked which activity details they are willing to share anonymously. The daily activities we listed are eating, sleeping, exercising, and entertainment. The length of the study was mentioned as two months. The participants had the freedom to skip a survey on any particular day. The participants remained completely anonymous, i.e., even we did not know which details are provided by which participant. Participants had the freedom to choose a code name for them, and each day, along with the details of activities they provided their self-selected code names so that we can group responses from the same user. We also sent out a copy of the general consent form they have to sign if they agree to participate in this study (Appendix A.2). This study was approved by the IRB review board of University of Virginia and it was mentioned in the general consent form.

For participating in the survey, they did not need to do anything out of their comfort zone. We had no influence on the potential participants that can force them to participate in the study. If they did not fill out the survey, no other additional contact was made to them. If they declined to participate in the study in the survey, no additional contact were made to them. If they agreed to participate in the two month study by selecting the set of activity details they were willing to share, a corresponding form was sent to them daily for two month starting from their preferred starting date. Appendix B.1 shows the google form that was automatically sent out to them daily in the morning for two months continuously. As we see from Appendix B.1, the questions asked daily included the timings of the meals along with temporal properties of sleep episodes, exercise events and entertainment activities the participants did each day.

From the pre-screening survey, 11 participants agreed to participate in this study and the Google form in Appendix B.1 was sent out to them daily for two months.

### 3.1.4 Data Collected from Our Deployment

For comprehensive evaluation, we deployed our system in a 30-year old male graduate student's home who lives alone and collected data for six months continuously. Table 3.1 shows the list of sensors we deployed. Among the sensors, there are two USB microphones, one in bed room and the other in the living room, each of which is connected to a laptop. The resident used the microphones to record the start and end times of all his daily activities for these six months with help of our ground truth collection system *Vocal-Diary* (described in Section 3.2). These activity labels are used as ground truth to evaluate our activity recognition algorithm *AALO*, and also to evaluate our anomaly detection system *Holmes*. The daily activities logged by the resident include 'sleep', 'breakfast', 'dinner', 'lunch', 'prepare meal', 'cook', 'snack', 'dishwashing', 'watching TV', 'toilet', 'shower', 'laptop use', and 'out of home'.

Location	Object	Symbol	Location	Object	Sysmbol
Bed Room	Microphone	X1	Living Room	Microphone	X2
Bed Room	Motion	C8	Living Room	TV	T1
Bed Room	Bed	B2	Kitchen	Motion	C2
Bed Room	Laptop	L2	Kitchen	Refrigerator	L1
Bath Room	Sink	F1	Kitchen	Freezer	M1
Bath Room	Toilet Flush	O1	Kitchen	Microwave	N1
Bath Room	Shower	E1	Kitchen	Toaster	P1
Bath Room	Motion	C6	Kitchen	Plates Cupboard	D1
Living Room	Motion	C4	Kitchen	Sink	H1
Living Room	Main Door	K1	Kitchen	Spice Cabinet	B1
Living Room	Dining Table	D2	Kitchen	Stove	S1

**Table 3.1: List of Sensors in Our Deployment.**

The bed sensors we used are made with accelerometer sensors as used in [31]. For detecting the episodes when the resident watches TV we used a Hobo data logger [70] that has a light sensor. We placed this logger just in front of the TV so that the light sensor directly faces the

TV screen. Based on the amount of light it senses and the variation in light, we can detect when the TV is ON. To detect when the stove is ON, we attached a temperature sensor to the stove surface. All the other sensors in Table 3.1.4 are X10 motion and contact sensors.

To collect data from the X10 sensors, we used the framework presented in [31] that requires an X10 receiver connected to the laptop. The temperature sensor for the stove and the Hobo data logger for TV store data in their internal flash memory. We downloaded data from them periodically, extracted the events based on thresholds and used the events as binary data. To process the audio data collected from the microphones and store the activity labels, we designed, implemented and evaluated *Vocal-Diary* which is described in the next section.

## 3.2 Vocal-Diary

Activity recognition systems based on in-home sensors are used for different applications such as home health care, energy monitoring, and security. Accurate annotation of daily activities, i.e., ground truth is necessary for training activity recognition systems. Accuracy of activity recognition systems depends on sufficient amount of ground truth for training. Ground truth collection requires active participation from residents which is challenging. The challenges include ensuring comfort and privacy of residents, maintaining accuracy in ground truth, considering different real home scenarios that may cause errors in activity labeling, and addressing the reality that residents may forget to log activities.

Existing ground truth collection systems suffer from different shortcomings such as use of cameras [19, 20] which is a privacy concern and very tedious, real-time logging by the user [6, 4] which is not suitable in real home settings for sufficiently long training duration, manual annotation of activity labels from sensor data based on some rules [21, 22] which is time-consuming and may not be always accurate, and necessity of wearing a microphone that may be uncomfortable for users [5, 8]. Also, none of the existing ground truth collection

systems address the fact that residents may forget to log the begin and / or end of activities occasionally.

We present *Vocal-Diary*, a voice command based ground truth collection system where residents log activities by specific voice commands, that we designed, implemented, evaluated, and used in our deployments. *Vocal-Diary* is privacy-aware, and robust to different environmental noise in the home and day-to-day conversations among the residents by using two-way acknowledgments. To increase robustness, *Vocal-Diary* uses a speaker recognition system that is trained with the voice segments of all the residents in a home to ensure that the voice commands are spoken by the residents. Also, *Vocal-Diary* utilizes the sensor data produced by the underlying activity recognition system to query a resident periodically to check if he / she forgot to log any activity.

The main contributions of *Vocal-Diary* are:

1) A novel ground truth collection system to collect accurate activity labels by listening to specific voice commands from residents with help of one / more microphone(s) in the home. The system will be made publicly available so that other research groups can use it.

2) The novelty of *Vocal-Diary* includes the use of two-way acknowledgement for listening to voice commands from a resident and integration of speaker identification in the pipeline for robustness. *Vocal-Diary* is privacy-aware, does not need a resident to carry a microphone (in each room, one microphone is placed in a suitable place), and a resident does not need to manually turn the microphone on / off when giving voice commands.

3) Another novel feature of *Vocal-Diary* is to query residents periodically to check if they forgot to log any activity by voice commands with the help of the sensor data produced by the underlying activity recognition system.

4) We evaluate *Vocal-Diary* by deploying in three homes (two single-resident, one double-resident) for one month each. Results show that *Vocal-Diary* increases precision by at least 40% and recall by at least 10% compared to a one-way voice command recognition based system.

The rest of this section is organized as follows. Section 3.2.1 details different components of the *Vocal-Diary* system. Section 3.2.2 describes how *Vocal-Diary* queries residents based on sensor events. Section 3.2.3 presents the advantages of *Vocal-Diary*. Section 3.2.4 describes the details of experiments and evaluation results. We conclude in Section 3.2.5.

### 3.2.1 System Description

Here we describe the two key components of *Vocal-Diary*: voice command recognition and speaker recognition. Following that we explain how these two components are used together to build the end-to-end system of *Vocal-Diary*.

#### Voice Command Recognition

We implement the voice command recognition program using Microsoft Speech API (SAPI [28]) in C# using the .NET environment. We use a recognition grammar that employs the following usage pattern: a resident begins an activity saying “system <activity name> begin”. *Vocal-Diary* then plays back a pre-recorded audio file that asks for confirmation of the same activity being started such as “you are beginning to <activity name>”. If the system understood correctly, the resident then says “system yes” as a two way confirmation.

We find from tests that two-way acknowledgement is necessary since the microphones are often moderately far from residents and there may be different environmental noise, so the accuracy of voice command recognition is not always perfect. When the resident finishes the activity, he / she needs to say “system <activity name> end”, and the same confirmation above is used. For start / end of any activity, the activity name, timestamp, and start / end status are logged. The recognition grammar consists of a fixed vocabulary of activities.

#### Speaker Recognition

Despite using the voice command recognition with two-way acknowledgement, sometimes different environmental noise or daily conversation may be erroneously recognized as commands

from residents. To address this, we implement a speaker recognition program to classify such noise as not being spoken by the residents. We use the open-source MARF framework [71] to implement the speaker recognition program in Java. We convert the speaker recognition program in Java to a .jar executable and invoke it from the voice command recognition program in C#.

The speaker recognition program needs training. At the start of a deployment, each resident speaks the voice segments “system <activity name> start” and “system <activity name> end” for each activity, and the two segments “system yes” and “system no”. These voice segments from each resident along with few segments that contain different sources of noise (sounds from kitchen appliances, footsteps, opening and closing of doors, washing machine, and TV) construct the training data. We have seen from our experiments that ambient noise such as these are often detected as voice commands by the Microsoft speech API.

The noise sounds are recorded at the start of deployment in each home. In our experiments, residents give voice commands when within 1 – 5 feet of a microphone (during both training and testing). Note that we do not consider the scenario when a resident may give a command when there is ambient noise in the environment (e.g., when the TV is on). However, we believe *Vocal-Diary* can also address such scenarios by training with such recordings (i.e., giving commands when TV or microwave is on). Currently the utility of the speaker recognition system is in identifying the cases when sounds from the environment are wrongly detected as commands by the voice recognition module.

The MARF framework provides options for different pre-processing (normalization, end-pointing, high, low, and band pass filters), feature selection (e.g., fast fourier transform, linear predictive coding), and classification (e.g., neural networks, nearest neighbors) techniques. We train with different combinations and use the combination that performs the best on the training data. This combination may be different for different homes. The trained program is used for speaker recognition in real time.

## End-to-End System

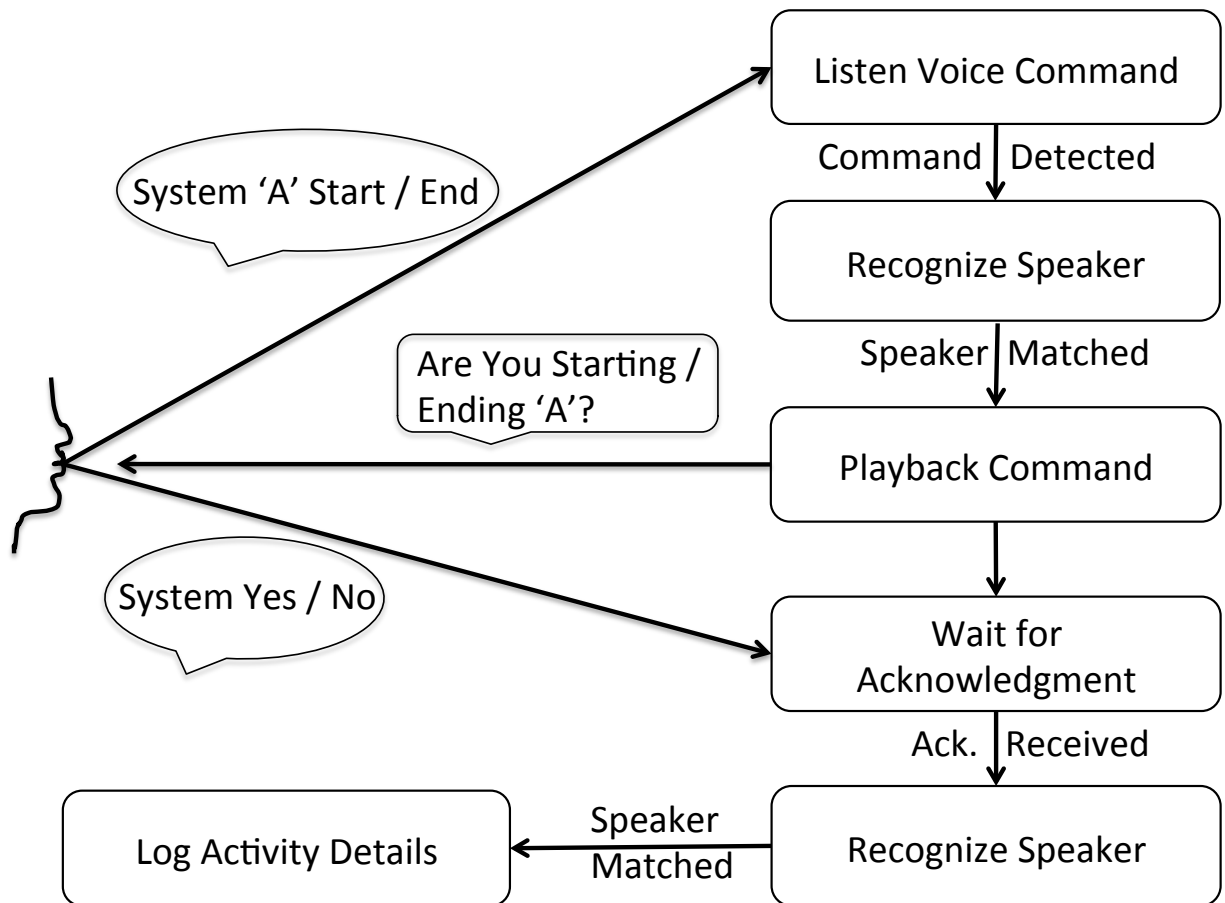
Figure 3.1: *Voice-Diary* End-to-End System.

Figure 3.1 shows the end-to-end system. ‘Listen Voice Command’ module waits for a voice command in the specific format described above. When it listens to a command (correctly or incorrectly), it invokes the ‘Recognize Speaker’ program to check if the voice belongs to any of the residents. If it does not, then *Vocal-Diary* logs this error in a separate debug file. If it does, then the ‘Playback Command’ finds the pre-recorded audio file for that specific activity’s start / end. Note that for each activity there are two pre-recorded audio files in the system that plays back the query whether the resident actually started / finished a particular activity. After playing the correct audio file, ‘Wait for Acknowledgment’ module waits for acknowledgement.

After listening to the audio played back, the resident acknowledges by saying “system yes” if it was the correct command. Else the resident says “system no”. If the resident says ‘no’ or does not say anything, *Vocal-Diary* does not write anything in the activity log, but it logs this error in a separate debug file. If the resident acknowledges yes, *Vocal-Diary* once again verifies if this voice segment belongs to the same resident of the home. If it does, then finally *Vocal-Diary* writes the timestamp, activity name and start / end status in the activity log.

Ideally, there should be one microphone in a suitable place in each room of the home. *Vocal-Diary* logs voice commands collected by each microphone in separate files and combines all of them offline. We can use wireless microphones in each room that send data to a laptop in the home. Another approach can be using a USB microphone connected to a Beaglebone with flash memory in each room as used in [72]. *Vocal-Diary* is totally privacy-aware. Therefore, residents should be comfortable with the presence of a microphone in each room. However, if a resident does not want any microphone in a particular room / rooms, the voice commands have to be given in a room where there is a microphone.

### 3.2.2 What if Residents Forget

*Vocal-Diary*, or any other ground truth collection system requires a resident to inform the system (by voice commands in case of our system) before starting and ending each activity. However, in reality a resident may forget to do so occasionally. Therefore, *Vocal-Diary* also reminds a resident periodically to give voice commands.

*Vocal-Diary* utilizes the sensors that the underlying activity recognition systems use (e.g., motion, contact, door, bed sensors). If *Vocal-Diary* can access such data, then it works in the following way to remind a resident:

a) After entering each room, if a resident uses one or more objects in the room (this can be detected from the sensor firings) but does not log any activity with voice command within *ENTRY\_THR* minutes, then *Vocal-Diary* queries the resident “Have you forgotten to log

an activity?”. In reply, a resident can ignore the query or log an activity.

b) If a resident ignores a query, then *Vocal-Diary* does not query again within *REPEAT\_THR* minutes which is configurable. This is to ensure that the queries do not become a nuisance for residents. If the resident does not want to listen any such query at all, *Vocal-Diary* is configured accordingly.

Note that this feature is dependent on the availability of the sensor data generated by the underlying activity recognition system which is true in one of our deployments.

### 3.2.3 Advantages

#### Privacy-Aware

*Vocal-Diary* is privacy-aware, as it does not record day-to-day conversation. It only listens to voice commands in specific format. Moreover, whenever a resident gives voice commands, their raw voice is not recorded. Only the timestamps, activity names and start / end status are logged. The audio files recorded during training contain the voice of a resident. However, we delete them as they are not used after training. We believe that these features will make residents feel comfortable in using *Vocal-Diary* in their homes.

#### Robust

*Vocal-Diary* is robust in the presence of different environmental noise that may arise in a real home settings including, but not limited to sound from TV, music player, dish washer, washing machine, microwave, coffee maker, footsteps, opening / closing of doors. Such noise can be recognized as voice commands by Microsoft Speech API SAPI [28]. However the use of speaker recognition and two-way acknowledgement by *Vocal-Diary* makes sure that they are not logged as ground truth. Also, residents can have regular conversation with each other, with visitors, or with someone over phone.

## Ease of Use

Unlike [5, 8], *Vocal-Diary* does not require a resident to wear any headset. A microphone can be placed in any suitable location in the room because *Vocal-Diary* is robust even in the presence of different environmental noise. In our experiments, residents give voice commands when within 1 – 5 feet of a microphone and when there is no ambient noise from the environment (e.g., TV). We believe with adequate training, *Vocal-Diary* can also work in the scenarios when residents give commands in the presence of noise. Also, residents do not need to turn on / off the microphone before speaking each command as the microphone is always on. However, it only listens to specific commands and does not record any other noise or conversation.

## Supports Multi-Resident Homes

If there are multiple residents in a home, *Vocal-Diary* is initially trained with each of the resident’s voice. After training, all residents can interact with *Vocal-Diary* in the same way. *Vocal-Diary* identifies each voice based on the speaker recognition program and adds the identifier along with other information in the log file.

### 3.2.4 Evaluation

The evaluation consists of three parts. First, we evaluate how accurately *Vocal-Diary* recognizes voice commands. Then we evaluate the effectiveness of querying the residents. Finally, we investigate the feasibility of voice commands as a ground truth collection mechanism.

We deployed *Vocal-Diary* in two single-resident and one two-resident homes for one month each. For evaluation, raw audio of any voice command detected by *Vocal-Diary* (correctly or incorrectly) is saved as a .wav file so that we can listen offline to verify if it is a voice command. Note that the files are only saved for evaluation purpose. We listen to the recordings to calculate how many activity logs were actually voice commands, i.e., number of true positives (TP), how many voice commands were not logged as activities, i.e., number

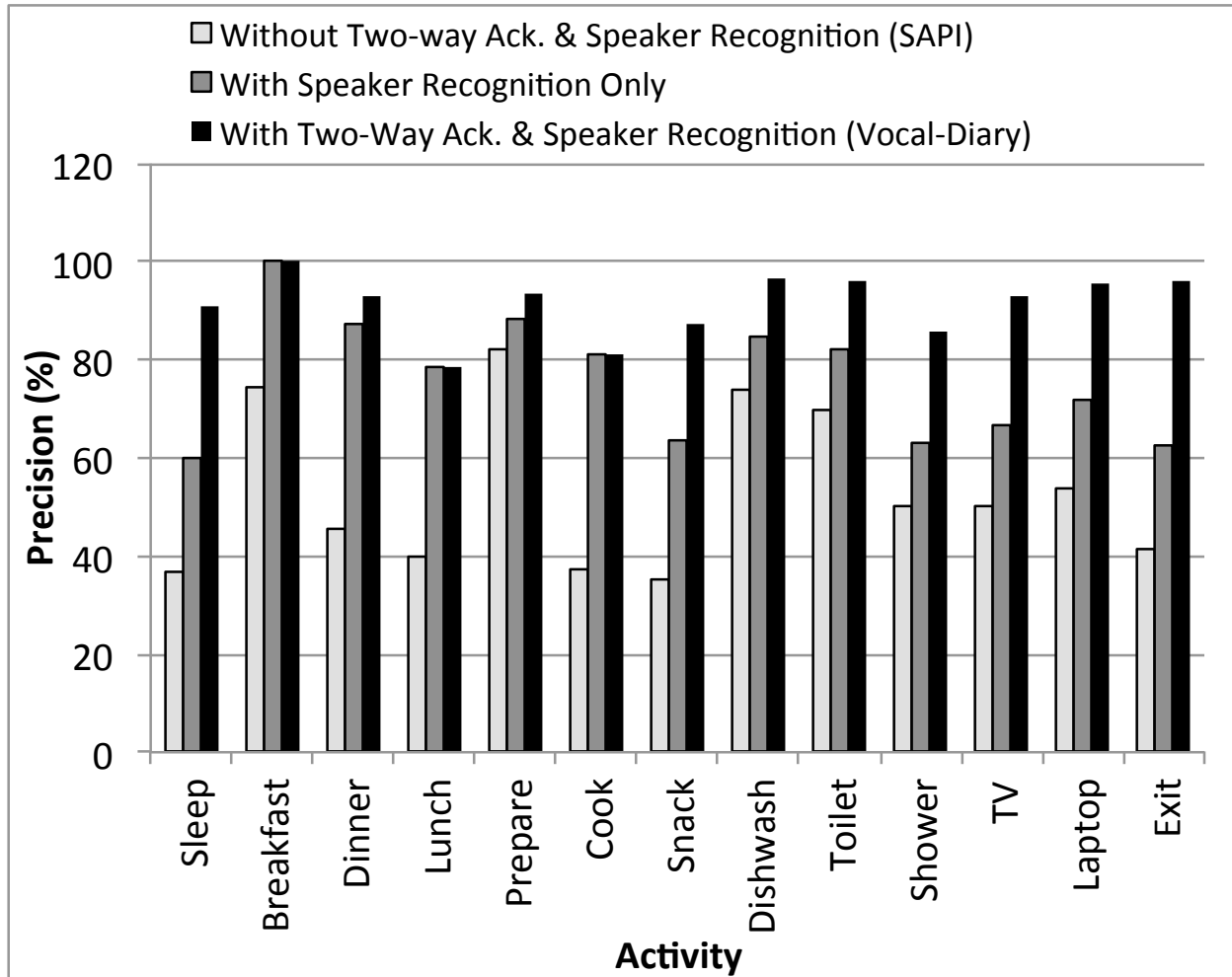
of false negatives (FN), and how many recordings containing noise or other conversation were logged as activities, i.e., number of false positives (FP). As evaluation metrics, we use precision and recall which are defined in Equation 3.1. We got for IRB approval for such experiments (See Appendix C.1). The residents knew their voice recordings are recorded for evaluation.

$$\begin{aligned} precision &= \frac{TP}{TP + FP} \\ recall &= \frac{TP}{TP + FN} \end{aligned} \tag{3.1}$$

### Voice Command Accuracy

Figure 3.2 shows precision values for single-resident home ‘1’. For all activities, the basic SAPI based system (which has no two-way acknowledgement and no speaker recognition) has very low precision values. This is mainly due to different sounds generated by environmental noise. Using speaker recognition increases the precision values significantly for all activities as it helps in removing the false positives. False positives may also occur due to an actual voice command being detected as a different voice command and / or other day-to-day conversations by residents being detected as voice commands. Integrating two-way acknowledgement system, which ensures that *Vocal-Diary* does not log any detected command without acknowledgement, helps in removing such false positives. Still there may exist some false positives, because either sometimes noise is detected as “system yes” and/or error in speaker recognition.

Figure 3.3 shows the recall values for the same home. The SAPI based system has false negatives, this is because sometimes a voice command is detected as another one. This is due to the variation in the ways different people utter the same word. Because SAPI does not train per speaker, such errors are not surprising. Adding the speaker recognition feature cannot remove all such errors. However, *Vocal-Diary* removes such false negatives with the help of two-way acknowledgments and achieves 100% recall for this single-resident home.



**Figure 3.2:** Precision values for single-resident home 1. Adding speaker recognition increases the precision for all activities, and adding two-way acknowledgement increases it more.

Figure 3.4 and 3.5 show average precision and recall values for all activities in the three homes; home 3 is the double-resident home. For all three homes, *Vocal-Diary* increases precision values significantly and achieves 100%. On average, *Vocal-Diary* increases precision by at least 40% and recall by at least 10% compared to a SAPI based system without two-way acknowledgement and speaker recognition.

In calculating precision and recall for the double-resident home, *Vocal-Diary* is considered accurate if it correctly differentiates sounds caused by environmental noise from voice of residents. Whether it can assign a voice command to the correct resident is not considered.

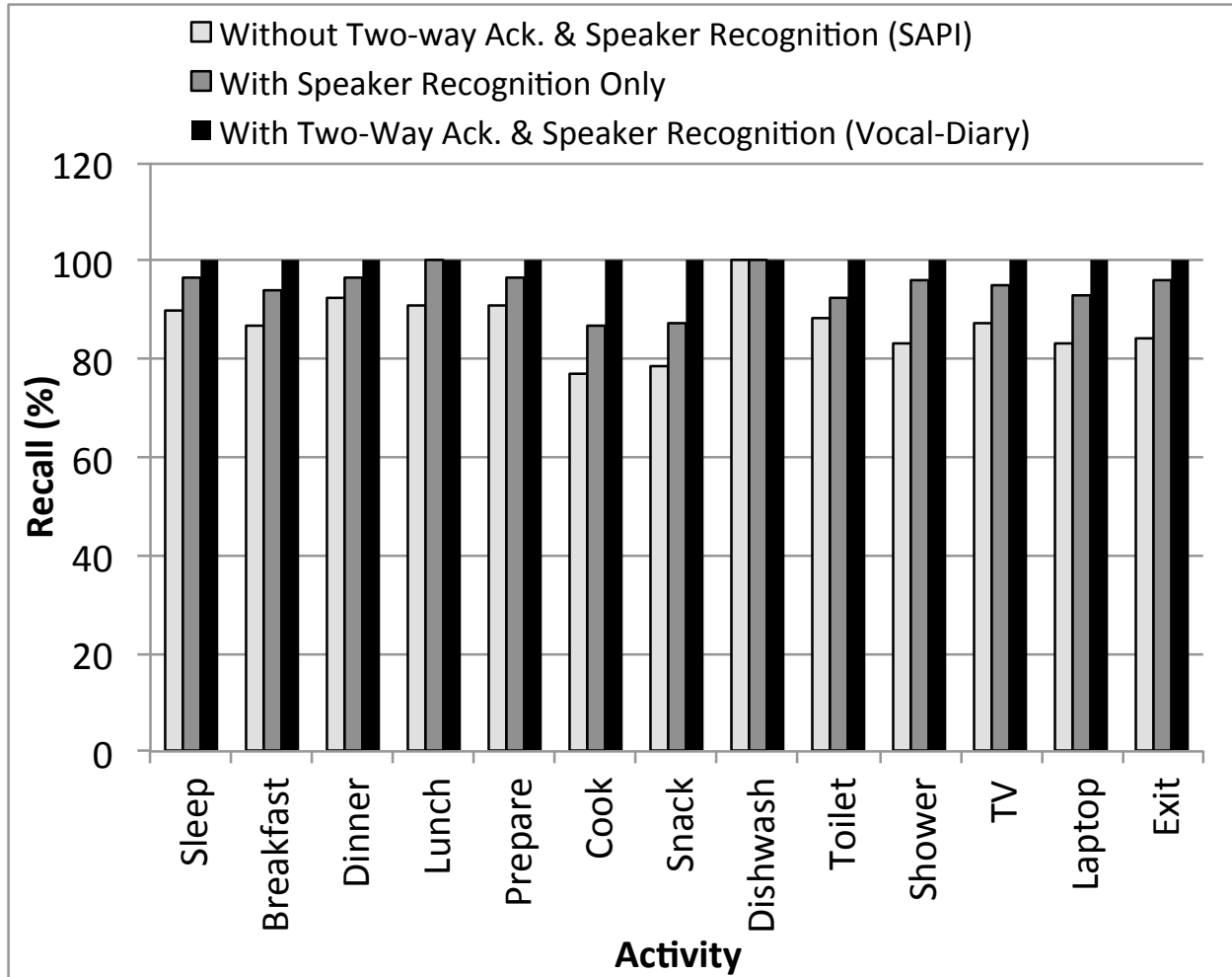
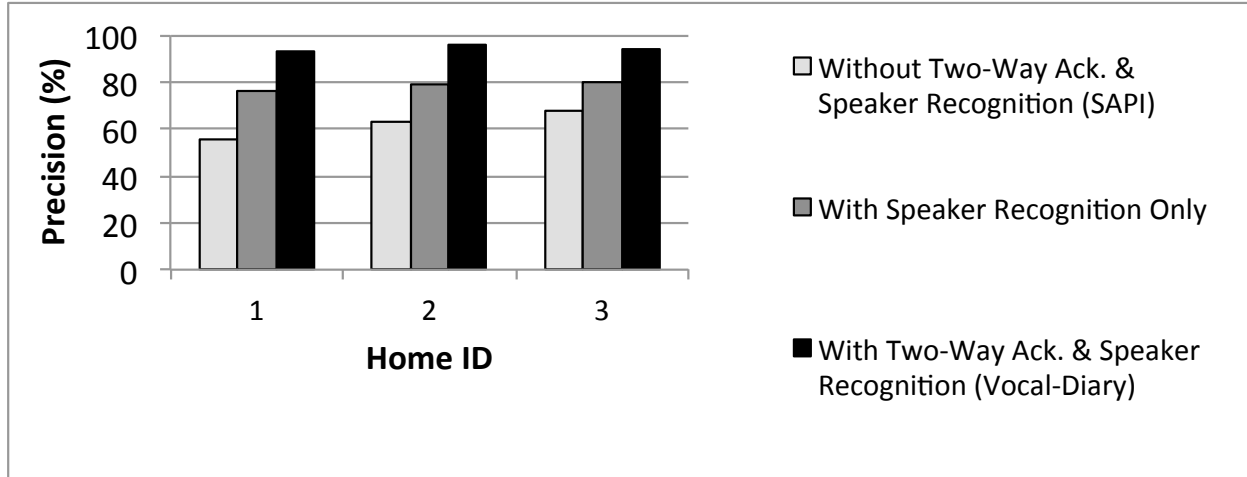


Figure 3.3: Recall values for single-resident home 1. *Vocal-Diary* achieves 100% recall for all activities with the help of speaker recognition and two-way acknowledgement features.

However, if we consider that, the average precision and recall values for all activities drops to 88% and 95%, respectively. MARF framework does not support advanced features (e.g., MFCC) and classifiers (e.g., support vector machine). Implementing a speaker recognition program using these would increase accuracy.

### Effectiveness of Querying Residents

To evaluate the effectiveness of querying the residents, we deploy the system in single-resident home ‘1’ for 15 days. During this time, *Vocal-Diary* had access to all the sensor data from the

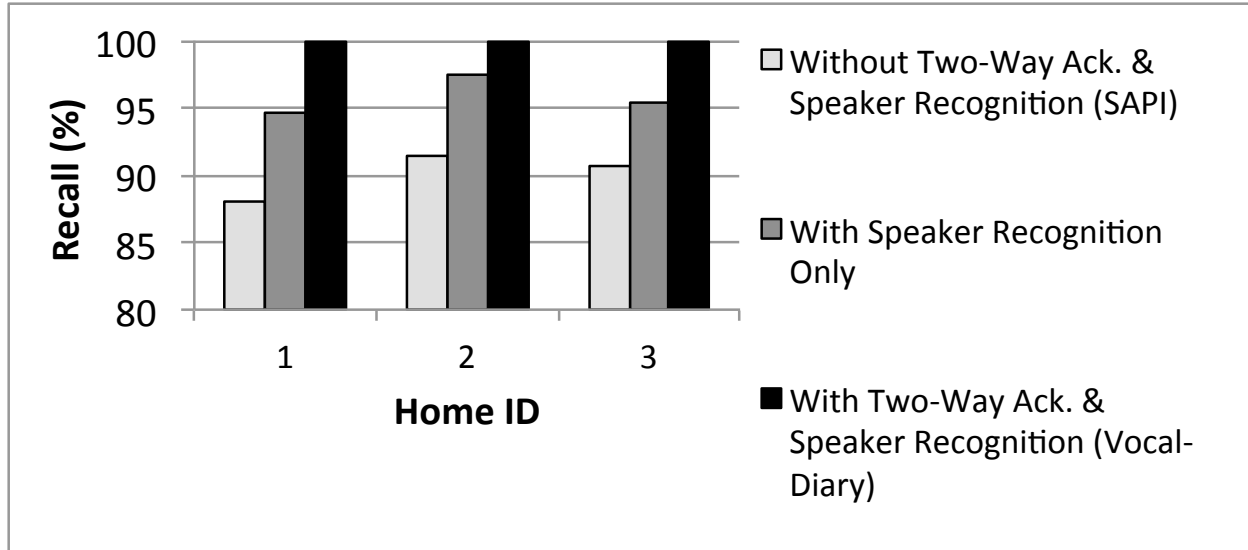


**Figure 3.4: Average precision values over all activities for three homes. Home 3 has two residents; Home 1 and 2 have single resident.**

activity recognition system that include motion sensors in each room, door sensors, contact sensors in various objects of daily use (e.g., microwave, freezer, sink, toilet, shower), and pressure pads in bed and chair. We set the value of *ENTRY\_THR* as 1 minute and the value of *REPEAT\_THR* as 5 minutes. During the 15 days, the resident did not log activity start / end times by voice commands on purpose for 25 times; so the experiments here are in a controlled setting. *Vocal-Diary* accurately detected all the 25 instances and queried the resident for the ongoing activity status. This shows that *Vocal-Diary* can help in logging activities by querying the residents when they forget to do so.

### Feasibility of Voice Commands to Collect Ground Truth

So far we have shown results from evaluation of how accurate *Vocal-Diary* is in recognizing voice commands and in querying residents if they forget to give voice commands. However, we also need to evaluate the feasibility of using such a system for a long period. If the residents do not give voice commands or do not reply to the queries, those scenarios will not be reflected in the above experiments. Therefore, we evaluate *Vocal-Diary* by deploying it along with an underlying activity recognition system (that uses in-home sensors) for three



**Figure 3.5:** Average recall values over all activities for three homes. Home 3 has two residents; Home 1 and 2 have single resident.

months continuously in single-resident home ‘1’. During this deployment, the feature of querying the resident was not used.

Here, we need ground truth to evaluate our ground truth collection system. From the sensor firings, all activity segments were manually annotated offline as in [22] with feedback from the resident who also logged activities using *Vocal-Diary*. Results show that there were 992 total activity instances during the three months of which 59 activity instances were not logged by voice commands. Therefore, the resident used *Vocal-Diary* to log activities in more than 94% of the cases. Given the effectiveness of querying residents discussed above, we hypothesize that most if not all of the 6% of missed activities would have been captured if the querying was used. In the future we plan to compare *Vocal-Diary* to ground truth detection based on offline logging and cameras.

### 3.2.5 Summary of *Vocal-Diary*

*Vocal-Diary* is a privacy-aware, easy-to-use and robust ground truth collection system based on voice commands which shows high accuracy in evaluation. The accuracy in voice command

recognition is achieved by two-way acknowledgement and speaker recognition. The ease of use, robustness, and high accuracy come at the cost of additional training and microphones in each room. However, the microphones are inexpensive (with Beaglebones [72]) and the training effort is minimal (few minutes per resident). Comprehensive evaluation is necessary for the effectiveness of querying residents.

Finally, the software will be made publicly available online at [www.cs.virginia.edu/Vocal-Diary/VoiceDiaryzip](http://www.cs.virginia.edu/Vocal-Diary/VoiceDiaryzip).

### 3.3 Conclusions

To evaluate the new algorithms and systems presented in this dissertation, we use two public data sets, one commercial safety system dataset, data collected by online survey for two months from 11 participants, and data collected from our own deployments (that use *Vocal-Diary* to collect ground truth for six months). This chapter describes all these sources of data in detail and presents the design and evaluation of *Vocal-Diary*.

## Chapter 4

# AALO: Activity Recognition using Active Learning

In this chapter, we present the design, implementation and evaluation of *AALO*, our novel activity recognition system for single person smart homes using active learning in the presence of overlapped activities. *AALO* applies data mining techniques to cluster in-home sensor firings so that each cluster represents instances of the same activity. Users only need to label each cluster as an activity as opposed to labeling all instances of all activities. Once the clusters are associated to their corresponding activities, our system can recognize future activities. To improve the activity recognition accuracy, our system preprocesses raw sensor data by identifying overlapping activities.

The rest of the chapter is organized as follows. Section 4.1 discusses the motivation of *AALO*. Section 4.2 lists the technical contributions. This is followed by a description of our novel framework for training activity recognition systems in Section 4.3. Section 4.4 details how to use our system for activity recognition. Section 4.5 then presents evaluation results. We conclude in Section 4.6.

## 4.1 Motivation

Due to increasing number of elderly people and single households living alone, wide-spread deployment of sensors has become prominent in home environments for detecting medical emergencies and assessing behavioral changes. In such smart homes, living spaces and objects used for daily activities are instrumented with passive sensors. When a resident moves from one room to another or uses different objects that have attached sensors, a series of sensor firings with corresponding timestamps are generated that allow us to automatically detect which activity the resident is currently performing, its duration and what objects are used for this activity. Accurate detection and summarization of these daily activities are essential for many remote home healthcare applications such as assessing behavioral rhythms ([2, 3]), monitoring cognitive decline ([4]). In this work, our focus is on unobtrusive long-term monitoring of daily activities of single person homes on a daily basis.

However, existing activity recognition algorithms suffer from many practical problems. Many of them ([5, 6, 7, 8, 9]) are based on supervised learning where the training data needs ground truth i.e., accurate labeling of all activities. To ensure high accuracy, the classifiers need to be trained with long traces of data that may range from months to years. However, collecting ground truth for such a long period is difficult. Either the resident has to keep record of all the activities which is not convenient or we need to use cameras and label each activity manually which may not be practical. There are some existing unsupervised activity recognition algorithms that do not need ground truth ([10, 11, 12]). They either require mining activity models from web definitions or depend on domain knowledge regarding activities and the environment (e.g., which objects are used during an activity). Such systems may not be generalized to wide variety of home environments and deployments. Therefore, we need an activity recognition system that does not require either accurate labeling of all instances of all activities by users, or any specific domain knowledge about activities and environments.

We approach this problem with the insight that different daily activities are performed in

different rooms, each of them triggers a different set of sensors to fire and is often performed during similar time periods of day. Therefore, we divide the sensor firings into room-level occupancy episodes (segmentation), and then for each room, we find the group of sensors that are frequently fired together (mining) in similar times with similar durations (clustering). Our hypothesis is that each such group represents a daily activity and if we can automatically detect these groups from the raw sensor firings, users can just label each group as an activity so that all the instances of this group can be automatically labeled. Our approach is a type of active learning; a learning technique where the system chooses the subset of training data that needs to be labeled by users ([73]). We use unsupervised clustering to find the clusters representing daily activities and users need to label each cluster as one activity.

However, one practical problem that needs to be addressed is overlapped activities. In any clustering based activity recognition approach that do not use activity labels for each individual activity instance for training, overlapped activities introduce challenges as activities may span across multiple clusters or there may be multiple activities within the same cluster. Existing activity recognition algorithms, that are not supervised, do not address these challenges. For example, people may leave the kitchen in the middle of cooking to do something else and come back again to finish cooking. Here, one activity spans multiple occupancy episodes. Alternatively, people may cook and have a drink at the same time while in the kitchen. Here, multiple activities occur in the same occupancy episode. To address these challenges, clustering based activity recognition algorithms (which *AALO* is) have to learn how activities are done without being overlapped so that when activities are overlapped, each individual instances can be identified. This is what *AALO* accomplishes.

## 4.2 Contributions

In this paper, we present a novel **A**ctivity recognition system for a single person home using **A**ctive **L**earning considering **O**verlapped activities (*AALO*). Our main contributions are:

- A novel framework for training activity recognition systems that includes segmentation, mining and clustering of low level sensor events. The novelty in the segmentation step is in the way *AALO* preprocesses raw sensor data by identifying overlapping activities across multiple occupancy episodes which is a concern for any clustering based activity recognition approach. The novelty in the itemset mining and the clustering steps is in the way *AALO* captures the temporal, spatial and object use regularities in the activities of daily living to represent them in terms of these regularities.
- An activity recognition system based on active learning that automatically recognizes new room-level occupancy episodes as members of one of the clusters (i.e., activities) constructed during training. The novelty is in the use of active learning so that after training users only need to label each cluster as one activity. *AALO* in this way facilitates feasibility of long term training which ensures high accuracy.
- We evaluate *AALO* with two public datasets that have data from four single resident homes, and data from one single-resident apartment where we deployed our system for six months. Evaluation results show that *AALO* performs as good as the state of the art supervised activity recognition algorithms (e.g., HSMM) even with significantly less amount of ground truth (for example, for three months of training *AALO* requires a resident to label 29 clusters as activities whereas HSMM requires a resident to label 1016 activity instances). *AALO* also performs better than the multi-instance based semi-supervised activity recognition algorithm (with settings that are practical).

### 4.3 Framework For Training

Figure 4.1 shows the block diagram of our training framework for the activity recognition system. The input  $I$  to the system is a sequence of pairs of the form  $(s_i, t_i)$  where  $s_i \in S$  ( $S$  is the set of all sensors deployed in a home) represents a sensor firing at time  $t_i$ . The set of sensors  $S$  may include passive infrared (PIR) to detect motion in a specific area, reed switches

to measure open / close states of doors and cupboards, pressure mats to measure sitting on a couch or lying in bed, float sensors to measure the toilet being flushed; temperature sensors to measure the use of the stove or shower. We assume that each sensor is associated with only one room  $r \in R$  ( $R$  is the set of all rooms in a home) and this information is available to our system.  $I$  consists of all the sensor firings and their corresponding timestamps during the entire training period. Now we describe the different steps of our framework.

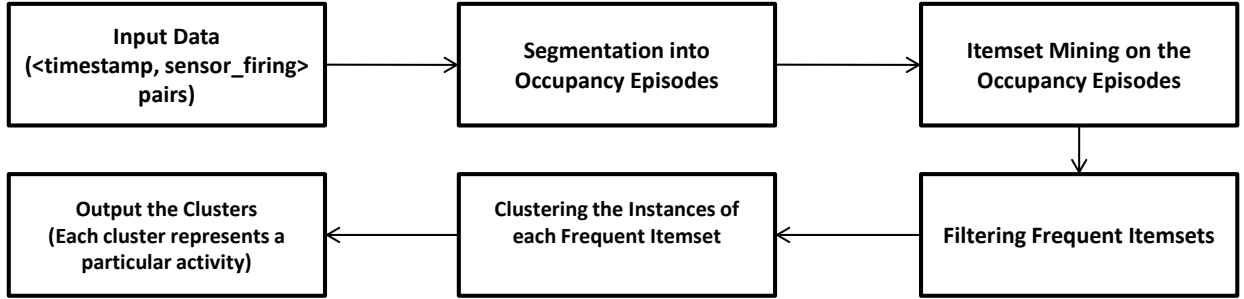


Figure 4.1: Block Diagram of the Training Framework for textitAALO.

### 4.3.1 Segmentation into Occupancy Episodes

Most activities of daily living have spatial regularity. For example, we cook in the kitchen and sleep in the bedroom. Therefore, the first step of our training algorithm is to segment consecutive sensor firings based on which room the sensors are in. Algorithm 1 shows the pseudo code of the segmentation algorithm. The output of this algorithm is a set of room occupancy episodes of the form  $(roomID, entranceTime, duration, usedSensors)$ . When calculating the duration of an occupancy episode at line 9 of this algorithm, we consider the time interval between the last and first sensor firings of that room during that episode. Alternatively, we could consider the time interval between the first sensor firing in the next room and the first sensor firing of the previous room as the duration of the last occupancy episode of the previous room. We did not choose this option, because based on the floor plan of different homes and sensor deployment, sometimes it may happen that in between being in

these two rooms, the resident was in a room where there are no sensors. Therefore, choosing this option would lead us to infer that the resident was in the last room for the entire period.

---

**Algorithm 1** Segmentation Algorithm

---

```

1: {Input:  $I$ , a sequence of pairs of the form  $(s_i, t_i)$ }
2: {Output:  $E$ , a set of room occupancy episodes of the form
   ( $roomID_i, entranceTime_i, duration_i, usedSensors_i$ )}
3:  $E = []$ ;
4:  $previous\_room = room[s_1]$ ;
5: for each  $\langle s_i, t_i \rangle$  in  $I$  do
6:   if  $room[s_i] \neq previous\_room$  then
7:      $new\_segment.room\_ID = previous\_room$ ;
8:     { $new\_segment.entranceTime$  = start time of the last occupancy episode;}
9:     { $new\_segment.duration$  = duration of the last occupancy episode;}
10:    { $new\_segment.usedSensors$  = sensors fired during the last occupancy episode;}
11:     $E.add(new\_segment)$ ;
12:     $previous\_room = room[s_i]$ ;
13:   else
14:     {keep track of which sensors are used during this occupancy episode}
15:   end if
16: end for

```

---

Our hypothesis is that if we find some segments that have similar sensor uses, start times and durations over the course of the entire training period, then such segments represent one daily activity. However, one practical problem against this hypothesis arises from the fact that people may not always start and finish an activity within the same occupancy episode. For example, people may get up in the middle of sleeping, visit the toilet and then come back to sleep again. Similarly, in the middle of cooking, people may leave the kitchen to do something else and come back again to finish cooking. In such cases, one instance of an activity can expand across multiple occupancy episodes. Each of these episodes may not have the start times or durations or set of used sensors that are representative of the actual activity. Accordingly, the corresponding instances of that activity would remain undetected. Here we discuss our solution to solve this problem which is generic and can be applied to any home deployment.

### Successive Occupancy Episode Merging

To address this practical problem, we construct a new occupancy episode by merging two occupancy episodes in the same room that are separated by less than a *time interval threshold*. For each room, we define a *time interval threshold* as the normal time interval between two successive visits to the same room. To calculate this threshold for each room, we enumerate time intervals between all the successive visits to that room during the entire training period and calculate the first and third quartiles ( $Q_1$  and  $Q_3$ , defined as the 25th and 75th percentiles). We set the *time interval threshold* of this room as  $(Q_1 - h)$ , which represents the lower outer fence of a corresponding box plot ( $h = 3 \times (Q_3 - Q_1)$ ). Any value less than the lower outer fence is an extreme outlier. We use this technique, because during such occurrences of overlapping activities, the time interval between successive room visits are expected to be shorter than usual and also such occurrences do not occur too frequently. However, if such occurrences are frequent i.e., the resident frequently leaves a room while performing a specific activity in that room, in that case the time interval will no longer be less than the calculated time interval threshold. Accordingly, no new occupancy episodes will be considered. In such cases, that particular activity is usually done in two episodes and each episode will be individually clustered.

For two successive occupancy episodes in a room  $r$ ,  $(entranceTime_i, duration_i, usedSensors_i)$  and  $(entranceTime_j, duration_j, usedSensors_j)$  ( $j > i$ ), if we find that the time interval between them  $(entranceTime_j - entranceTime_i - duration_i)$  is less than  $r$ 's *time interval threshold*, then by merging them we create a new occupancy episode with start time  $entranceTime_i$ , duration  $entranceTime_j + duration_j - entranceTime_i$ , and set of used sensors  $usedSensors_i \cup usedSensors_j$ . We do not delete the two smaller episodes, as each of them may represent one activity by itself. If the merged episode or the smaller episodes do not represent any activity, the clustering step would ignore them.

### 4.3.2 Itemset Mining on Occupancy Episodes

Specific activities of daily living are performed using specific sensors (i.e., objects). For each room  $r$ , there are a set of occupancy episodes of the form  $(r, entranceTime_i, duration_i, usedSensors_i)$ , where  $i = 1, 2, \dots, \dots, \text{Number of Occupancy Episodes in } r$ . In this step, from each of these tuples we only use  $usedSensors_i$  which is of the form  $\{s_{ij}\} (s_{ij} \in S)$ . The temporal characteristics of these tuples are used in the next step. We apply frequent itemset mining ([74]) on  $usedSensors_i$  of all occupancy episodes of room  $r$  where each sensor  $s_{ij}$  is an item and the group of sensors fired in each occupancy episode  $\{s_{ij}\}$  is a transaction. The goal of the itemset mining is to find the groups of sensors (i.e., items) that are frequently fired together (i.e., occur together in transactions); they are called frequent itemsets. Our hypothesis is that each frequent itemset represents an activity.

We use a state of the art itemset mining algorithm Apriori ([74]). For each room  $r$ , we run Apriori separately with  $\{s_{ij}\}$  as input ( $i = 1, 2, \dots, \dots, \text{Number of Occupancy Episodes in } r$ ). As output, we get the set of frequent itemsets  $\{FI_k\} (k = 1, 2, \dots, \dots, \text{Number of Frequent Itemsets for } r)$ , where each  $FI_k$  is a set of sensor firings (i.e., items) of the form  $\{s_{kl}\} (l = 1, 2, \dots, \dots, \text{Number of Sensors in Frequent Itemset } FI_k)$ . An itemset is considered to be frequent if the number of different occupancy episodes (i.e., transactions) in which the itemset occurs is more than a threshold number of days. In the Apriori algorithm, this threshold is called the *support threshold* which we specify before running the algorithm.

After getting the output, for each frequent itemset  $FI_k$ , we find the occupancy episodes in room  $r$  where  $FI_k$  occurs and we construct the set of tuples  $\{(startTime_{km}, duration_{km})\}$ ,  $m = 1, 2, \dots, \dots, \text{Number of Occupancy Episodes where } FI_k \text{ occurs}$ . Here,  $startTime_{km}$  is the earliest timestamp when any sensor  $s_{kl} \in FI_k$  fires during occupancy episode  $m$ . Suppose,  $endTime_{km}$  is the latest timestamp when any sensor  $s_{kl} \in FI_k$  fires during occupancy episode  $m$ .  $duration_{km}$  is defined as the difference between  $endTime_{km}$  and  $startTime_{km}$ . Each frequent itemset along with thus constructed set of tuples are used in the next step for filtering significant frequent itemsets.

### 4.3.3 Filtering Frequent Itemsets

One disadvantage of Apriori is that it produces redundant itemsets. To remove redundant itemsets, we could have used an itemset mining algorithm that produces a set of maximal frequent itemsets. A set of maximal frequent itemsets  $FI$  has the property that it does not have any pair of itemsets  $FI_1, FI_2 \in FI$  ( $FI_1 \neq FI_2$ ) such that  $FI_1 \subset FI_2$  or vice versa. However, in daily activities, it may be normal that  $FI_1$  is a subset of  $FI_2$ , but  $FI_1$  occurs by itself (when  $FI_2$  does not occur) during significant number of occupancy episodes. Suppose, for a room ('kitchen'), two frequent itemsets are  $\{s_1, s_2, s_3, s_4\}$  (represents the activity 'prepare breakfast') and  $\{s_1, s_3\}$  (represents the activity 'prepare coffee'). Here,  $\{s_1, s_3\}$  is a subset of  $\{s_1, s_2, s_3, s_4\}$ , therefore each occupancy episode where  $s_1, s_2, s_3$ , and  $s_4$  fire,  $s_1$  and  $s_3$  also fire i.e., whenever the user prepares breakfast, he also prepares coffee. However, in this room (i.e., 'kitchen'), there may be such occupancy episodes where only  $s_1$  and  $s_3$  fire i.e., the user may also prepare coffee at other times of the day. If such instances are more than the *support threshold*, then both  $\{s_1, s_2, s_3, s_4\}$  and  $\{s_1, s_3\}$  should be included in the set of frequent itemsets so that we can identify both these activities separately. We do not use an itemset mining algorithm that produces a set of maximal frequent itemsets, because it would have deleted  $\{s_1, s_3\}$ . Alternatively, we define and reduce redundant itemsets in the following way.

As stated earlier, we represent an instance  $ins_k$  of a frequent itemset  $FI_i$  by the tuple  $(startTime_{ik}, duration_{ik})$  ( $k = 1, 2, \dots$ , Number of Occupancy Episodes where  $FI_i$  occurs). An instance  $ins_p = (startTime_{ip}, duration_{ip})$  temporally covers another instance  $ins_q = (startTime_{iq}, duration_{iq})$  if  $startTime_{ip} \leq startTime_{iq}$  and  $(startTime_{ip} + duration_{ip}) \geq (startTime_{iq} + duration_{iq})$ . For each instance  $ins_k$  of a frequent itemset  $FI_i$ , we consider the instance  $ins_k$  as *covered* by another frequent itemset  $FI_j$ , if  $FI_i \subset FI_j$  and  $FI_j$  has an instance that *temporally covers*  $ins_k$ . For each frequent itemset  $FI_i$  of a room, we remove all its instances that are covered by instances of other frequent itemsets of the same room. After removing covered instances, if total number of remaining instances of  $FI_i$  is less than the

*support threshold*, we define  $FI_i$  as a redundant frequent itemset and delete it. In this way, we remove redundant frequent itemsets and get the set of necessary frequent itemsets.

Note that, in the set of frequent itemsets we have constructed thus far, for a room, there can be two instances of two different frequent itemsets that may belong to the same occupancy episode. For example, it may be the case that on some days, the user is cooking dinner and at the same time doing laundry in the washing machine in the same occupancy episode. If there are many other days, when the user does not do these two activities during the same occupancy episode, then we will have two different frequent itemsets, one consisting of sensors related to cooking, and the other related to sensors related to washing machine. Some instances of these two will belong to same occupancy episodes. In this way, we can differentiate and identify overlapping activities in the same room. However, if the user always does two activities in the same room, then our system will not be able to differentiate between two such activities and will generate one frequent itemset.

#### 4.3.4 Clustering Instances of Each Frequent Itemset

As we hypothesized earlier, each frequent itemset represents one activity. However, there may be multiple activities that use the same set of sensors (e.g., prepare breakfast and prepare dinner). To differentiate among such activities, we need to consider the times of day when the group of sensors are used and also the durations. Therefore, in this step, we cluster the instances of each frequent itemset based on their temporal characteristics i.e., start times and durations.

We use the DBSCAN clustering algorithm ([75]) which is a density based clustering algorithm. For each frequent itemset  $FI_i$  of a room, we run DBSCAN separately on the set of tuples  $\{(startTime_{ik}, duration_{ik})\}$  ( $k = 1, 2, \dots$ , *Number of not Covered Instances  $FI_i$  has*). We normalize each attribute of each tuple before clustering. The major advantage of DBSCAN is that we do not need to specify how many clusters there are. This is important,

because we do not know how many different activities a resident performs in different rooms. DBSCAN automatically calculates the neighborhood radius of each cluster.

Note that, there may be some frequent itemsets for which some instances are not part of any clusters. During training, we do not consider such instances as part of any activity and ignore them as outliers. After training, when our system is used for activity recognition, we report such instances (that are not within the neighborhood radius of any cluster; details in Section 4.4) as irregular / anomalous behavior. Also, there may be some frequent itemsets for which DBSCAN does not produce any significant cluster. For such frequent itemsets, we try to cluster them again based on durations of each instance only. This is because, there are activities that do not have any regularity about which time of day they occur. However, they may have similar durations (e.g., the user can take a drink at any time of day; the duration of this activity should be similar). Finally, if there is any such frequent itemset for which even clustering based on durations does not produce any significant cluster, we remove that frequent itemset.

After the clustering, for each room, we get one or more clusters each of which represents a particular event that happens in that room, uses same set of objects, may or may not start in similar times and last for similar durations. We consider each of these events (represented by each cluster) as an activity. After training, we present a summary of each cluster (e.g., used sensors, average start time, duration) to the resident so that he/she can label each cluster as an activity.

### 4.3.5 Output

The output is the set of all clusters constructed by applying DBSCAN on all frequent itemsets of all rooms separately. Each cluster  $C_i$  is represented by the tuple  $(Used\_Sensors_i, Mean\_Start\_Time_i, Mean\_Duration_i, Probable\_Label_i)$ .  $Mean\_Start\_Time$  may be null for some clusters. The labeling is done by the resident of the home after looking at the other attributes of the cluster. Once a cluster is labeled by the user, all the instances of that

cluster are automatically labeled as that activity. In this way, users need to label only a small number of clusters instead of labeling all the instances of all the activities.

## 4.4 Activity Recognition

Figure 4.2 shows how to use *AALO* for activity recognition after training. Once the clusters are labeled as specific activities by users, our system can recognize future occupancy episodes as activities. During activity recognition, we construct room occupancy episodes from raw sensor firings as described in the previous section. As each room occupancy episode  $e$  is created, we take its set of used sensors  $u_e$  and find the set of clusters  $\{C_j\}$  such that  $Used\_Sensors_j \subset u_e$  ( $j \in \{1, 2, 3, \dots, \dots, Number\ of\ Clusters\}$ ). There can be multiple such clusters. Because, the user may do multiple activities in the same occupancy episode or, even with the same set of used sensors, there may be multiple activities based on the temporal characteristics.

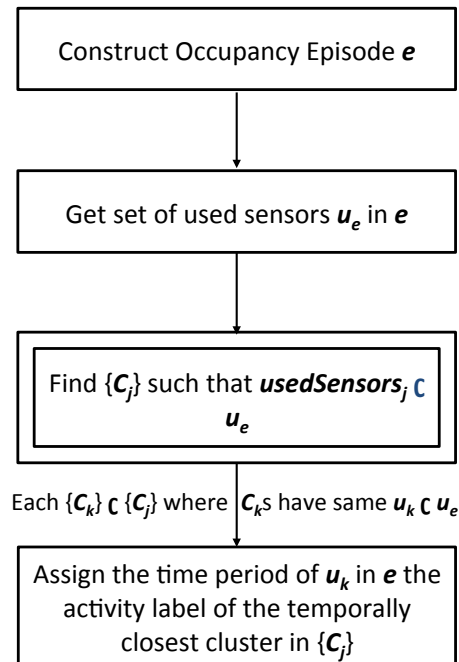


Figure 4.2: Using *AALO* for Activity Recognition

For each  $\{C_k\} \subset \{C_j\}$  ( $j, k \in \{1, 2, 3, \dots, \dots, \text{Number of Clusters}\}$ ) such that all the clusters in  $\{C_k\}$  have the same  $Used\_Sensors = u_l$  ( $u_l \subset u_e$ ), we construct the tuple  $(startTime_{el}, duration_{el})$ . Here,  $startTime_{el}$  is the earliest timestamp when any sensor  $s_e \in u_l$  fires during occupancy episode  $e$ . Suppose,  $endTime_{el}$  is the latest timestamp when any sensor  $s_e \in u_l$  fires during occupancy episode  $e$ .  $duration_{el}$  is defined as the difference between  $endTime_{el}$  and  $startTime_{el}$ . From the set of clusters  $\{C_k\}$ , we find the cluster that is temporally nearest to the tuple  $(startTime_{el}, duration_{el})$  and also has the tuple within its neighborhood radius. We assign the episode  $(startTime_{el}, duration_{el})$  the activity label of that cluster. If we do not find any such cluster, then we report this episode as an irregular one.

The cluster  $C_i \in \{C_k\}$  is temporally nearest to the tuple  $(startTime_{el}, duration_{el})$  if  $(startTime_i, duration_i)$  is the nearest point from  $(startTime_{el}, duration_{el})$ . This helps in differentiating among activities that trigger the same set of sensor firings, but are done in different times of day or last for different durations. Note that, multiple activities can take place in the occupancy episode in a room, and we accommodate such scenarios by assigning activity labels to each episode of  $\{(startTime_{el}, duration_{el})\}$ .

## 4.5 Evaluation

In this section, we detail the evaluation of *AALO* and present the results. First, we present the evaluation results on the dataset described in 3.1.1 (Kasteren dataset). Then, we evaluate *AALO* on the data we collected from our deployment as described in Section 3.1.4. Finally, we detail the evaluation results on the dataset described in Section 3.1.1 ('Casas Dataset 1').

### 4.5.1 Evaluation on Kasteren Dataset

In this section, we present the evaluation results on the Kasteren dataset as described in Section 3.1.1. The dataset contains 26 days of data. The list of sensors and their corresponding

symbols are shown in Table 4.1.

Sensor Location	Symbol	Sensor Location	Symbol
<i>Microwave</i>	a	<i>Dishwasher</i>	h
<i>Bathroom Door</i>	b	<i>Toilet Flush</i>	i
<i>Toilet Door</i>	c	<i>Freezer</i>	j
<i>Cups Cupboard</i>	d	<i>Pans Cupboard</i>	k
<i>Fridge</i>	e	<i>Washing machine</i>	l
<i>Plates Cupboard</i>	f	<i>Groceries Cupboard</i>	m
<i>Frontdoor</i>	g	<i>Bedroom Door</i>	n

**Table 4.1: List of Sensors in the Kasteren Dataset.**

### The Set of Clusters

Room	Number of Clusters
Bedroom	1
Toilet	4
Shower	1
Kitchen	12
Outside	1

**Table 4.2: Number of Clusters for Each Room in the Kasteren Dataset.**

To show the accuracy of clustering, we train our system with 26 days of data. Accordingly, for each room, we get a number of clusters (shown in Table 4.2) each of which corresponds to an activity that is performed in the room. From this table we see that there are a total of 19 clusters. Users only need to label these 19 clusters as activities after we generate the clusters. If we use any supervised learning system, then user has to manually label all instances of all activities (291 in total as shown in Table 4.3). In this way, our system ensures feasibility of long term training. Note that, each cluster contains a number of instances of the activity it represents; as a threshold in DBSCAN, we set the minimum number of instances in each cluster to 15% of the total number of days in the dataset (i.e., four days). Also, in the itemset mining algorithm we set the *support threshold* as 15% of the total number of days. These

Activity	Number of Instances
Sleep	24
Prepare Breakfast	20
Prepare Dinner	9
Get Drink	20
Get Snack	12
Load / Unload Dishwasher	9
Load / Unload Washing Machine	7
Use Toilet	114
Take Shower	23
Brush Teeth	16
Leave House	34
Receive Guest	3

**Table 4.3: Instances per Activity in the Kasteren Dataset.**

thresholds ensure that each candidate cluster has to have activity instances in at least four different days.

Table 4.4 shows the set of clusters generated for *Kitchen*. We label each cluster by ourselves, based on the sensors, time of day and duration of each cluster. From this table, we see that some activities have multiple clusters. Clusters 2 and 3 have the same sets of sensors, but they are differentiated based on their temporal characteristics. Similar is the case with Clusters 5 and 8. Some of the clusters (7, 10, 11 and 12) do not have any temporal regularity in start times; however, the durations of their instances are similar. Note that, some of the activities are overlapped within the same occupancy episode (e.g., ‘load / unload dishwasher’ is sometimes overlapped with either ‘prepare breakfast’ or ‘prepare dinner’). Our system can still identify ‘load / unload dishwasher’ as a separate activity.

Now, we evaluate the improvement in training due to our techniques of merging successive occupancy episodes that belong to the same activity. Firstly, we train on the raw data; we refer to it as *Normal Clustering*. Secondly, in the raw data, we merge successive occupancy episodes that belong to the same activity, and then perform mining and clustering on the changed data; we refer to it as *Clustering after Successive Occupancy Episodes Merging (SOEM)*. Figure 4.3 shows the performance of training under these two configurations for the

Cl. ID	Used Sensors	Mean Start Time (HH:MM)	Std. Start Time (Min.)	Mean Duration (Min.)	Std. Dur. (Min.)	Probable Label
1	{e, f, m}	9:25	70	1.76	1.65	prep. breakfast
2	{a, e, j}	9:12	65	6.59	3.78	prep. breakfast
3	{a, e, j}	18:50	45	30.24	9.25	prep. dinner
4	{d, f, m}	19:20	60	40.38	17.84	prep. dinner
5	{j, m}	19:30	40	20.65	5.62	prep. dinner
6	{f, k}	19:25	45	10.54	3.29	prep. dinner
7	{d, e}			3.98	2.57	get drink
8	{j, m}	21:54	90	1.71	1.28	get snack
9	{a, f}	20:12	150	2.31	0.75	get snack
10	{l}			4.23	2.39	use wash. mach.
11	{f, h}			4.25	1.5	use dishwasher
12	{h, k}			4.76	2.45	use dishwasher

Table 4.4: Set of Clusters for Kitchen for the Kasteren Dataset.

activity ‘Sleep’. The x-axis shows day of month and the y-axis shows the durations (in hours) of the sleeping episodes as detected by the clustering methods and also as recorded by the resident in the ground truth.

From Figure 4.3, we see that for *Normal Clustering*, many instances of the ‘Sleep’ activity are not included in the ‘Sleep’ cluster (2nd, 6th, 7th, 8th, and 25th) as indicated by durations of zero for the corresponding nights. However, *Clustering after Successive Occupancy Episodes Merging (SOEM)* shows that merging successive occupancy episodes in the *Bedroom* improves the performance; four of the five missing instances of ‘Sleep’ are included in the ‘Sleep’ cluster. This is because during some nights, the resident goes to the toilet in the middle of sleep and comes back and sleeps again. If we do not combine the occupancy episodes before and after this toilet visit, then the individual episodes do not have the start time or duration characteristics of ‘Sleep’.

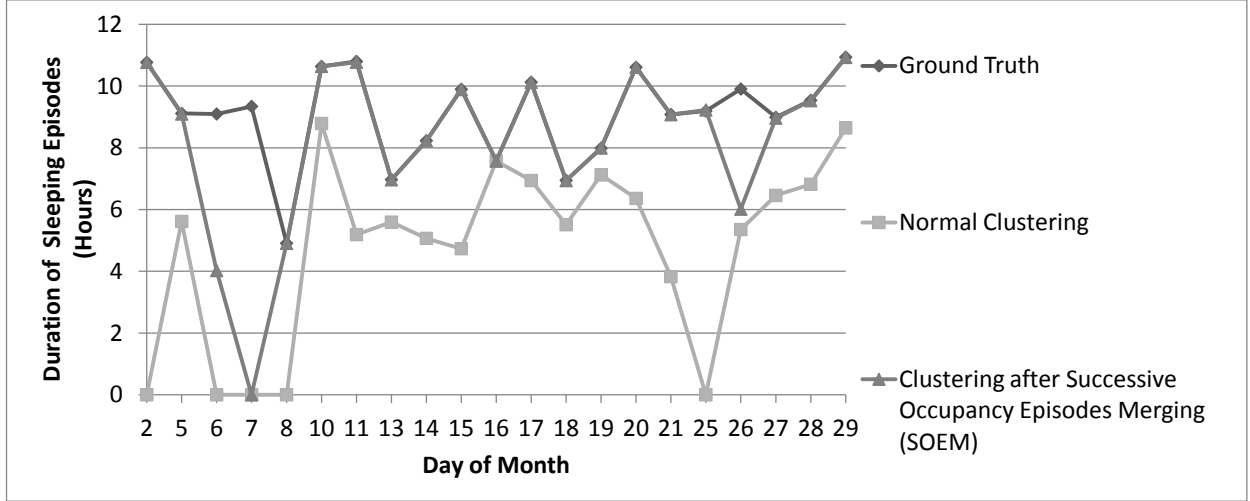


Figure 4.3: Comparison of different instances of the cluster corresponding to ‘Sleep’ with the actual instances of ‘Sleep’ as recorded in the ground truth in the Kasteren dataset.

$$TSE_a = \frac{\text{Total durations of } \mathbf{a} \text{ that remain undetected}}{D_a} \quad (4.1)$$

$$AIE_a = \frac{\text{Total instances of } \mathbf{a} \text{ that remain undetected}}{N_a} \quad (4.2)$$

As evaluation metrics, we use time slice error and activity instance error. For each activity  $a \in A$  ( $A$  is the set of all activities), time slice error  $TSE_a$  is defined in Equation 4.1. Here,  $D_a$  represents sum of durations of all the instances of activity  $a$  in the dataset. Also, activity instance error  $AIE_a$  is defined in Equation 4.2. Here,  $N_a$  represents total number of instances of activity  $a$  in the dataset. Figure 4.4 and 4.5 show the  $TSE_a$  and  $AIE_a$  for each activity  $a$ , respectively. We do not show the training error for the activity ‘Brush Teeth’, because there were no sensors to recognize this activity. From the figures, we see that due to merging successive occupancy episodes, the errors improve only for the activities ‘Sleep’ and ‘Prepare

Dinner’. The reason is that in our dataset, only these two activities have large durations and other activities are overlapped only with them. Compared to *Normal Clustering*, it reduces training time slice error by a maximum of 44% (for ‘Sleep’ activity) and on average by 14% for all activities. Similarly, compared to *Normal Clustering*, it reduces training activity instance error by a maximum of 27% (for ‘Prepare Dinner’ activity) and on average by 10% for all activities.

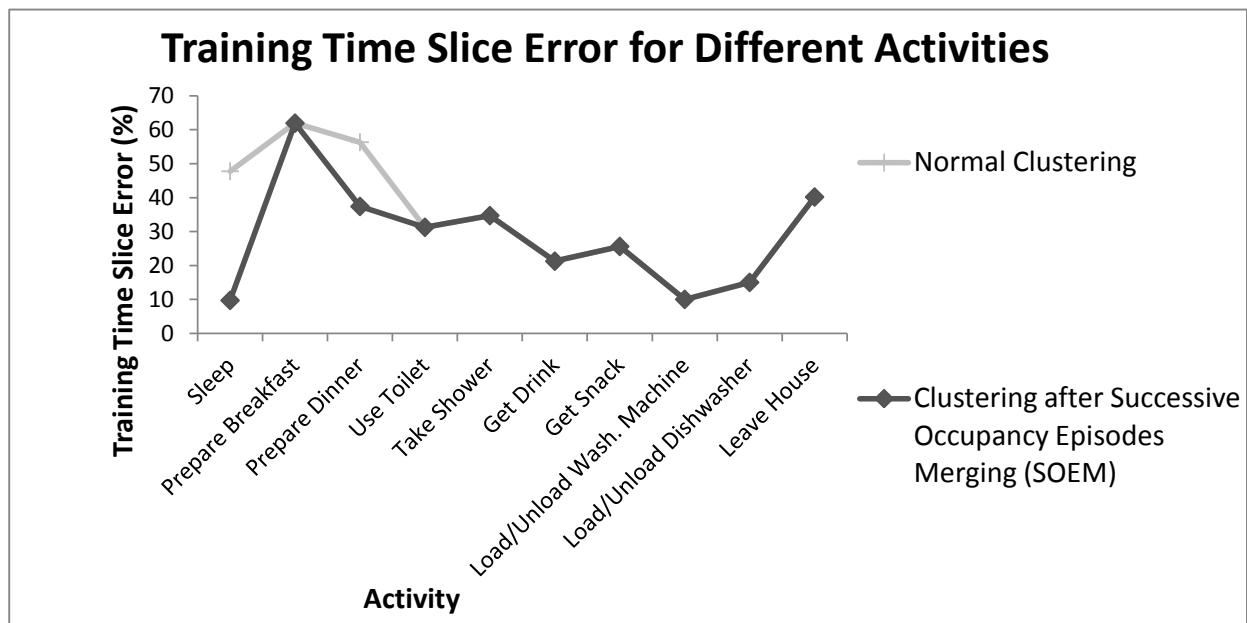


Figure 4.4: Training time slice error for each activity in the Kasteren dataset.

The results presented in this subsection show that *AALO* groups different activities of daily living that have spatial and temporal regularity, and trigger similar group of sensors to fire under different clusters. Moreover, by considering the practical problem of temporally overlapped activities in different rooms, *AALO* improves the performance of clustering. Also, by including set of sensors used as a feature for clustering, *AALO* can differentiate among activities that have similar temporal and spatial regularity, but use different objects.

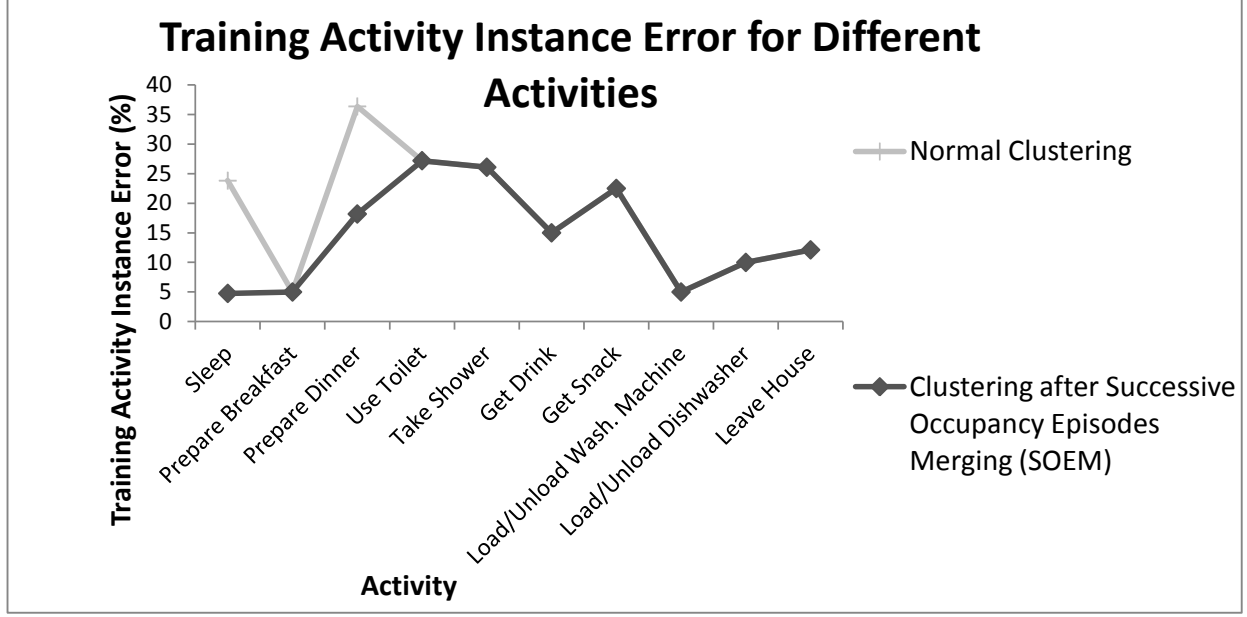


Figure 4.5: Training activity instance error for each activity in the Kasteren dataset.

### Evaluation of Activity Recognition Accuracy

To evaluate the activity recognition performance of *AALO*, we perform leave one day out cross-validation on the 26 days of data in the dataset. The metric for comparison is time slice error  $TSE_a$  for each activity as defined in Equation 4.1. We take the average of the time slice errors from cross-validation. During each step of cross-validation, we train our system with 25 days of data which generates a set of clusters; then we use the trained system on the remaining day to label its room occupancy episodes as instances of the clusters as described in Section 4.4. Thus we get the activity labels and we compare them with the ground truth to calculate the time slice error.

Table 4.5 shows the confusion matrix for all activities (percentage values). The values represent average over all runs in cross-validation. There are two extra columns in the matrix: ‘Idle’ and ‘Irregular’. ‘Irregular’ corresponds to the cases when a frequent itemset is not within the neighborhood radius of any cluster. This may happen when an activity happens in an unusual time or for an unusual duration (compared to the corresponding training set). ‘Idle’

corresponds to the cases when our system infers that no activity is going on.

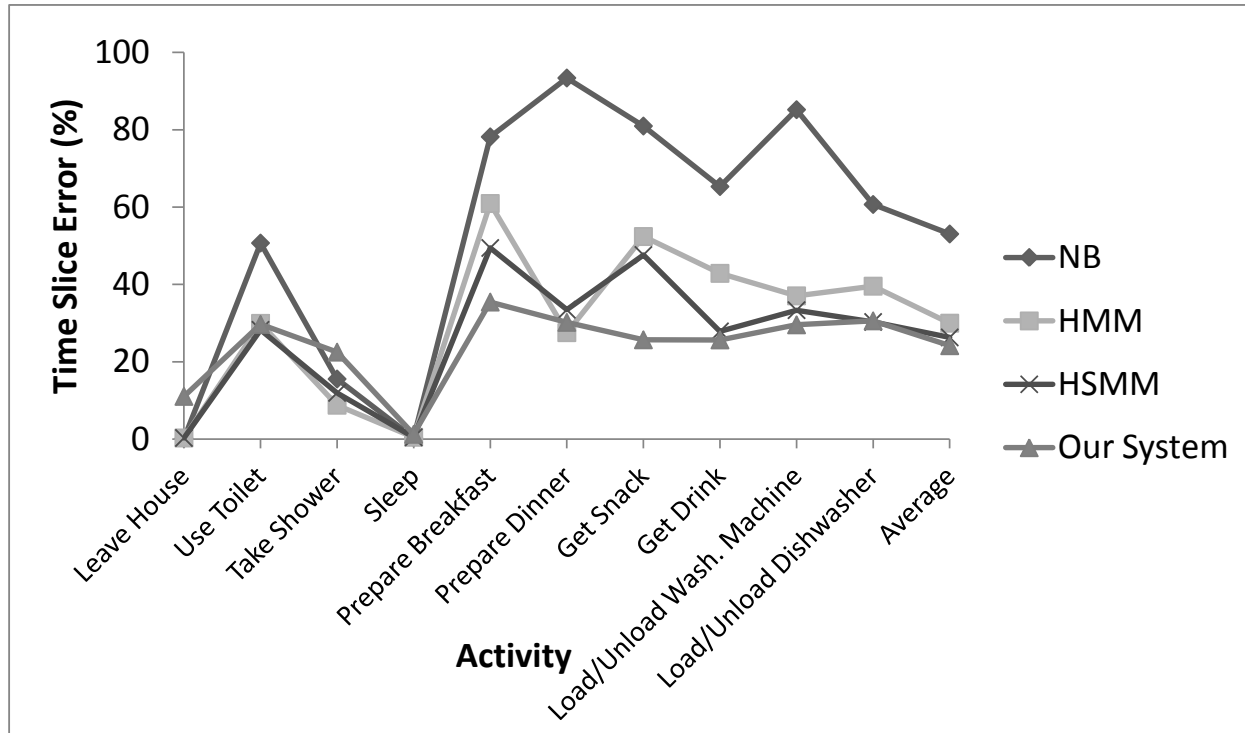
	Leave House	Use Toilet	Take Shower	Sleep	Prep. Bfast	Prep. Dinner	Get Snack	Get Drink	Wash. Mach.	Dish washer	Idle	Irregular
Leave House	89.9	0	0	0	0	0	0	0	0	0	0	10.1
Use Toilet	0	70.3	0	0	0	0	0	0	0	0	14.2	15.5
Take Shower	0	0	77.5	0	0	0	0	0	0	0	4.2	18.3
Sleep	0	0	0	98.7	0	0	0	0	0	0	0	1.3
Prep. Bfast	0	0	0	0	64.5	0	4.2	6.3	0	5.8	10.4	8.8
Prep. Dinner	0	0	0	0	0	69.8	2.4	2.9	0.5	1.3	13.5	9.6
Get Snack	0	0	0	0	0	10	74.3	4.6	0	0	0	11.1
Get Drink	0	0	0	0	11.1	9.4	5.2	74.3	0	0	0	0
Wash. Mach.	0	0	0	0	0	10	0	0	70.4	0	0	19.6
Dishwasher	0	0	0	0	4.6	20.5	0	5.5	0	69.4	0	0

**Table 4.5: Confusion Matrix for all activities (percentage values) for the Kasteren dataset. The rows represent actual activities and columns represent the predicted activities. An entry in row  $x$  and column  $y$  represents percentage of time activity  $x$  was recognized as activity  $y$ .**

As we can see from Table 4.5, many time slices of some activities are mistakenly predicted as ‘Idle’. One reason for this is that we classify sub-periods of occupancy episodes (the duration for which a subset of sensors fire) as activities as opposed to classifying the whole occupancy episode as one activity. Also, in some time slices, according to the ground truth user starts an activity. However, corresponding sensors start to fire after some time slices.

### Comparison with Supervised Activity Recognition Algorithms

We compare the performance of *AALO* with Naive Bayesian (NB), Hidden Markov Model (HMM) and Hidden Semi Markov Model (HSMM) classifiers presented in [5, 8] where the authors divide raw sensor data into fixed length time slots and classify each slot based on the sensors fired within that slot. We set the time slot length of their system to 60 seconds. They convert the raw sensor data to different formats; we compare with the ‘last’ format where once a sensor fires, it is considered to be firing until a different sensor fires. We choose this format, because it gives the highest accuracy for their system. There is another format named ‘changepoint + last’ which gives higher accuracy, but that code is not publicly available. Figure 4.6 shows comparison of the average time slice error of our system with them.



**Figure 4.6: Comparison of Time Slice Error for Activity Recognition among different classifiers for the Kasteren dataset.**

For some activities ('Leave House', 'Take Shower', 'Sleep'), some instances take place in unusual times (e.g., the resident leaves house in midnight, or takes a shower in the evening, or goes to sleep at 3 AM); such instances occur rarely in the entire dataset. Therefore, our system considers these instances as outliers which makes the time slice error higher for these activities. NB, HMM and HSMM perform well for these activities, because they only depend on a single sensor and these algorithms take decisions based on those particular sensor firings ignoring temporal characteristics. NB performs much worse than others, because it classifies each time slot independently without considering the previous activity or durations of the current / previous activity. HMM and HSMM do not perform any preprocessing regarding overlapped activities, still they do not perform much worse than us. This is because they classify each 60 second time slot during which usually there is no overlapping.

Overall, Figure 4.6 shows that the average of time slice errors over all activities for our system (24.15%) is much lower than NB (53.04%), and almost similar to HMM (29.97%) and

HSMM (26.29%). In spite of being an active learning based approach, our system performs as good as the state of the art supervised activity recognition systems. None of these supervised techniques take time of day into account during activity recognition. If they are implemented in a way that time of day is also considered, then their performance should further improve. For example, there would be fewer errors due to confusion between ‘prepare breakfast’ and ‘prepare dinner’. However, *AALO* has the benefit of using significantly fewer number of user-labeled activity instances in the training set.

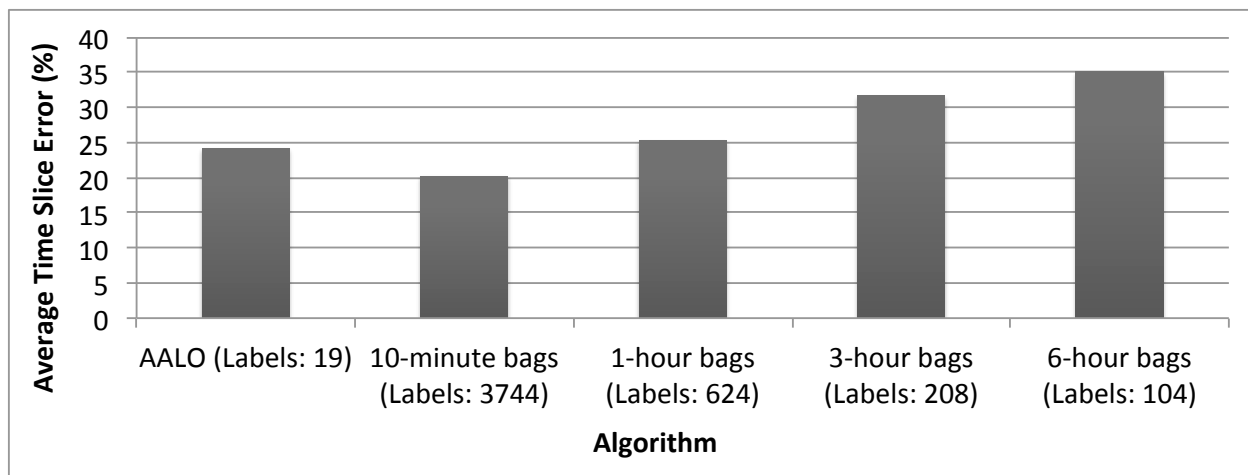


Figure 4.7: Average time slice error for all activities in the Kasteren dataset over all the folds of leave-one-day-out cross-validation for *AALO* and semi-supervised algorithm with different bag sizes (i.e., different amount of labeled ground truth.)

### Comparison with a Semi-Supervised Activity Recognition Algorithm

Next, we compare the performance of *AALO* with a semi-supervised activity recognition algorithm. We compare with the semi-supervised algorithm in [49] that applies multi-instance learning for activity recognition from sparsely labeled data by introducing bags-of-activities. The activity labels do not have to be provided for each data point but rather on a very coarse level; sensor data is grouped into bags and the labels are provided for the bags. The algorithm uses multi-instance SVM (miSVM) for activity recognition. In the multi-instance setting the bag labels provide only partial information about the labels of their comprising

instances. We use the bag sizes of ‘10 minutes’, ‘1 hour’, ‘3 hours’, and ‘6 hours’. A bag size of ‘10 minutes’ require that in every 10 minutes, the list of activities performed during the last 10 minutes have to be specified as ground truth. Similarly a bag size of ‘3 hours’ require that in every 3 hours, the list of activities performed during the last 3 hours have to be specified as ground truth. The bigger the bag size, the less amount of ground truth is necessary.

Figure 4.7 shows the average time slice errors for all activities over all the folds of leave-one-day-out cross-validation for *AALO* and miSVM for different bag sizes. It also shows the amount of labeled ground truth needed for different settings. From this figure we see that *AALO* shows comparable performance with miSVM with bag size ‘10 minutes’. However, miSVM requires ground truth every 10 minutes which adds up to a total of 3744 times over the course of 26 days. Increasing the bag size decreases the amount of user labeling, however it also increases the error in activity recognition. We believe that providing activity labels in every 3 or 6 hours should be acceptable for the residents. *AALO* performs better than miSVM for such bag sizes, and therefore should be considered as a better alternative.

### 4.5.2 Evaluation on Our Deployment

The details of our deployment in a 30-year old male resident’s home are in Section 3.1.4. Here the list of sensors and their symbols are given again in Table 4.6. We collected data from this single-resident apartment for six months continuously. The ground truth for activities are logged by the resident using microphones with the help of *Vocal-Diary*.

#### The Set of Clusters

To demonstrate the different numbers of clusters generated by *AALO*, we train it with the first three months of data. The set of clusters for each room after training is shown in Table 4.7. From the table, we see that there are a total of 29 clusters that the resident has to label. There were a total of 1016 activity instances in these three months. If we use any supervised

Location	Object	Symbol	Location	Object	Sysmbol
Bed Room	Microphone	X1	Living Room	Microphone	X2
Bed Room	Motion	C8	Living Room	TV	T1
Bed Room	Bed	B2	Kitchen	Motion	C2
Bath Room	Sink	F1	Kitchen	Refrigerator	L1
Bath Room	Toilet Flush	O1	Kitchen	Freezer	M1
Bath Room	Shower	E1	Kitchen	Microwave	N1
Bath Room	Motion	C6	Kitchen	Toaster	P1
Living Room	Motion	C4	Kitchen	Plates Cupboard	D1
Living Room	Laptop	L2	Kitchen	Sink	H1
Living Room	Main Door	K1	Kitchen	Spice Cabinet	B1
Living Room	Dining Table	D2	Kitchen	Stove	S1

**Table 4.6: List of Sensors in Our Deployment.**

activity recognition algorithm, the resident has to label all 1016 instances. In this way *AALO* reduces amount of necessary user labeled data significantly.

Room	Number of Clusters
Bed Room	1
Bath Room	5
Living Room	8
Kitchen	13
Outside	2

**Table 4.7: Number of Clusters for Each Room for the Data from Our Deployment.**

Table 4.8 shows the set of clusters for the living room. The activities in the living room includes eating meals, watching the TV and using the laptop. From this table, we see that the resident always has lunch and dinner while watching the TV. Therefore, *AALO* cannot differentiate having lunch or dinner from watching TV. However, *AALO* can recognize all the episodes when the resident watches the TV only. Based on temporal properties, there are different clusters for the activities ‘watch TV’ and ‘use laptop’.

Cl. ID	Used Sensors	Mean Start Time (HH:MM)	Std. Start Time (Min.)	Mean Duration (Min.)	Std. Dur. (Min.)	Probable Label
1	{C4, D2}	9:30	60	6	2	breakfast
2	{C4, D2, T1}	13:38	30	10	4	lunch and TV
3	{C4, D2, T1}	20:10	45	14	5	dinner and TV
4	{C4, T1}	19:50	60	40	25	TV
5	{C4, T1}	23:05	25	60	30	TV
6	{C4, T1}	10:24	70	48	29	TV
7	{C4, T1}	13:32	38	65	32	TV
8	{C4, L2}	20:35	24	72	35	Laptop Use
9	{C4, L2}	18:12	19	55	25	Laptop Use

**Table 4.8: Set of Clusters for Living Room for the Data from Our Deployment.**

### Evaluation of Overlapped Activity Detection

We evaluate the effectiveness of successive occupancy episode merging component of *AALO* in detecting overlapped activities. There are many instances in this dataset when an activity spans across multiple occupancy episodes in a room. We calculate the time slice error in the case when we do normal clustering without merging successive occupancy episodes and in the case when do merge (*AALO*). There are total six months of data. We evaluate using cross-validation approach where in each fold we train with any three full months of data and then test on the remaining three months of data. Therefore, we have a total of 20 folds. We calculate the average time slice error in activity recognition as defined in Equation 4.1 over all the 20 folds for each activity. Figure 4.8 shows the average time slice errors for both approaches.

From Figure 4.8, we see that without successive occupancy episode merging, the normal clustering performs significantly worse for many activities. Specially for the kitchen activities, as they are sometimes interleaved with other occupancy episodes. Successive occupancy episode merging helps *AALO* to merge them and construct occupancy episodes with regular temporal characteristics.

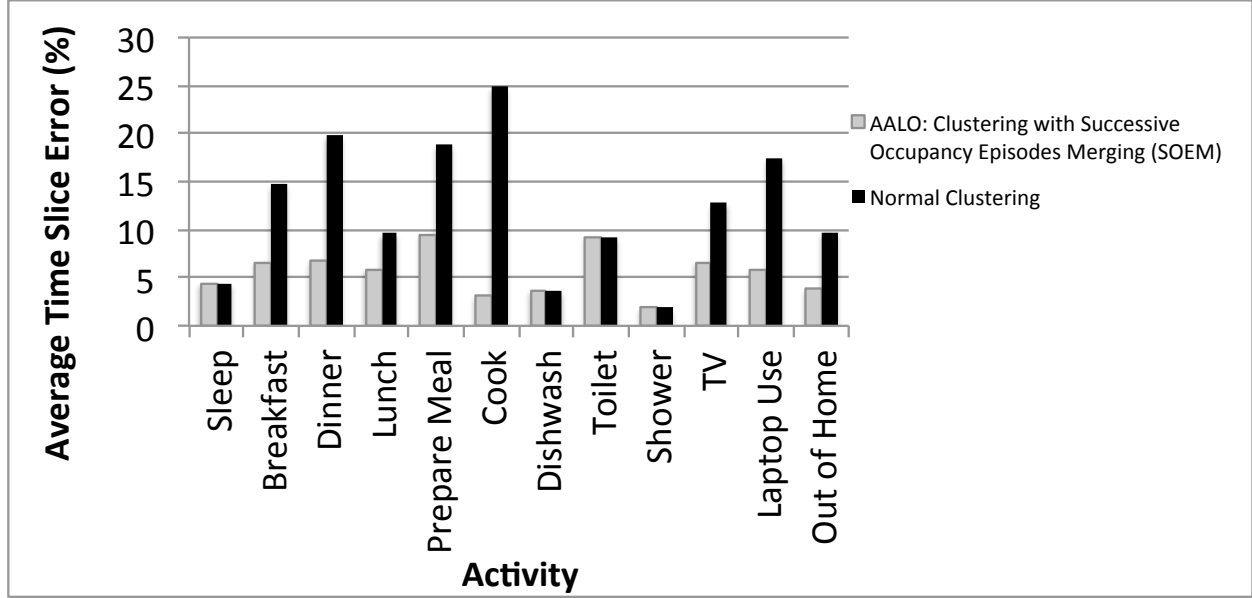


Figure 4.8: Average time slice error for all activities in the data from our deployment over all the folds of cross-validation for normal clustering and for clustering with successive occupancy episode merging (*AALO*).

### Comparison with Supervised Activity Recognition Algorithms

We compare the activity recognition accuracy of *AALO* with Hidden Markov Model (HMM) and Hidden Semi Markov Model (HSMM) based activity recognition algorithms. We collected data for six months. We evaluate using cross-validation approach where in each fold we train with any three full months of data and then test on the remaining three months of data. Therefore, we have a total of 20 folds. We calculate the average time slice error in activity recognition as defined in Equation 4.1 over all the 20 folds for each activity. Figure 4.9 shows the time slice errors for different algorithms.

From Figure 4.9, we see that the time slice error for *AALO* is less than 10% for all activities and almost similar to HSMM for most of the activities. HMM often performs significantly worse than HSMM, as HMM does not consider the duration of stay in a particular state (i.e., activity). HSMM performs very well for activities depending on a single sensor (e.g., ‘watch TV’, ‘laptop use’). This evaluation shows that *AALO* provides comparable performance to HSMM which is a supervised activity recognition algorithm.

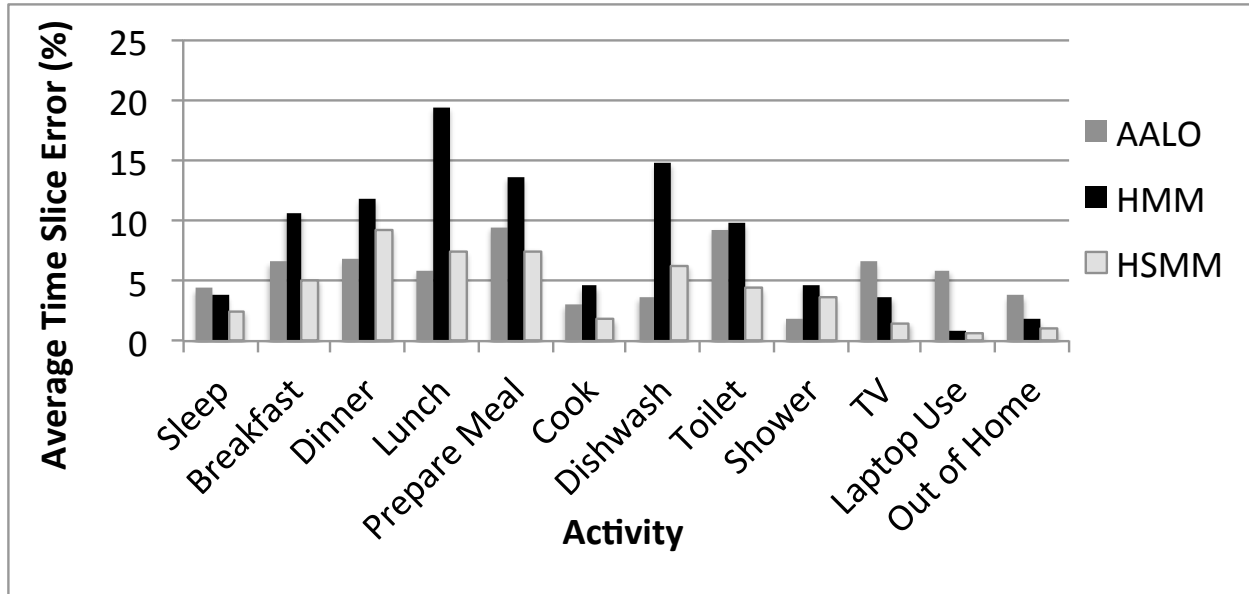


Figure 4.9: Average time slice error for all activities for the data from our deployment over all the folds of cross-validation for the data collected from our deployment.

### Effect of Retraining

We evaluate the effect of retraining on the performance of *AALO*. We train *AALO* with the first two, three and four months of data to test the number of clusters generated for different activities. If there is any change in the temporal or object use regularities for the resident, *AALO* should generate new clusters with additional training data. The results of this experiment is shown in Figure 4.10. From this figure, we see that for some of the activities (e.g., ‘dinner’, ‘lunch’, ‘prepare meal’, ‘TV’), indeed new clusters are generated. Note that in such cases where new clusters are generated, a resident only needs to label the new clusters as activities. In this way, *AALO* facilitates periodic retraining.

Next we test if these new cluster are important in recognizing activities after the first four months. In this experiment, as before we train *AALO* with the first two, three or four months of data and calculate the time slice error for each setting on the remaining two months of data. The results are shown in Figure 4.11. This figure demonstrates that for those activities the periodic retraining generated new clusters, the time slice error reduces

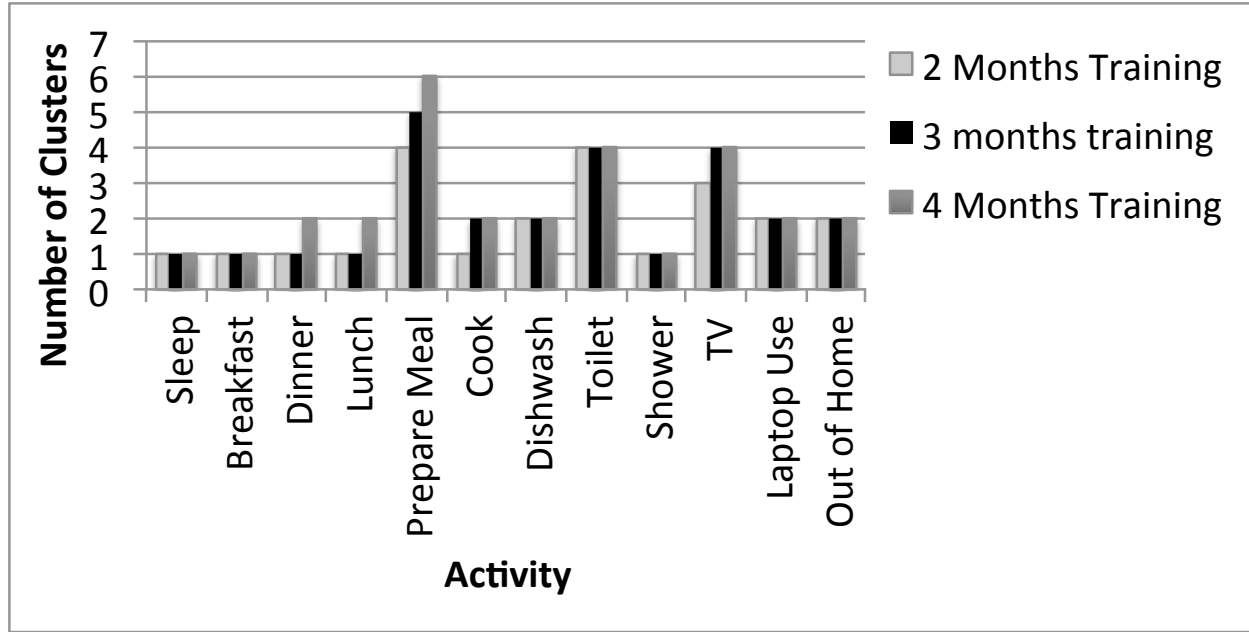


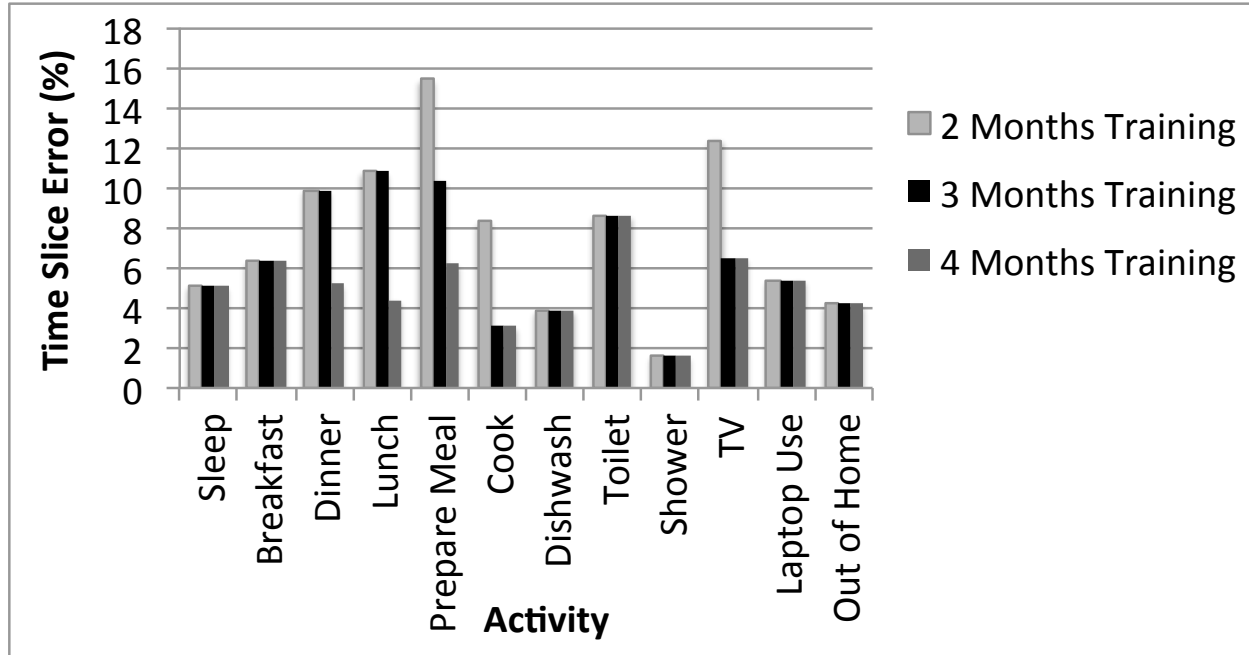
Figure 4.10: Number of clusters for different activities with varying training period for the data from our deployment.

after retraining. Therefore, periodic retraining is necessary to maintain the high accuracy of an activity recognition system.

### Comparison with a Semi-Supervised Activity Recognition Algorithms

As before, we compare the performance of *AALO* with the multi-instance based semi-supervised algorithm miSVM. Here also we do a cross-validation with each three full months of data as training and the remaining data for testing. We calculate the average of time slice error over all the activities over all the folds. We varied the bag size of miSVM in the following range: '10 minutes', '1 hour', '3 hours' and '6 hours'. Figure 4.12 shows the average time slice errors for different settings of miSVM along with that of *AALO*.

From Figure 4.12, we see that the error in activity recognition for miSVM increases with increasing the bag size. Note that amount of user labeled data also reduces with increasing the bag size. Although the '10 minute' bag size outperforms *AALO*, it requires the resident to log ground truth every 10 minutes, either in real time or off line from memory. This is



**Figure 4.11: Effect of retraining on training time slice error for different activities in the data from our deployment.**

not very comfortable and therefore impractical. The bag size of ‘3 hours’ or ‘6 hours’ are reasonable to use, however *AALO* performs better than these two settings.

### 4.5.3 Evaluation on Casas Dataset

Here we present the evaluation results for data from one of the apartments in the Casas dataset that contains data from different sensors deployed in a single-resident apartment along with activity labels (described in Section 3.1.1 as ‘Casas Dataset 1’). The set of sensors in this apartment includes motion sensors and door open / close sensors. There are not many sensors on different objects of daily use which affects the activity recognition accuracy for some activities. We have six months of data from this apartment.

Table 4.9 shows different number of clusters generated for each activity when we train with the first three months of data. As we can see from this table, the activity ‘Sleep’ has three clusters. This is due to the fact that the resident sometimes gets up from sleep in the middle of the night for toilet visits, and then goes back to sleep. Because it happens in a

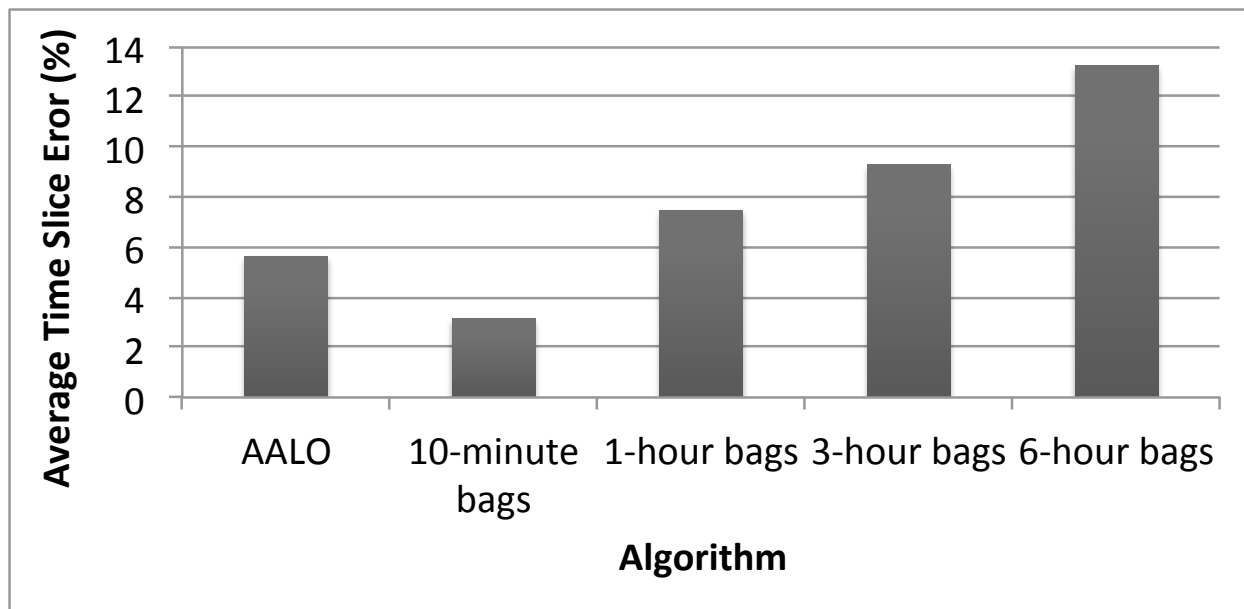


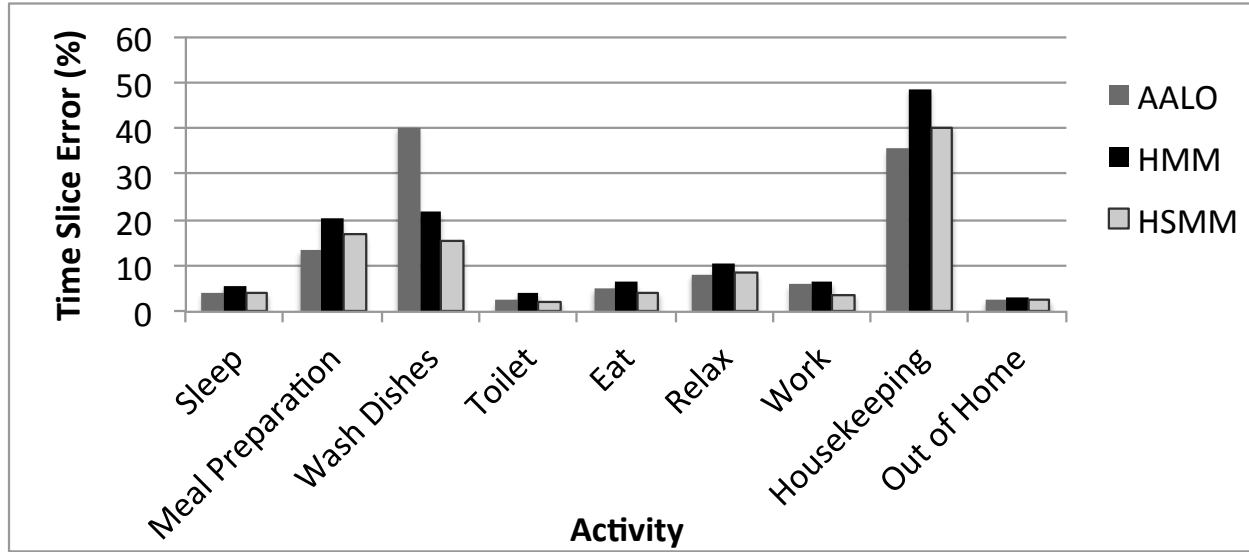
Figure 4.12: Comparison of time slice error of *AALO* with miSVM having different bag sizes for the data from our deployment.

Activity	Number of Clusters	Activity	Number of Clusters
Sleep	3	Relax	4
Meal Preparation	4	Work	4
Wash Dishes	1	Housekeeping	1
Eat	3	Out of Home	3
Toilet	6		

Table 4.9: Number of clusters for different activities for ‘Casas Dataset 1’.

significant number of nights, the ‘successive occupancy episode merging’ component of *AALO* does not merge these two fragmented sleep episodes for such nights. As a result, there is one cluster for the sleep episodes before toilet visits in the middle of the night, one cluster for the sleep episodes after the toilet visits, along with one cluster for the nights when the resident sleeps without any interruption. ‘Out of Home’ also has three clusters, one representing episodes of very little duration when the resident goes out (few minutes), one representing outside visits lasting few hours, and one representing longer outside visits.

We compare the activity recognition performance of *AALO* with HMM and HSMM. As before, with the six months of data we do cross-validation where in each fold there is three

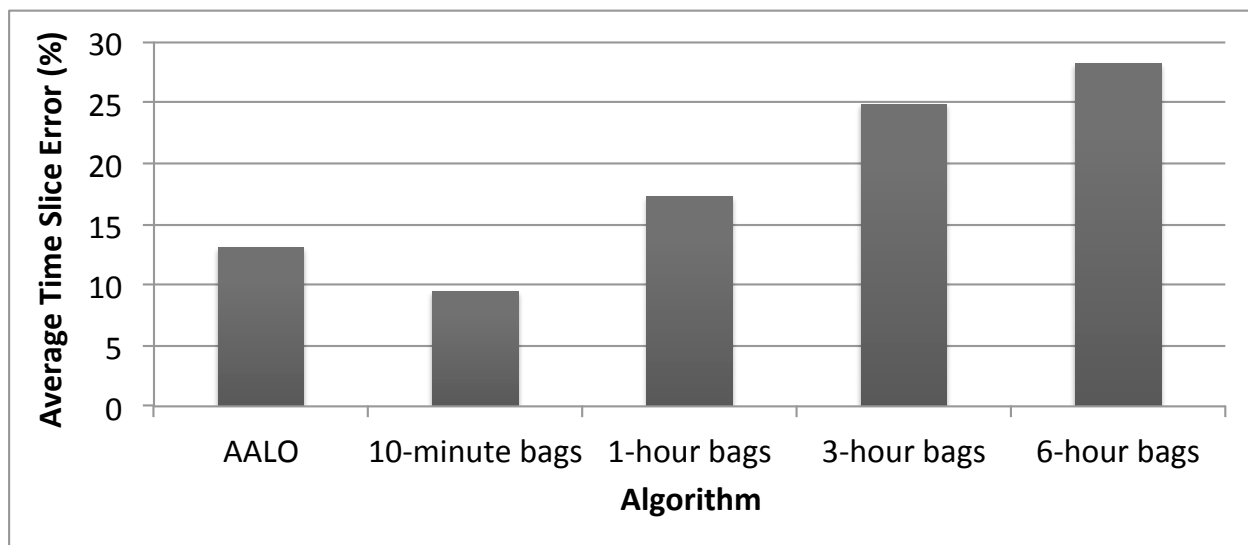


**Figure 4.13:** Comparison of average time slice errors over all the folds of cross-validation of *AALO* with HMM and HSMM for different activities for ‘Casas Dataset 1’.

full months of data, and the remaining three months serve as testing data. Figure 4.13 shows the average time slice errors over all the folds for each activity. The figure shows us that *AALO* has almost same amount of error as HSMM for most of the activities. However, one exception is the activity ‘Wash Dishes’. As we mentioned earlier, there are only motion sensors and door open / close sensors in this dataset. There is no specific sensor to detect dish washing. Therefore, *AALO* has only temporal properties for recognizing this activity which does not perform very well. HMM and HSMM perform significantly better for this activity. The reason is that ‘Wash Dishes’ always happens after the activity ‘Eat’. Because HMM and HSMM can learn this relationship, they can recognize this activity better. From this dataset, we learn this limitation of *AALO* that without any object use and temporal regularity, an activity may not be accurately recognized, whereas algorithms like HMM and HSMM may exploit other properties such as the previous activity. However, for that ground truth for each activity instance is necessary.

Another activity for which no algorithm performs well is ‘Housekeeping’. This is because this activity spans across multiple rooms. The resident roams around in multiple rooms

during this activity. *AALO* performs slightly better than the other two algorithms, this is because this activity always involves one of the motion sensors (the motion sensor in a closet) which does not fire for any other activity. *AALO* can learn this object use regularity automatically. However, when the resident moves to other rooms, *AALO* also fails. This exposes another limitation of *AALO* which is not recognizing activities that span across multiple rooms and when the resident does not use any specific sensor to do such activities.



**Figure 4.14:** Average time slice errors over all activities over all the folds of cross-validation for *AALO* and miSVM with different bag sizes for ‘Casas Dataset 1’.

The comparison of *AALO* with the multi-instance based semi-supervised activity recognition algorithm, miSVM, for different bag sizes is shown in Figure 4.14. Here we did a cross-validation with the six months of data where in each fold, there are three full months data for training and the remaining for testing. With a bag size of 10 minutes, miSVM performs better than *AALO*, specially for the activity ‘Wash Dishes’. As before, the performance of miSVM worsens with increased bag sizes, i.e., with less frequent labeling of ground truth. For the bag sizes of ‘3 hours’ and ‘6 hours’, which may be considered as comfortable for residents, miSVM performs worse than *AALO*.

#### 4.5.4 Summary of Evaluation

In summary, we have shown evaluation results of comparing the performance of *AALO* with state of the art supervised and semi-supervised activity recognition algorithms. The evaluation dataset consists of public data from two single resident homes (one of whom has six months of data), and data from one single-resident apartment where we deployed our system for six months. We also evaluate the effect of our successive occupancy episodes merging technique on the performance of *AALO*. Evaluation results show that *AALO* performs as well as the state of the art supervised activity recognition algorithms (e.g., HSMM) and performs better than the multi-instance based semi-supervised activity recognition algorithm with settings that are practical (i.e., ‘3 hour’ or ‘6 hour’ bag sizes). This performance of *AALO* is achieved with minimal amount of user labeled ground truth which also facilitates periodic retraining.

## 4.6 Conclusion

*AALO* ensures ease of use in practical deployments, because it does not need ground truth for all instances of all activities during training. We address overlapping activities in two ways: 1) in the preprocessing step, we identify if an activity consists of multiple occupancy episodes with another activity overlapped in between them; and 2) in the itemset mining step, we identify multiple activities temporally overlapped in the same occupancy episode. The performance of our system which handles overlapped activities is as good as the state of the art supervised activity recognition systems (HMM, HSMM) and is better than the multi-instance based semi-supervised activity recognition algorithm with settings that are practical.

# Chapter 5

## Semantic Anomaly Detection

In this chapter, we present *Holmes*, a comprehensive anomaly detection system for daily in-home activities. *Holmes* accurately learns a resident’s regular behavior by considering normal variability in daily activities based on specific days of the week, combining activity instances of a day / multiple days together to find the features that best represent regularity, and by detecting temporal and causal correlations among multiple activities. Also, based on resident and expert feedback, *Holmes* learns semantic rules that explain specific variations in daily activities in specific scenarios. Together the comprehensive modeling of regular behavior and the set of semantic rules help *Holmes* to reduce false alarms which is essential for the reliability of any anomaly detection system.

The rest of the chapter is organized as follows. In Section 5.1, we discuss the motivations behind *Holmes*. We list the contributions of *Holmes* in Section 5.2. Section 5.3 explains different types of anomalies that can be detected using *Holmes*. Section 5.4 describes the architecture of *Holmes* and different components of the system. Following that, Section 5.5 presents the details of experiments and evaluation results. Then we conclude in Section 5.6.

## 5.1 Motivation

Due to increasing numbers of elderly people living alone, wide-spread ubiquitous deployment of sensors has become prominent for monitoring in-home activities of daily living (e.g., eating, sleeping, exercising, and entertainment). Research in accurate detection and summarization of these daily activities has progressed significantly over the last decade [5, 19, 26, 50, 12] which enables long-term monitoring of a resident's in-home activities, learning normal behavior, and detecting deviation from normal behavior i.e., anomalies. Detecting anomalies in daily in-home activities is the most important component of many home health care applications such as assessing behavioral rhythms[2, 3], and monitoring cognitive decline [4, 16].

An anomaly detection system needs to model a resident's regular behavior accurately, and define types of deviation from the model that would be considered anomalies. Regular behavior consists of what in-home activities a resident performs, which time of day each activity takes place, how long an activity lasts, and relationships among different activities. Regular behavior may depend on day of the week, season, and other events of a resident's life (e.g., special occasions, visitors, medications). Anomalies can be classified as point, collective and contextual anomalies [56]. In the domain of in-home activities, point anomalies consider each instance of each activity independently and decide whether that instance is normal or not. Collective anomalies consider groups of activity instances together to determine whether the group is normal or not. Contextual anomalies consider activities under some context (e.g., day of week, under medication). Since human behavior is complex, an anomaly detection system for in-home activities should be designed to address these multiple types of anomalies. Otherwise, an anomaly detection system would become unreliable due to false positives and false negatives.

There exist many [13, 14, 15, 16, 17, 18] anomaly detection algorithms for monitoring behavioral anomalies. They either consider each activity instance as an independent data point (point anomalies) or consider sequence of activities together (collective anomalies). However, none of the existing algorithms combines context with point and collective anomalies.

Consequently, they cannot find the variability in resident's behavior in different days of week if it exists. To address this, an anomaly detection system needs to consider different models for different days of the week and automatically merge those that are similar.

Regular behavior may consist of different features for different in-home activities. For some activities, features generated from individual activity instances may be important. However, for some other activities, we may need to combine multiple activity instances from a day / week / some other period to construct useful features that represent regularity. However, existing anomaly detection systems do not address this issue; they use features from individual activity instances. Moreover, anomaly in an activity may be caused by a previous activity on which the later activity is temporally dependent on. In such cases, reporting anomalies for the later activity would generate false alarms. Also, a resident's behavior may also be affected by different explainable scenarios (e.g., new medication, special occasions, visitors). We argue that there must be a mechanism in an anomaly detection system that can learn such explainable scenarios as semantic rules from a resident's feedback. These semantic rules are important to reduce false alarms generated by the system that would make it unreliable.

## 5.2 Contributions

We present *Holmes*, a comprehensive anomaly detection system for daily in-home activities. *Holmes* learns a resident's normal behavior in terms of in-home activities from training data. Note that based on the training data, each activity may have multiple models as part of normal behavior (e.g., weekdays, weekends, Fridays). *Holmes* automatically learns these multiple models for each activity using a novel context-aware (i.e., day of week) hierarchical clustering algorithm. *Holmes* also learns temporal (e.g., if two activities often happen concurrently, if one activity often happens before another) and causal (e.g., if duration of one activity depends on another activity) relationships among multiple activities using sequential pattern

mining and itemset mining algorithms. A highly accurate normal behavior assessment system forms the critical base upon which to detect anomalies.

*Holmes* is also designed to use semantic rules that define logical deviations from regular behavior. *Holmes* starts with an initial set of predefined rules defined from domain knowledge. As the system runs, newly detected anomalies are verified by the resident / experts to be included as new rules if appropriate. If there is repeated occurrence of one specific scenario (i.e., semantic rule), *Holmes* trains models for that particular scenario for future use; thus minimizing or even eliminating unnecessary false alarms.

The main contributions of *Holmes* are:

- A total system composed of a comprehensive and multi-model system for modeling regular resident behaviors carefully integrated with a sophisticated multi-level and semantic anomaly detection system.
- The novelty in modeling regular behavior includes the collection of following features: hierarchically merging clusters from different days of the week ensuring that the merged clusters do not become too generalized compared to the original ones, preservation of noise points found in training for future use during retraining, the use of a combination of features extracted from both individual activity instances and group of activity instances combined over a specific period, and use of sequential pattern mining and itemset mining algorithms to learn groups and sequences of activities that are part of a resident's regular behavior (i.e., temporal and causal correlations among activities).
- The novelty in the anomaly detection arises due to the combination of features that include: combining point, collective and context anomalies to ensure reliability, use of semantic rules that represent logical deviation from regular behavior to reduce false alarms, and learning new semantic rules based on resident / expert feedback.
- Evaluation of *Holmes* using 1) six months of activity data collected from one single-resident home; 2) User-labeled in-home activity data from three apartments, each of

which has at least four months of data, from a publicly available dataset; 3) one data set from *BeClose* (a commercial senior safety system provider [23]) that contains four months of activity data from deployment of the *BeClose* system in one single-resident home; and 4) activity data collected by a online survey from 11 participants for two months each. Our evaluation shows that *Holmes* reduces false positives by at least 46% and false negatives by at least 27% compared to three state of the art anomaly detection systems. Also, the use of semantic rules reduces the number of false positives by an additional 20% for *Holmes*.

### 5.3 Types of Anomalies

In this section, we give an overview of different kinds of anomalies in daily activities that can be detected using *Holmes* along with corresponding examples. Figure 5.1 shows that anomalies can be mainly divided into point and collective anomalies. Both point and collective anomalies and all their sub-types fall under context anomalies, this is because all these kinds of anomalies may be normal in some contexts, while they may be anomalies in other contexts. For example, it may be normal for the resident to be in home all day during weekends. However, it may not be normal for the residents to be home all day during weekdays; here, day of the week is the context. Therefore, we always need to consider the context while detecting any of these anomalies.

The first step of any anomaly detection system is to learn the normal behavior based on training data. Once regular behavior is modeled and labeled, an anomaly detection system can detect any deviation from the normal behavior that is represented by and learned from the training data. Behavior is defined by the activities detectable by the underlying sensing system. Therefore, the kinds of anomalies that can be detected by *Holmes* are dependent on the training data. If an activity did not take place during the training period, then any anomaly in that activity during the testing period can not be detected. Similarly, if any

feature of an activity is not available in the training data, then *Holmes* can not detect any anomaly for that activity in that particular feature.

Note that different applications may need to monitor for a different subset of anomalies that *Holmes* can detect. It is outside the scope of this dissertation to propose or evaluate what anomalies need to be detected for different applications. *Holmes* can detect the kinds of anomalies presented in this section. It is up to the domain experts to decide what anomalies they want to detect for their applications. Now, we discuss the different sub-types of both point and collective anomalies.

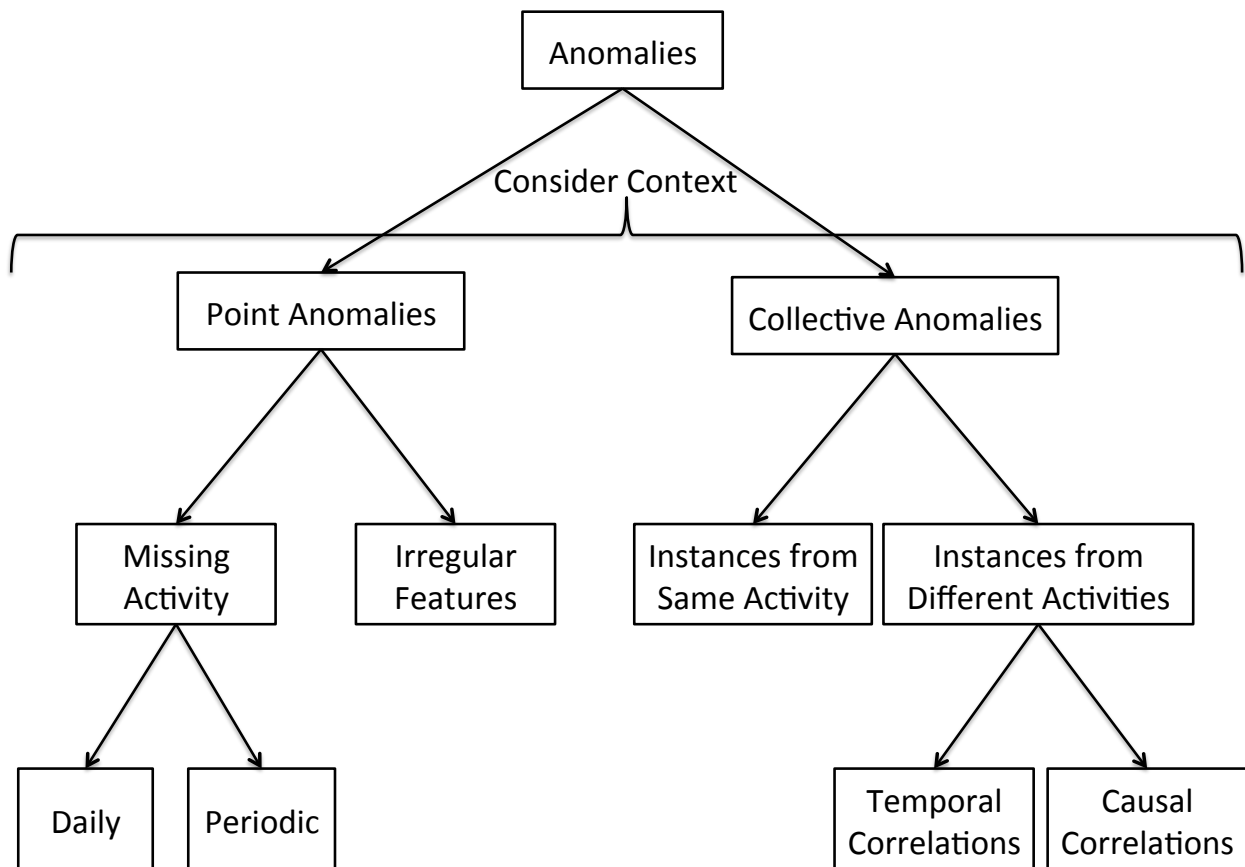


Figure 5.1: Different types of anomalies that can be detected by *Holmes*.

### 5.3.1 Point Anomalies

Figure 5.1 shows that two types of point anomalies can be detected using *Holmes*. Firstly, whenever an activity takes place, that instance of the activity may be normal or an anomaly based on the features used to model that activity during training. For example, when a resident has dinner, that dinner episode may be an anomaly based on the time of the day, the duration of the dinner, or any other feature. A new activity instance is considered to be anomaly if it is more than two standard deviation away from the mean of the cluster that represents the corresponding activity (see Section 5.4.4 for details). As pointed out earlier, whether an individual activity instance is a point anomaly or not may also depend on the context.

The other kind of point anomalies occur when an activity does not take place when it should, according to the training. For example, if during training, an activity always takes place daily, then its absence on a particular day should be considered as an anomaly. Anomalies such as these based on missing activities can be detected by *Holmes*. There can also be periodic activities, that do not take place daily. They always takes place after a certain period in the training data. For example, a resident may often cook after every 2 – 3 days during training. In testing data, if the resident does not cook in more than 3 days, then that may be considered as an anomaly. The two sub-types under ‘Missing Activity’ in Figure 5.1, i.e., ‘Daily’ and ‘Periodic’ represent these two cases. Note that, even if there is no ‘Missing Activity’ anomaly for an activity in a day, there may be anomaly for that activity that day based on irregular features. Also, whether absence of an activity in a specific period is normal or an anomaly may also depend on the context.

### 5.3.2 Collective Anomalies

In case of collective anomalies, multiple activity instances, from the same activity or from multiple activities, are combined and this collection is labeled as normal or an anomaly. In case of ‘Instances from Same Activity’, activity instances from the same activity are combined

for a specific period, and features are extracted from the collections to train and learn regular behavior. For example, if we combine all the toilet visits in a day, and the features extracted are total number of toilet visits, then significant variation from the training data in the total number of toilet visits each day can be considered as an anomaly. A collection of activity instances from the same activity is considered to be an anomaly if the features extracted from that collection falls outside two standard deviation of the cluster that represents the corresponding activity collection (see Section 5.4.4 for details). Note that all activity instances over a day should be combined only for activities that happen more than once a day. If an activity happens at most once a day, then only considering the point anomalies for that activity is sufficient. However, that activity can still cause a collective anomaly where we combine activity instances from multiple activities. *Holmes* can also be configured to combine activity instances over a week, or over a month for a particular activity, and monitor for anomalies in the features extracted from such collections.

For the other kind of collective anomaly, i.e., in case of ‘Instances from Multiple Activities’, instances from multiple activities are combined from training data to find any temporal or causal correlations among different activities. Temporal correlations exist among a set of activities if they always happen in a specific order, i.e., they constitute a sequence, and / or if there is regular time interval among the successive activities in the sequence. For example, in the training data, a resident may often use laptop before having dinner, or a resident may often watch TV while having breakfast. During testing, if such an order is broken, then it may considered as an anomaly. *Holmes* can be trained to detect such temporal correlation based anomalies. Causal correlations among activities exist when the occurrence of one / more activities are often accompanied by the occurrence of one / more other activities in the training data. *Holmes* can learn such correlations from the training data and detect such anomalies in testing data. Note that all kinds of collective anomalies may depend on the context.

### 5.3.3 Illustrative Example

As part of the evaluation of *Holmes*, we deployed our system in a single-resident apartment and collected data for six months. In this section, we discuss different anomalies of Figure 5.1 that were detected in this deployment, when we train *Holmes* with the first three months of data and test on the last three months of data. Details of these anomalies are presented in Section 5.5. When we train *Holmes* with the first three months of data from this deployment, the set of activities in the training data included ‘Sleep’, ‘Lunch’, ‘Breakfast’, ‘Out of Home’, ‘Cook’, ‘Prepare Meal’, ‘Shower’, ‘Dinner’, ‘Dishwash’, ‘Laptop Use’, ‘Toilet’, ‘Snack’, and ‘TV’. After training, we look for anomalies in these activities during the remaining three months of data.

In the three months of testing data, a total of 38 point anomalies were detected which were due to some activities not taking place in specific days, and some activity instances having irregular features. In addition, a total of 16 collective anomalies of type ‘Activity Instances from Same Day’ were detected because total count and / or total duration features of some collection of activity instances for some activities were outside two standard deviation from the mean of the corresponding clusters. Also, our system detected 5 anomalies based on violation of the temporal and causal correlations among different activities that were learned during training. Note that, if any application needs to monitor for only a subset of these activities, or a subset of anomaly types for different activities, then *Holmes* can be configured accordingly.

## 5.4 System Description

In this section, we first formulate the problem with notations that are used throughout the rest of the section. Then, we present the end-to-end system architecture of *Holmes* which is followed by in-depth explanation of the important components of our system.

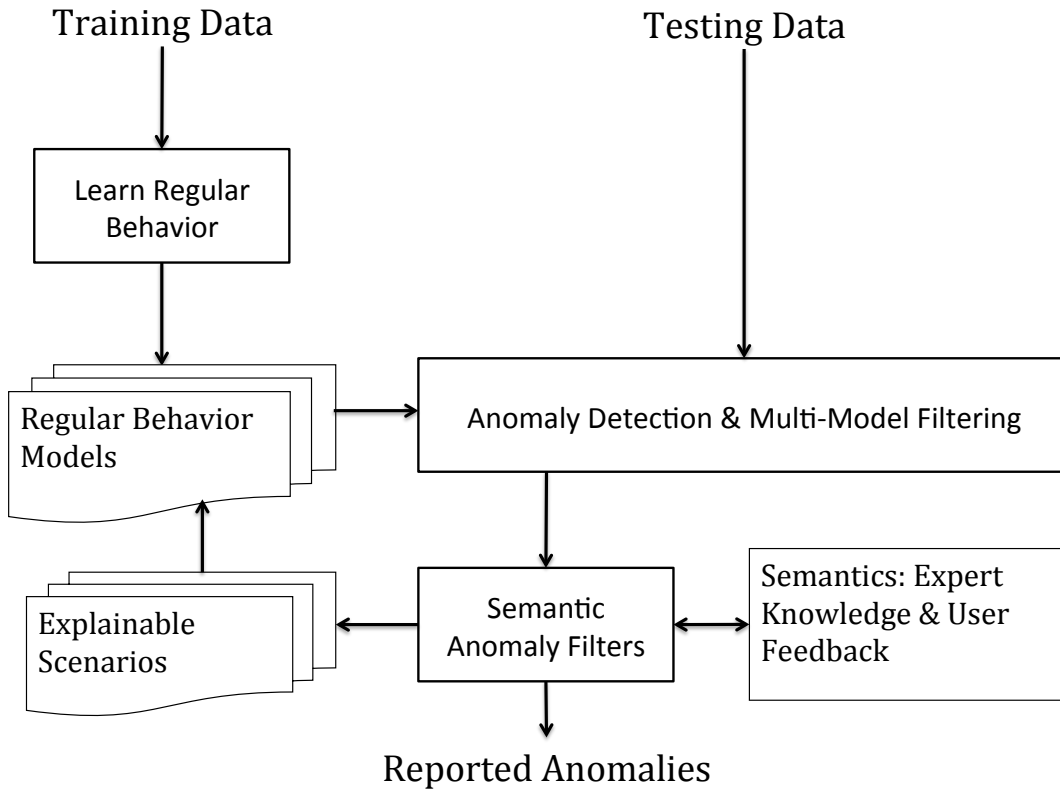
### 5.4.1 Problem Formulation

In this work, we focus on in-home activities of daily living e.g., sleeping, eating (breakfast / lunch / dinner / snack), meal preparation, toileting, showering, entertainment (watching TV), and laptop use. Whenever the resident performs an activity, we get an instance for that activity. An activity instance is represented by an activity identifier, the time when the activity starts, and the total duration of that activity; we use the notation (*activityID*, *startTime*, *duration*) for this representation. The start time and duration of each activity instance may be logged by the user, or may be detected by an activity recognition system based on in-home sensors.

The research problem we address in this work is how to learn a resident’s regular behavior from training data containing activity instances and monitor new activity instances to find their deviation from the learned regular behavior i.e., to find anomalies. Behavior is modeled not just by simply considering each activity instance in isolation. Rather, we model behavior by considering the context (day of week) of each activity, and the complex temporal and causal interactions among. Note that *Holmes* does not aim to detect emergencies (e.g., fall detection) that require immediate attention.

### 5.4.2 *Holmes* Framework

Figure 5.2 shows the *Holmes* framework for anomaly detection. The training and testing data consist of sets of activity instances. The function of the ‘Learn Regular Behavior’ component of *Holmes* (Section 5.4.3) is learning a resident’s regular behavior from the training data. Since human behaviors are complex, this results in multiple regular behavior models. Based on the learned models, the ‘Anomaly Detection and Multi-Model Filtering’ module (Section 5.4.4) detects anomalies in the activity instances of the testing data first. Next, this module filters out those detected anomalies that do not satisfy the anomaly conditions in all the corresponding models learned during the ‘Learn Regular Behavior’



**Figure 5.2: *Holmes* Framework for Anomaly Detection**

module. Only the anomalies that pass the filtering are forwarded to the ‘Semantic Anomaly Filters’ module.

The ‘Semantic Anomaly Filters’ module, described in Section 5.4.5, filters out those anomalies that may be explained by ‘Expert Knowledge & User Feedback’ i.e., semantics. For example, the user may be recovering from a major operation, or there may be a power outage which caused loss of sensor data. This module detects such scenarios and does not report them as behavioral anomalies. Such scenarios are saved in ‘Explainable Scenarios’. If there is significant number of instances of a particular scenario, a new behavioral model is developed for this scenario and stored in the ‘Regular Behavior Models’ so that in the future such scenarios are not detected as anomalies. Thus the set of regular behavior models

is continuously enriched.

### 5.4.3 Learn Regular Behavior

First, we explain how each activity is modeled. Then we describe how temporal and causal correlations among activities are identified. The result is a set of models representing complex regular behavior in terms of daily activities.

#### Modeling Each Individual Activity

One part of learning complex behavior is to model regularities in each individual activity. For example, when does the resident usually go to bed, duration of sleep, how many times a week the resident goes to the gym, whether the resident usually has lunch at home: all this information may be indicative of regular behavior and deviations from that may indicate change of lifestyle or may be due to some illness. We propose a novel context-aware hierarchical clustering algorithm to model each individual activity. The context we consider here is ‘day of the week’. Our hypothesis is that people may have different routines for different days of the week; no existing algorithms address this variability. We model each day’s behavior (e.g., Sunday, Monday) separately and then merge models for multiple days (e.g, merge Friday and Saturday together) if they are very similar. If the resident indeed behaves differently in different days of the week, then *Holmes* can identify it. Before describing the clustering algorithm, we explain the feature set which is an important part of any clustering algorithm. If adequate data is available, the algorithm can also handle other context and model behavior accordingly e.g., for different seasons, for 4th of July, for Thanksgiving.

**[Feature Selection]** For modeling each activity instance, we use the features *startTime* and *duration*; we cluster based on these two features. If any additional feature is available for an activity, then that can also be incorporated in the feature set. For example, sleep can have an additional feature named *sleepQuality*, breakfast can have an additional feature *calories*. We do not have these features in our data sets. For each activity, we also combine all activity

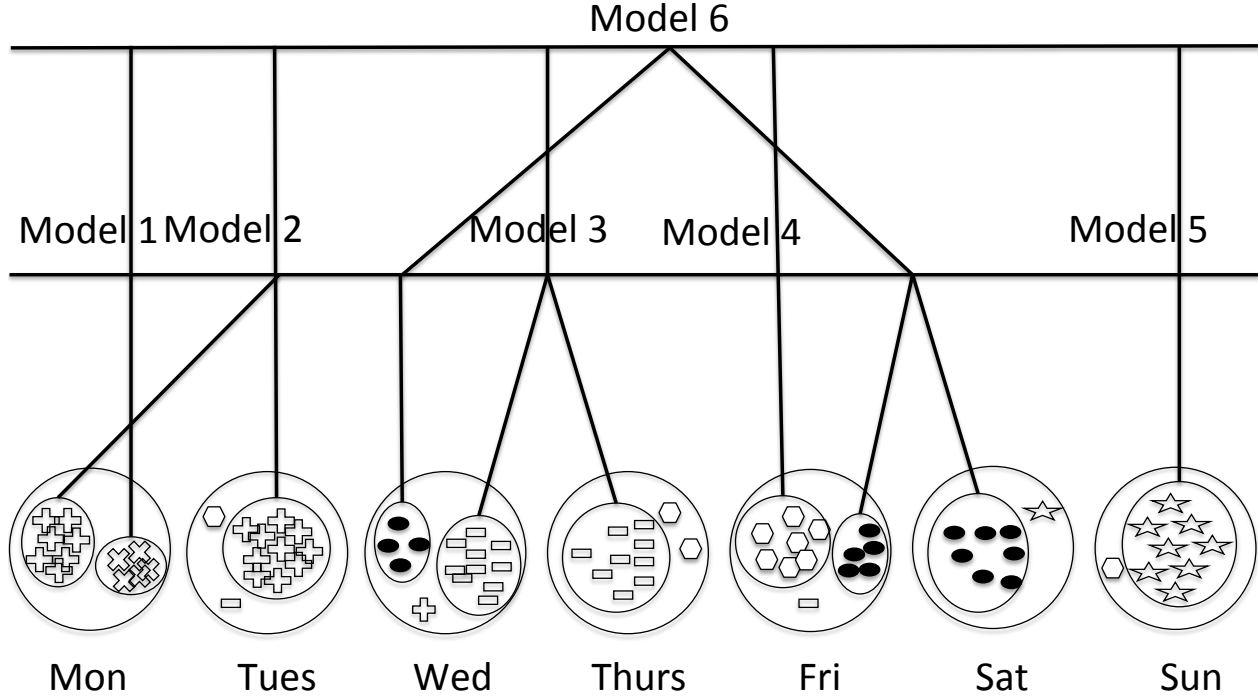
instances in a day and calculate the following two features: 1) *totalCountPerDay* represents how many times an activity takes place in a day 2) *totalDurationPerDay* represents the total duration of an activity in a day (the sum of durations of all activity instances). Note that we also consider the days when an activity does not happen (i.e., *totalCountPerDay* is zero). As a result, if an activity does not take place often in a specific day, then that is also represented by a cluster.

We do not use a classification based approach for modeling regular behavior, because it is difficult to get data for abnormal behavior for training. We use a clustering based approach so that we can model how the resident behaves normally, and after training any behavior that is significantly different from the normal behavior is considered as an anomaly (Section 5.4.4).

**[Context-aware Hierarchical Clustering Algorithm]** Figure 5.3 explains how our context-aware clustering algorithm works in a bottom-up hierarchical way. The novelty of this algorithm includes merging clusters from different days of the week hierarchically ensuring that the merged clusters do not become too generalized compared to the original ones, preservation of the noise points for future use during retraining, and use of combination of features extracted from both individual activity instances and group of activity instances combined over a specific period. Note that the algorithm runs separately for each activity.

At the bottom layer, for each day of the week, the algorithm combines the activity instances of an activity and clusters them based on their features. When we model the activity instances independently, the features are *startTime* and *duration*. When we model the collection of all activity instances of a day, the features are *totalCountPerDay* and *totalDurationPerDay*. For clustering in the bottom layer, we use the DBSCAN clustering algorithm [75] which is a density based clustering algorithm. The major advantage of DBSCAN is that we do not need to specify how many clusters there are. This is important, because we do not know how many different activities there are. DBSCAN has two parameters; one is *min\_pts* which is the minimum number of points in a cluster, and the other is *Eps* which is the minimum distance

between two data points for them to be considered in the same cluster. We only specify the parameter  $min\_pts$  and use the corresponding value of  $Eps$  that DBSCAN suggests. After clustering, DBSCAN marks each point as belonging to a cluster or as noise.



**Figure 5.3: Per-activity Context-aware Hierarchical Clustering**

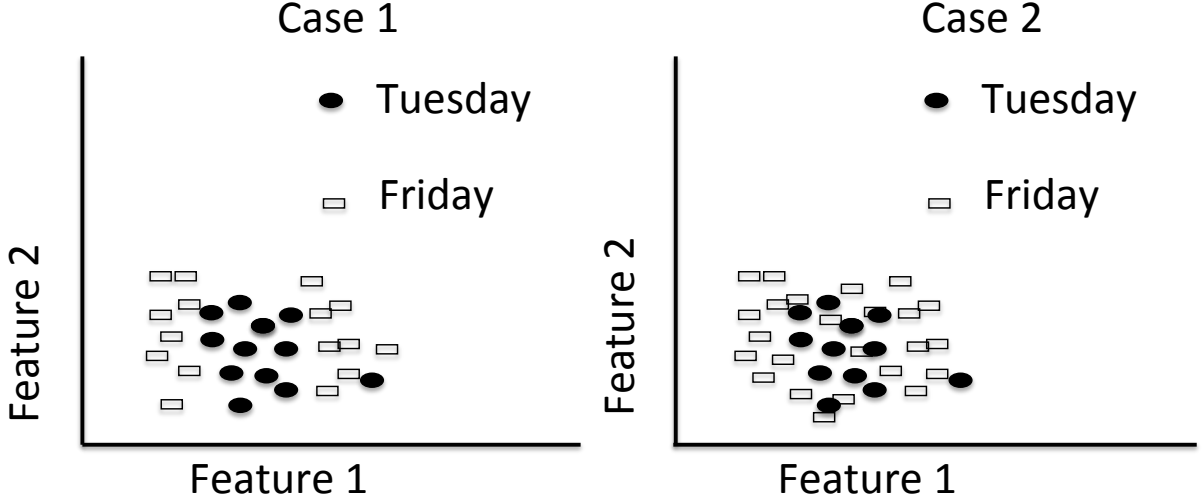
Note that there may be some activity instances that are not part of any cluster. During training, we do not consider such instances as part of any model i.e., their regular behavior. However, we do not throw these activity instances out. After training, when *Holmes* is used for anomaly detection, if an activity instance appears that does not fall into any regular model for that day, we try to combine them with similar other activity instances that were left alone during training. In this way, *Holmes* periodically retrains to capture changes in regular behavior which may be identified as anomalies by a system that does not retrain.

From the bottom layer of Figure 5.3, we get one or more clusters for each day of the week. From here on, the algorithm progresses upwards like agglomerative clustering [76] i.e., it merges two clusters from the bottom layer which are closest to each other and advances the merged cluster to the upper layer. The important part of this merging step calculating

the distance between two clusters i.e., the linkage. For each cluster, we define the diameter of a cluster as the maximum distance between any two points. We merge two clusters with diameter  $d_1$  and  $d_2$  only if after merging the diameter of the new cluster  $d_{new}$  satisfies the relations in (5.1). Note that if only one of the conditions is satisfied, then we do not merge these two clusters. Merging in such cases would generalize the cluster (for which the condition is not satisfied) too much.

$$\begin{aligned} \frac{d_1}{d_{new}} &> MERGE\_THRESHOLD \\ \frac{d_2}{d_{new}} &> MERGE\_THRESHOLD \end{aligned} \quad (5.1)$$

In each layer of Figure 5.3, a cluster can be merged with multiple clusters under the merging condition, *Holmes* selects the one with minimum  $d_{new}$ . If a cluster cannot be merged with another one, it moves to the upper layer. The merged clusters move to the upper layer together as one cluster. If at any layer no new merging is possible, the algorithm stops.



**Figure 5.4: Sample scatter plot of instances of a specific activity for 2 different cases showing need of context-aware clustering**

We argue that our context-aware (day of the week being the context here) clustering algorithm is necessary to model regular behavior, because without it we may not capture if the resident behaves in a specific way on specific days of the week. For example, consider

the sample scatter plots of two different cases shown in Figure 5.4. If we cluster the activity instances without considering the day of the week, then all the activity instances of Tuesday and Friday would be clustered together in one cluster in both cases. We may separate the cluster of Tuesday in Case 1, if we specifically mention that we want three clusters. However, it is difficult to know how many different clusters there may be apriori. However, even then it won't be possible to separate instances of Tuesday in Case 2. This is why clustering activity instances from different days separately is necessary, and we go one step further by also merging days that have similar behavior.

Our algorithm creates a separate cluster for Tuesday, and two different clusters for Friday for Case 1. However, for Case 2, our algorithm creates one cluster for Tuesday and one cluster for Friday. Another point to note from Figure 5.4 is that there is one activity instance from Tuesday (in both cases, on the right side) which is abnormal. If we cluster all instances together, we are not able to identify it. One alternate to our solution is just to use different models (i.e., clusters) for different days of the week. We argue that such an approach would fail to capture several high level regularities in a user's behavior such as which days of the week behavior is similar.

Note that there may be activities that do not take place daily. If any of these activities happen only on specific days of the week, then it would be included in that days model. However, if any of these activities do not have any correlation with specific days of week, then there may not be enough instances of that activity in any specific day for the above clustering algorithm to execute. To address this, we cluster all instances of any such activity together based on *startTime* and *duration* using DBSCAN and get one or more models. Also, for each activity instance of such activities, we calculate a new feature named *interval* which is defined as the interval of this activity instance from the last time that activity took place in terms of days. We also test if the *interval* of the activity instances for any such activity follow any particular model.

## Modeling Activities Together

Modeling activities independently as above may not represent a resident's regular behavior completely. We also need to model causal and temporal correlations among activities. For example, if there are activities that often happen concurrently, or if one activity often happens after another activity, then we need to find these temporal correlations. Similarly, if the duration of resident's dinner is long only during the days when he/she goes to the gym, we also need to find such causal correlations. First, we need to do some pre-processing on the activity data.

**[Pre-processing]** Before applying data mining algorithms to find causal and temporal correlations among activities, we do pre-processing on activity instances. Note that activity instances are represented by tuples of the form  $(activityID, startTime, duration)$ . From the clustering step of the previous section each activity instance belongs to a unique model (cluster), and some instances do not belong to any cluster (considered not regular). We assign the instances not belonging to any cluster a default model number. In this pre-processing step, for each activity instance, we create two new tuples:  $(acID\_ModNo\_Start, startTime)$  and  $(acID\_ModNo\_End, endTime)$ . Note that the  $endTime$  for each activity instance can be calculated from  $startTime$  and  $duration$ .

**[Modeling Temporal Correlations]** The goal is to learn temporal correlations among activities. For example, two or more activities may often happen in a specific sequence, one activity may often takes place after another specific one (and the time interval between them may often be similar), two or more activities may often happen concurrently. We find the frequent sequences of activities and the normal time intervals between successive activities. We apply a sequential pattern mining algorithm [77] to learn these temporal correlations as frequent patterns. For example, for a particular day of the week, we may find a frequent pattern  $(ac1\_Mod11\_Start, ac2\_Mod21\_Start, ac2\_Mod21\_End, ac1\_Mod11\_End, ac3\_Mod31\_Start, ac3\_Mod31\_End)$  which represents that on that day, *activity2* frequently starts after *activity1* starts, and ends before *activity1* ends, and

*activity3* starts after both *activity1* and *activity2* end. We also know start and end times of all activity instances, so we can model the duration of each activity and intervals between successive events as normal distributions.

However, it may not be useful to find temporal correlations between breakfast and dinner, or between two activities one of which happen in the morning and the other at night. Therefore, we divide each day's tuples into multiple segments using a *SEG\_THRESHOLD*. Starting from the first tuple of the day, if the interval between two successive tuples exceeds *SEG\_THRESHOLD*, we end the current segment with all tuples found so far but the last one, and start a new segment with the last tuple as the starting one. In this way, each day's tuples are divided into multiple segments where each segment has a sequence of tuples. Our sequential pattern mining algorithm runs on the tuples generated from the activity instances of each day of week separately.

For each day of the week  $D_i$ , there is a set of segments  $\{s_{ij}\}$ , where each segment  $s_{ij}$  is a sequence of tuples of the form  $(acID\_ModNo\_Start, startTime)$  or  $(acID\_ModNo\_End, endTime)$ ; each such tuple is represented by  $(a_{ijk}, t_{ijk})$ . We consider this sequence  $\langle (a_{ijk}, t_{ijk}) \rangle$  corresponding to each segment  $s_{ij}$  as one stream  $st_{ij}$ . We apply sequential pattern mining [77] on the set of streams corresponding to each day of week to get the set of frequent patterns for that day.

We use a state of the art sequential pattern mining algorithm PrefixSpan [78]. For each day of week  $D_i$ , we run PrefixSpan separately with  $\{st_{ij}\}$  as input. For each member sequence  $st_{ij} = \langle (a_{ijk}, t_{ijk}) \rangle$ , the algorithm only considers the activity identifiers ( $\langle a_{ijk} \rangle$ ) and ignores the timestamps (we use timestamps in the next step). As output, we get the set of frequent patterns  $\{FP_{il}\}$ , where each  $FP_{il}$  is a sequence of activity identifiers of the form  $\langle a_{ilm} \rangle$ . A pattern is considered to be frequent if the number of different instances of that day of week  $D_i$  the pattern occurs is more than a threshold *FP\_THRESHOLD*; in the PrefixSpan algorithm, we set this threshold. For each day of week, thus we get a set of frequent patterns each of which represents a pattern of activities.

For each frequent pattern, we model the duration of each individual activity (that is part of the pattern) and the interval between successive activities separately as normal distributions. This enables us to identify cases where anomaly in start time or duration in one activity may cause irregularities in later activities in a pattern. In these cases, no anomaly should be reported in the later activities. However, if one activity in a pattern takes more / less time than usual, or the interval between any successive activities is more / less than usual, such cases should be reported as anomalies. After getting the frequent patterns for each day of the week, we merge similar patterns from multiple days.

**[Modeling Causal Correlations]** The sequential pattern mining technique enables us to understand temporal correlations among activities such as sequential orders, and time intervals among activities. From the frequent patterns, we can also infer causal correlations because all the activities of a frequent pattern often happen together. However, in some cases, there may be activities that do not have any temporal correlations, but only have causal correlations. This would happen if two activities often take place in a segment of the same day of the week, but in no specific order. The sequential pattern mining algorithm above would ignore such group of activities. In order to ensure that *Holmes* does not miss such causal correlations, we also apply an itemset mining algorithm on the segments of each day of the week. In our case, an itemset represents a group of activities, and the goal of the itemset mining algorithm is to find the frequent itemsets i.e., those where the group of activities often happen together within the same segment, irrespective of their order of occurrence.

Similar to the sequential pattern mining algorithm described in the previous section, the itemset mining algorithm is applied separately on the segments of each day of the week, and they have the same input. We apply a state of the art itemset mining algorithm named apriori algorithm [74]. As output, we get the set of frequent itemsets  $\{FI_k\}$ , where each  $FI_k$  is a set of activity identifiers. An itemset is considered frequent for a specific day of the week if its set of activities happen together more than a threshold ( $ITEM\_THRESHOLD$ ) number of different instances of that day. Note that if the activities of a frequent itemset

already belong to a frequent pattern generated by the previous section, then we discard that frequent itemset to avoid redundancy.

#### 5.4.4 Anomaly Detection and Multi-Model Filtering

After training, for each day of the week, we have one or more models (i.e., clusters) for each activity. For example, there are a total of 15 – 35 models for all activities in our evaluation (details in Evaluation section). Note that some of these clusters are calculated based on *startTime* and *duration* of the activity instances for that day of the week; each activity instance (except the irregular ones) belongs to one of these clusters. And the remaining clusters are calculated based on *totalCountPerDay* and *totalDurationPerDay*; each day (except the irregular ones) belongs to one of these clusters. For the later type of clusters, if an activity does not often take place in a specific day of the week, then a cluster is formed representing this so that during testing similar irregularity is not reported as an anomaly.

An anomaly score can be represented based on discrete values (e.g., ‘normal’, ‘abnormal’, ‘very abnormal’) or based on continuous values where a higher value would represent a more anomalous event (or the opposite). We choose a discrete representation because it is easier to interpret for the experts / caregivers. After training, for each cluster, we calculate the mean  $\mu_i$  and standard deviation  $\sigma_i$  for each feature  $i$  based on the data points that belong to that cluster during training. During testing, for a new data point  $x$  represented by  $(x_1, x_2, \dots, x_k)$ , we calculate the *Mahalanobis Distance*  $d$  from a cluster according to Equation (5.2). If the instances in a cluster are normally distributed along the  $k$  dimensions (i.e.,  $k$  features), then  $d \leq \sqrt{k}$  means  $x$  is within one standard deviation from the cluster center. If  $x$  is not within two standard deviations (i.e.,  $d > 2 * \sqrt{k}$ ) from the cluster center, we consider that  $x$  does not belong to this cluster.

$$d = \sqrt{\sum_{i=1}^k \frac{(x_i - \mu_i)^2}{\sigma_i^2}} \quad (5.2)$$

*Mahalanobis Distance* is widely used for cluster analysis, as it normalizes variation in each feature value by its standard deviation in the training data. Note that in (5.2), we show a simplified formula of the *Mahalanobis Distance* (for ease of understanding) where we assume the covariance matrix of the features is diagonal. However, if for any activity, the covariance matrix is not diagonal, we calculate the *Mahalanobis Distance* considering the covariance matrix.

At the end of each specific day of the week, we take the set of activities that took place that day and look for different anomalies shown in Figure 5.1. First of all, we look for any missing activity (a kind of point anomaly) that are either daily or periodic based on the corresponding clusters trained based on day based or interval based features. After that, we consider each individual activity instance which took place during that day. If an activity instance does not belong to any of the clusters representing that activity on that day, then we consider that activity instance an anomaly. Otherwise, we consider that activity instance normal. In this way, *Holmes* can detect point anomalies based on irregular features (as in Figure 5.1).

Next, for each activity, *Holmes* tests if there is any anomaly from the clusters based on collective features to detect the collective anomalies of type ‘Instances from Same Activity’ (Figure 5.1). During testing, for each activity, if there is an anomaly from the clusters based on features from individual instances, but no anomaly from the clusters based on collective features, then such anomalies are suppressed in this layer to reduce false alarms. Otherwise, if there is any anomaly from the clusters based on collective features and / or in the clusters based on features from individual instances, such anomalies are forwarded to the ‘Semantic Anomaly Filter’ layer if they are not caused by any prior activity instance in a frequent pattern (discussed below). This logic can be changed by the domain experts according to the necessities of different applications

We also report anomalies in the set of frequent patterns and frequent itemsets for each day of the week. In this way, *Holmes* detects collective anomalies of type ‘Activity Instances

from Multiple Activities’ as shown in Figure 5.1. For each frequent pattern (i.e., sequence of activities), if we find one or more activities are out of sequence, we report an anomaly in that pattern to the ‘Semantic Anomaly Filter’ layer. The time intervals between successive activities in a pattern is modeled as a normal distribution, and if any interval in testing data falls two standard deviation outside the mean, we also report it as an anomaly. Similarly, for each frequent itemset (i.e., group of activities), if only a subset of them takes place, we report an anomaly in that itemset.

Together, the set of frequent patterns and itemsets may modify number of anomalies in individual activity instances. For instance, during training, if we find that dishwashing often takes place after dinner, then there will be a corresponding pattern. Now, if during testing, on a particular day, dinner takes place long after normal time, then the start time of dishwashing will also be delayed. In that case only anomaly in dinner should be reported. In general, if any activity instance within a pattern is marked as abnormal by the *Mahalanobis Distance*, we check if the anomaly is due to any of the previous activity instances of the pattern. We can check that with the help of our models of duration of each individual activity and intervals of successive activity instances for that pattern. If the anomaly is due to a previous activity instance of the pattern, we do not report that anomaly. This checking helps in reducing false alarms. If the anomaly is not due to one / more anomalies in previous activities in a frequent pattern, then that anomaly is reported to the ‘Semantic Anomaly Filter’ layer. This logic can be changed by the domain experts according to the necessities of different applications

Our anomaly score calculation is generalized. If *Holmes* is used to monitor symptoms of a particular disease (e.g., depression, diabetes, dementia), then an expert can select a group of activities and specific sets of frequent patterns and itemsets so that *Holmes* monitors only anomalies in the selected cases. Also, an expert can assign different weights to anomalies in different activities or patterns. Note that anomaly scores are calculated on a daily basis (not in real time). An expert can look at these scores over a group of days / weeks for diagnosis.

### 5.4.5 Semantic Anomaly Filters

Here, the goal is to detect anomalies that may be normal under an explainable scenario. Each of these explainable scenarios is represented by a semantic rule. If *Holmes* is informed that one of the explainable scenarios is currently true, then it checks if any of the anomalies is related to that scenario so that the corresponding anomaly can be suppressed. If no such scenario is currently applicable, then that anomaly is reported to the expert / the resident. If a reported anomaly may be explained by a new scenario, that was not observed before, then the expert or the resident can notify *Holmes* about this scenario and the corresponding variations. *Holmes* can update the set of ‘Explainable Scenarios’ accordingly so that in future such anomalies can be suppressed under that scenario.

The ‘Explainable Scenarios’ may include: entertaining visitors, power outage, recovering from major medical operation, and extreme weather (cold/heat wave). This list is extensible over time. One of the novelties of our system is this set of semantic rules that will help to reduce false alarms in anomaly detection. For example, if our system detects that the user is out of house for more than a day, it does not look for anomalies in other activities e.g., eating, sleeping. It only reports that user is not in house and thus reduces false positives in other activities. As another example, if it is now known that a person is suffering from prostate problems, then frequent toilet visits are highly likely and thus should not be reported as anomalies. Another type of semantic filters are defined by experts in the form “Do not report anomaly in an activity if there are less than  $X$  anomalies in that activity in  $Y$  days”.

Note that *Holmes* does not claim or aim to detect these explainable scenarios automatically. However, if *Holmes* is notified about the presence of a specific scenario, it learns to look for specific anomalies that may occur during that scenario and filter them out. The more that scenario occurs, the better *Holmes* can learn the anomalies associated with it. We argue that there will be a common set of semantic rules applicable in any home. And, based on the feedback from residents and experts, new semantic rules can be learned.

## 5.5 Evaluation

In this section, we first describe the experimental data used for evaluation of *Holmes*. Then we discuss the state of the art systems to which we compare the performance of *Holmes*. Finally, we present the results of evaluation for each data set.

### 5.5.1 Experimental Data

We evaluate *Holmes* using four types of data. Firstly, we have collected activity data from a single resident home for six months for a comprehensive evaluation of *Holmes*. Secondly, we show evaluation results based on activity data collected by an online survey from 11 persons. Next, we evaluate using activity data from three single-resident apartments that are part of the publicly available Casas dataset. Finally, we use activity data collected by a senior safety system provider named BeClose for four months from a single resident home.

### 5.5.2 Compared Algorithms

We compare the performance of *Holmes* with three state of the art algorithms. We describe these algorithms briefly here.

#### Statistical Approach

We implement a statistical approach similar to [2]. For each activity, for each hour of the day, we calculate the mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of how long that activity takes place in that hour over the training period. Then, during testing, if the duration of an activity in a specific hour of the day is outside  $\mu \pm 2\sigma$ , then we consider it as an anomaly. Each activity instance can generate at most one anomaly.

## Statistical Approach with Temporal Correlations

Here, in addition to statistical measures, we also implement the temporal relationships among activities presented in [17] which models an activity based on what other activities happen before / after / during that activity. Based on the training data, a probability is calculated for each activity based on what activities happen before / after / during that activity. From that probability, we get an anomaly score and calculate the mean ( $\mu$ ) and standard deviation ( $\sigma$ ) of anomaly scores over all activity instances of an activity. During testing, if an activity instance's anomaly score falls outside  $\mu \pm 2\sigma$ , then we consider it as an anomaly based on temporal relationship. Adding these temporal relations help in reducing false positives generated by the statistical measures.

## Normal Clustering

Here, we cluster activity instances of each activity based on *starTime* and *duration* without considering what day of the week. A similar approach is proposed in [14]. A new activity instance is classified as anomalous or normal based on same technique as *Holmes* (no temporal correlation is used).

### 5.5.3 Threshold Values

There are different thresholds used in different parts of *Holmes* framework. Here we list the values used for these thresholds in our experiments. The value of *min\_pts* (used in DBSCAN) is set to 4, i.e., in each cluster there have to be at least four data points (four different days). We use a relatively low value so that we do not exclude too many data points as noise. [75] also suggests that using *min\_pts* more than four often produces the same sets of clusters as it produces when *min\_pts* is set to four. We set the value of *MERGE\_THRESHOLD* as 80%. Using this relatively high value ensures that we do not generalize clusters from different days too much.

As *SEG\_THRESHOLD*, we use 60 minutes. If interval between successive activities is more than an hour, we consider them in different episodes. As the value of both *FP\_THRESHOLD* and *ITEM\_THRESHOLD*, we use 80%. We select this relatively high value so that we do not get too many frequent patterns and item sets. Based on different application domains, different values may be appropriate for these thresholds and these should be set by the experts or estimated by experiments. We experimented with different values for these thresholds and use the values mentioned here as they produced most accurate results in terms of reducing false positives and negatives in our data.

### 5.5.4 Results

#### Results of Evaluation on Our Six-month Deployment Data

For the evaluation on our deployment data, we use the activity labels as produced by *AALO* based on the sensor data. Table 5.1 shows a list of clusters for all activities after training. For this table, we use the first three months data for training. We describe the clusters for each activity non-technically in the table for ease of understanding. For some activities (e.g., sleeping, breakfast), the clusters consist of only per instance features (i.e., *startTime* and *duration*); this means that there is no variation in the collective features (i.e., *totalCount* and *totalDuration*). For some activities, both types of features are used. For example, the resident’s regular behavior includes having dinner occasionally out of home on Friday / Saturday / Sunday; therefore, ‘No Dinner’ is included as a model for those days along with models based on timing and duration of dinner when in home. Some activities do not happen daily, they only happen every few days (e.g., cooking, shower). For such activities, the interval between successive occurrences is used in the models. Table 5.1 also shows us that some activities (e.g., lunch) take place in the home only on specific days (e.g., weekends).

*Holmes* also extracts 21 frequent patterns in daily activities after training with the first three months of data. Table 5.2 lists some of the patterns. Each of the patterns contains temporal and causal relationships among the activities that belong to that pattern. Patterns

Activity	No. of Models	Model Description	Activity	No. of Models	Model Description
Sleep	3	1) Mon - Thu 2) Fri - Sat 3) Sun	Dinner	3	1) Mon - Thu 2) Fri - Sun 3) Fri - Sun (No Dinner)
Lunch	3	1) Mon - Fri (No Lunch) 2) Sat - Sun 3) Sat - Sun (No Lunch)	Dishwash	3	1, 2) Mon - Sun (based on single & collective features) 3) Sat - Sun (No Dishwash)
Breakfast	2	1) Mon - Fri 2) Sat - Sun	Laptop Use	2	Based on single & collective features
Out of Home	3	1) Mon - Fri 2) Sat - Sun 3) Sat - Sun (out of town)	Toilet	3	1, 2) Mon - Sun (based on single & collective features) 3) Sat - Sun (collective features)
Cook	1	Based on interval & duration	Snack	1	Based on duraion
Prepare Meal	5	1, 2, 3) Mon - Sun 4) Mon - Fri (Collective) 5) Sat - Sun (Collective)	TV	4	1, 2) Mon - Thu (based on single & collective features) 3, 4) Fri - Sun (Same)
Shower	1	Based on interval & duration			

**Table 5.1: List of models based on 3 months of training data for all activities**

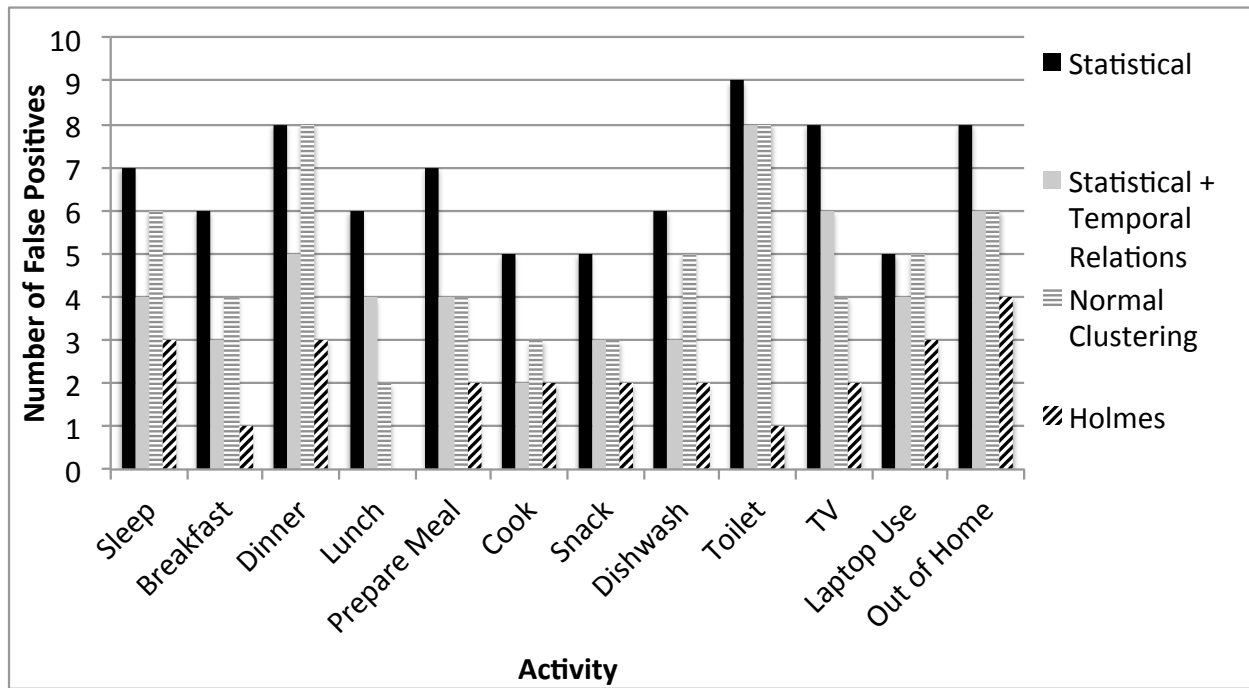
no. 1 and 2 show us that on weekends, the resident often has breakfast while watching TV. However, for weekdays, it is not the case. The other patterns in the table show us that while having lunch, dinner or snack, the resident often watches TV. ‘dishwash2’ in pattern no. 10 represents instances of dish washing that have longer duration and the pattern shows us that on days when the resident cooks, the duration of dish washing is longer.

ID	Pattern	Days
1	sleep_end, prepMeal, breakfast, leave_start	Mon - Fri
2	sleep_end, prepMeal, TV_start, breakfast, TV_end	Sat - Sun
3	prepMeal, TV_start, lunch, TV_end	Sat - Sun
4	leave_end, laptop	Mon - Sun
5	leave_end, TV_start, snack, TV_end	Mon - Sun
6	laptop, prepMeal, TV_start, dinner, dishwash1, TV_end	Mon - Sun
7	TV_end, laptop, sleep_start	Mon - Fri
8	laptop_end, TV, sleep_start	Mon - Fri
9	TV_end, sleep_start	Mon - Sun
10	cook, TV_start, dinner, dishwash2, TV_end	Periodic

**Table 5.2: A subset of frequent patterns generated from 3 months of training. If there is no other activity between the start and end of an activity, it is represented just by its name**

We compare the performance of *Holmes* on this data set with the three algorithms

described earlier. We train each algorithm on the training data, and test for anomalies on the testing data. We use a cross validation approach where in each fold, we use three months data for training, and the remaining three months for testing (e.g., data from month 1, 2, 3 for training and data from month 4, 5, 6 for testing; data from month 1, 3, 5 for training and data from month 2, 4, 6 for testing; total 20 folds). All the activity instances are hand labeled by the resident as regular / anomaly. Based on that ground truth, we calculate the average (rounded) number of false positives and false negatives in anomaly detection over all the folds by each algorithm. Figure 5.5 shows comparison of number of false positives of all the algorithms; the figure only shows the activities for which at least one of the algorithms have false positives. From this figure, we see that *Holmes* has the least number of false positives for all activities. For example, *Holmes* has zero false positives for lunch, and for toileting, the standard deviation approach has nine false positives while *Holmes* has only one.



**Figure 5.5: Average number of false positives generated by different algorithms across the cross-validation. *Holmes* generates the least number of false positives for every activity and reduces number of false positives by at least 46%**

The statistical approach performs the worst, mainly because it does not consider the

temporal relations among the activities. For example, if the resident returns home later than normal in one night, then all the subsequent activities are delayed. However, the only anomaly was his late arrival. The statistical approach would report anomalies in all subsequent activities. On average, the statistical approach generates a false alarm nearly everyday (80 false positives in 90 days on average). When we add the temporal relations to the statistical approach, the number of false positives decreases. *Holmes* performs even better than this algorithms because it learns temporal and causal relations in more detail than [17] by considering the sequence of activities rather than just considering temporal relations among pair of activities only. The normal clustering based approach performs better than the statistical approach because of modeling the temporal regularity better. However, it does not consider temporal relations among activities; therefore, it has many false positives. Number of false positives in ‘toilet use’ is very high in other algorithms, because the collective features are more significant in terms of representing normal behavior than exact timing of each toilet visit. On average (over all activities), *Holmes* reduces the number of false positives by 67%, 46%, and 56% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively.

Figure 5.6 shows comparison of number of false negatives of all algorithms; it only shows the activities for which at least one of the algorithms has false negatives. From this figure, we see that *Holmes* has the least number of false negatives for all activities. The main reason is that *Holmes* captures the variability in resident’s behavior on different days of the week. There are few instances for each of these activities in Figure 5.6, when the resident behaves unusually compared to his routine for a particular day of the week. *Holmes* correctly identifies such anomalies. On average (over all activities), *Holmes* reduces number of false negatives by 59%, 27%, and 51% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively.

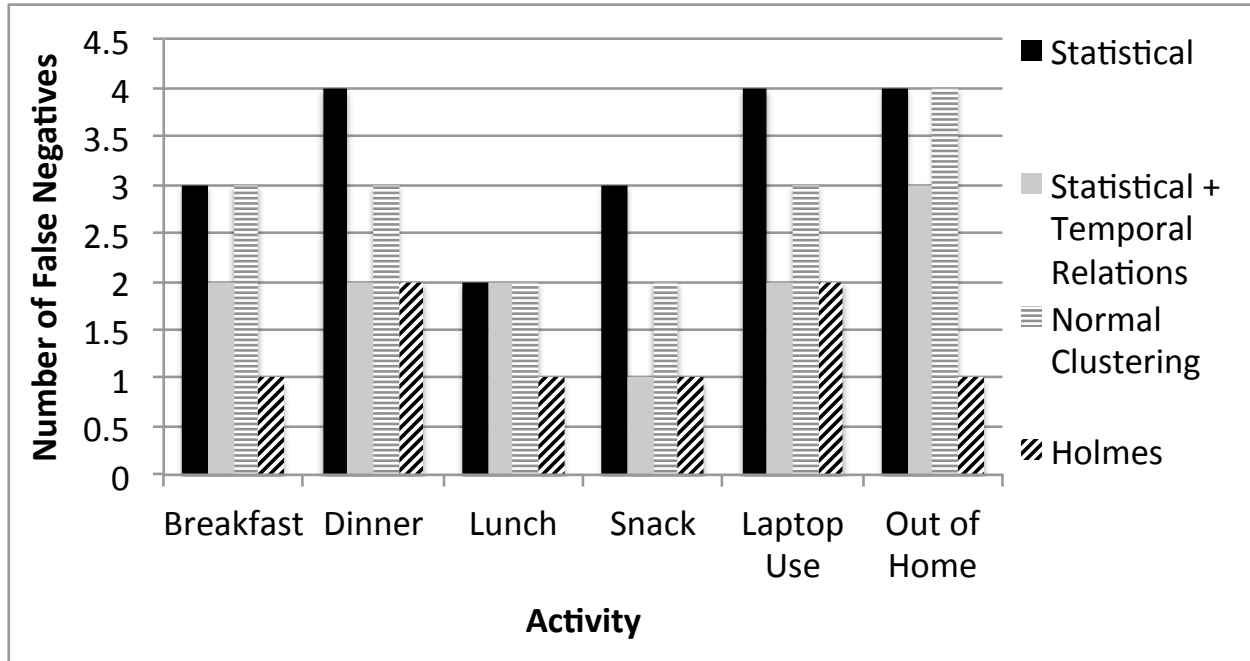
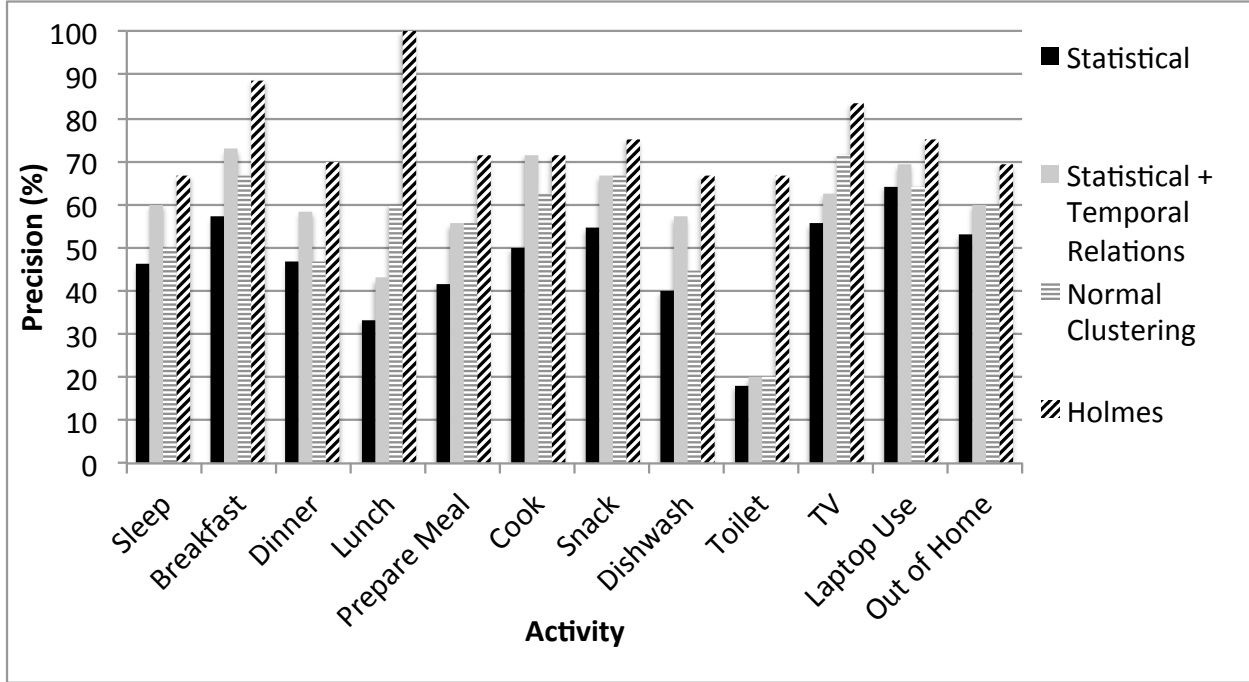


Figure 5.6: Average number of false negatives generated by different algorithms across the cross-validation. *Holmes* generates the least number of false negatives for each activity and reduces number of false negatives by at least 27%

$$\begin{aligned}
 precision &= \frac{TP}{TP + FP} \\
 recall &= \frac{TP}{TP + FN}
 \end{aligned}
 \tag{5.3}$$

Figure 5.7 and 5.8 show the corresponding average precision and recall values (defined in Equation 5.3) for different algorithms, respectively. *Holmes* has the highest precision and recall values for each activity. The relatively low precision values for some activities (even for *Holmes*) signify that those activities (e.g., ‘Sleep’, ‘Toilet’, ‘Dishwash’) the number of true anomalies are very low. On average, *Holmes* increases precision by at least 17%, and recall by at least 6%.

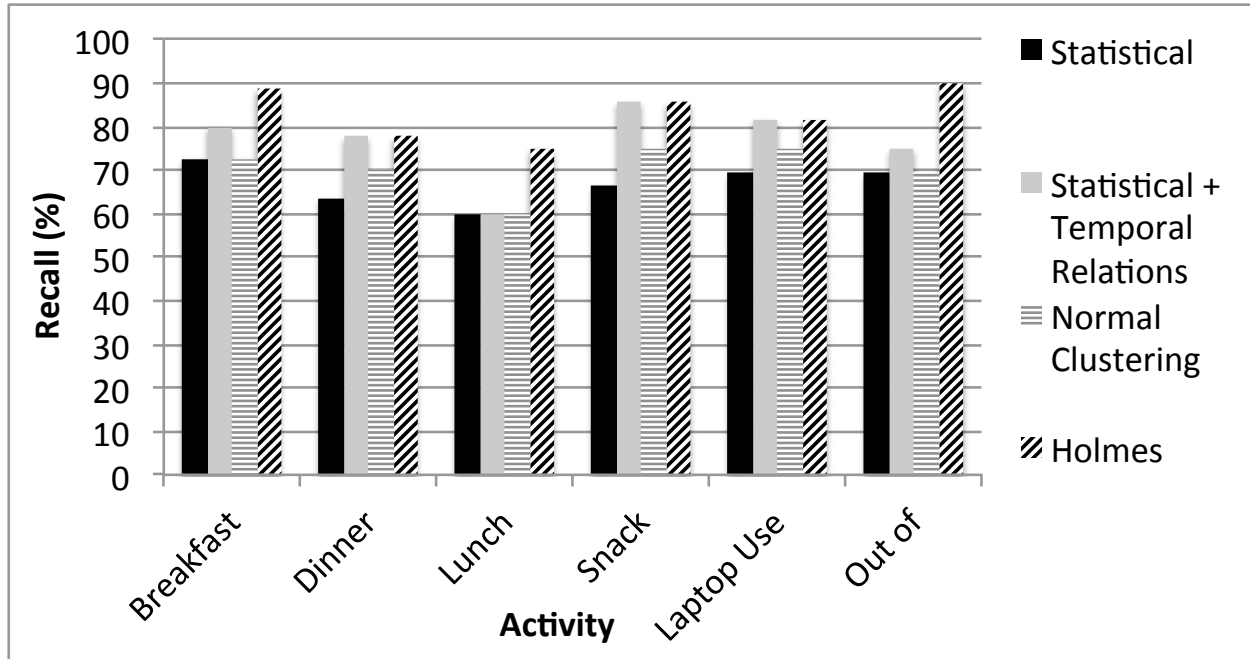
**Effect of Merge Threshold** Figure 5.9 shows the effect of *MERGE\_THRESHOLD* on the precision and recall values. Precision values do not vary with *MERGE\_THRESHOLD*



**Figure 5.7:** Average precision values for different algorithms across the cross-validation for different activities. *Holmes* has the maximum precision for each activity and increases precision by at least 17%

a lot, this is because false positives are mainly reduced due to using both point and collective features, and correlations among activities. However, at higher *MERGE\_THRESHOLD* values (more than 80%), precision values decrease due to the clusters being too specific. On the other hand, recall values generally increase with the increase of *MERGE\_THRESHOLD*. At lower values, most of the clusters for an activity are merged. Therefore, some clusters get very generalized which causes false negatives, and recall values decrease as a result. If we increase the *MERGE\_THRESHOLD* value, such generalization of clusters do not happen. However beyond 80%, we do not see significant increase in the recall values. Therefore, we use 80% as the *MERGE\_THRESHOLD* value.

**Effect of Semantic Rules** Based on the resident's feedback, we find the following semantic rules for this deployment: 1) The presence of guests in home causes specific variation in cooking, watching TV and the amount of time outside home; 2) The Ordering of food for



**Figure 5.8:** Average recall values for different algorithms across the cross-validation for different activities. *Holmes* has the maximum recall for each activity and increases recall by at least 6%

home delivery causes specific variations in preparing meal and dish washing; 3) Going to / coming back from a trip cause specific changes in entry / exit and sleeping; 4) Sickness causes variations in most of the daily activities.

For each of these rules, if *Holmes* remembers the corresponding variations in activities after the first occurrence, then the same variations may be considered as normal in later occurrences of similar scenarios. For *Holmes*, this reduces the number of false positives on average by 5 over all testing sets (20%). Note that *Holmes* does not detect such scenarios such as the presence of guests or ordering of food from outside. Rather, if such information is provided to *Holmes*, then it can learn the corresponding specific variations. We expect that as deployment time grows, more such semantic rules can be learned which will have a greater effect on reducing false alarms. Also, rules learned from one home may be applied to other homes.

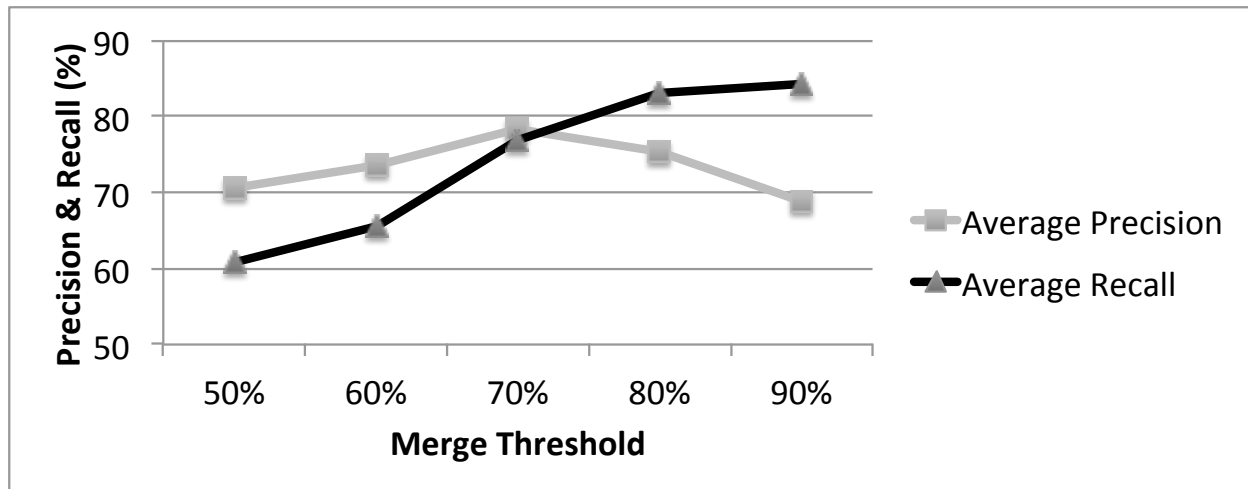


Figure 5.9: Effect of *MERGE\_THRESHOLD* on precision and recall values. We used the value 80%.

**Effect of Training with Actual Activity Labels** Note that we use the activity labels inferred by *AALO* to evaluate *Holmes*. However, as *AALO* makes error in recognizing activities, we investigate the effect of that error in the performance of *Holmes*. To do that, we train *Holmes* with the activity labels recorded by the resident using *Vocal-Diary* and calculate the average precision and recall in anomaly detection for different activities using cross-validation as before. Figure 5.10 shows the precision values of *Holmes* when trained with the activity labels recorded by *Vocal-Diary* and when trained with the activity labels inferred by *AALO*.

From Figure 5.10, we see that for most activities the precision values are same. For some activities (e.g., ‘Laptop Use’, ‘Prepare Meal’, ‘Dinner’, ‘Out of Home’), the precision is a little higher for the case where we train with the activity labels recorded by *Vocal-Diary*. Because the errors of *AALO* are mainly in time slices rather than in whole activity instances, they do not affect the performance of *Holmes* significantly.

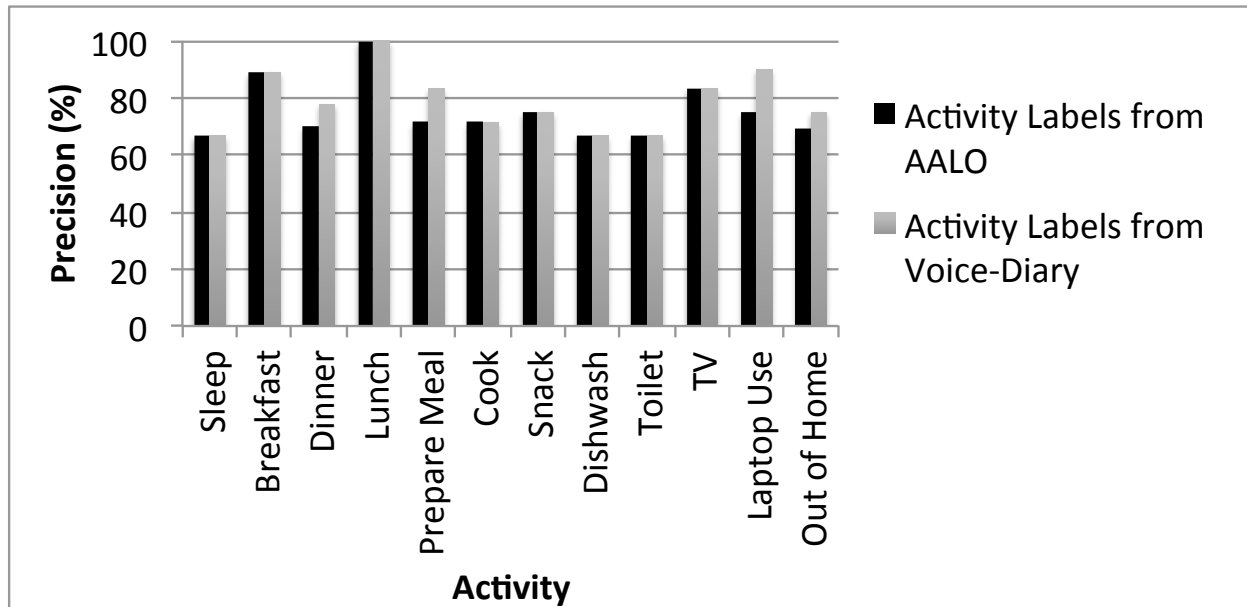
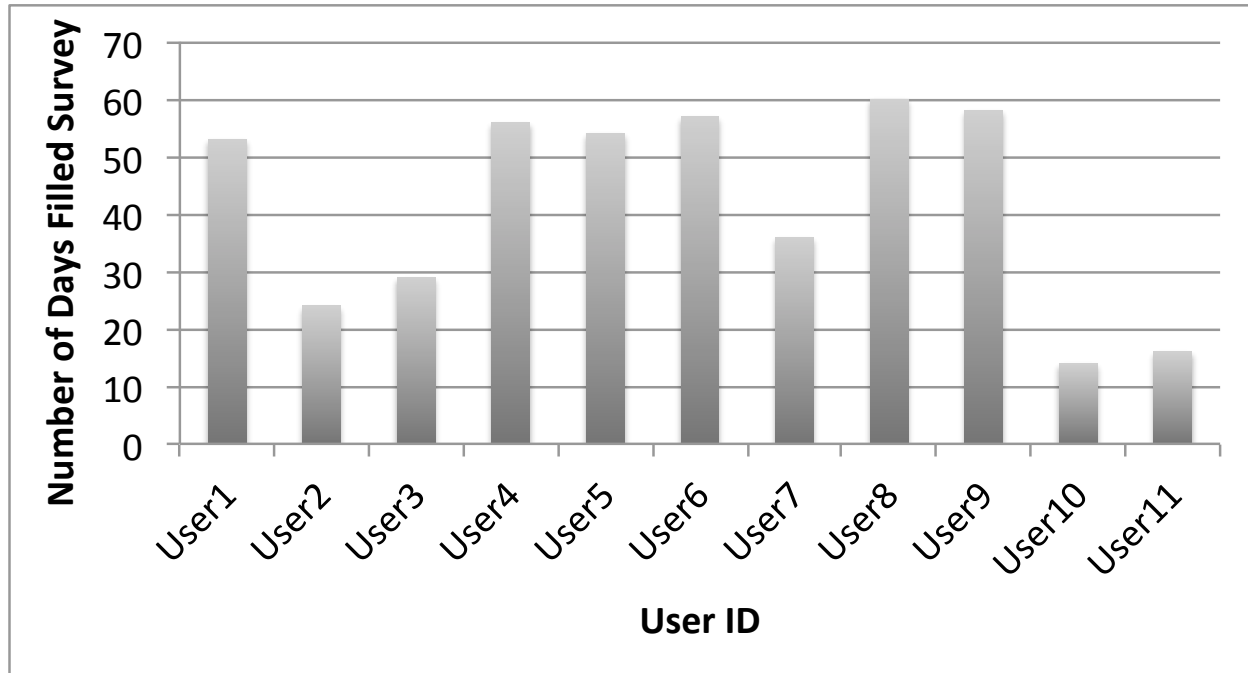


Figure 5.10: Comparison of precision values when *Holmes* is trained with actual activity labels collected by *Vocal-Diary* and when it is trained with activity labels produced by *AALO*.

### Results of Evaluation on Survey Responses

As described in Section 3.1.3, we sent daily surveys on activity details for two months to 11 persons who agreed to participate in the study. All the participants were graduate students with age 25 - 35. The questions in the survey are shown in Appendix B.1. The questions asked details for the following activities: ‘sleep’, ‘breakfast’, ‘dinner’, ‘exercise’, and ‘entertainment’. The participants also categorized each day as ‘normal’ / ‘not normal’, and in case of ‘not normal’, they also provided a reason for that (e.g., guests, sickness, family affairs, deadlines).

Figure 5.11 shows the number of days the daily survey was filled out by each participant. Six of the 11 participants filled out the forms regularly; each of these six recorded activity details for at least 50 days. We evaluate only using the responses from these six participants discarding the responses from others. Here, we see that only about 50% of the participants continued this study which indicates that online survey may not be the best way to collect ground truth.

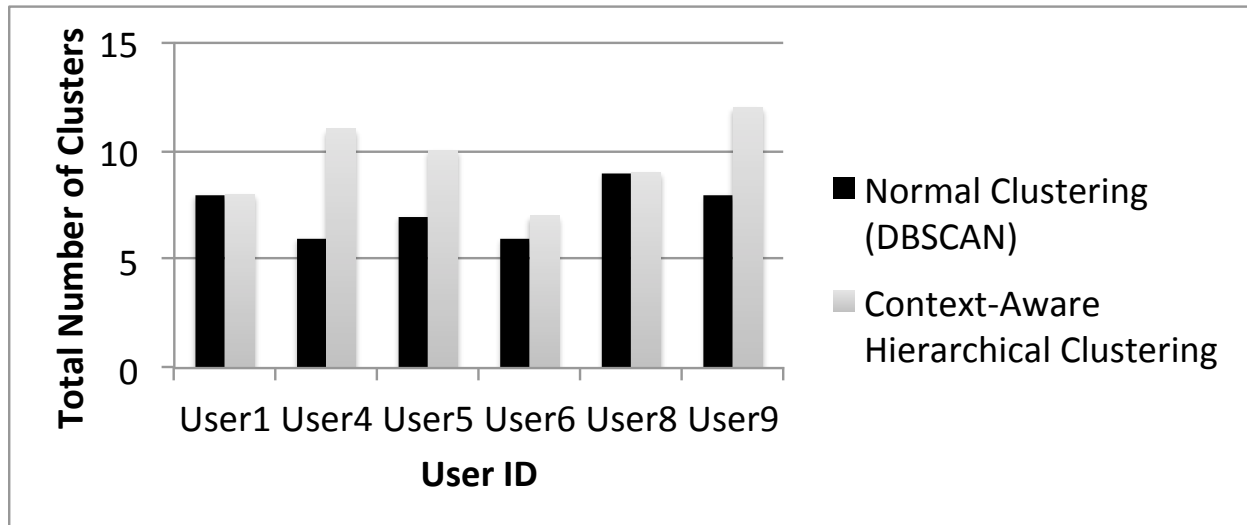


**Figure 5.11:** Number of days each survey participant filled out the daily survey.

We test if the day of the week has any effect on these activities for the participants. Figure 5.12 shows the total number of clusters for all activities when we apply our context-aware hierarchical clustering algorithm and when we apply the normal DBSCAN clustering algorithm. For these participants, we use all the days' data when they filled up the survey. From this figure, we see that for three participants (users 4, 5 and 9), our algorithms found effect of day of the week on one / more activities. For the other participants, the two approaches generate almost the same set of clusters which shows that our algorithm does not decrease the performance if a person's behavior does not depend on day of the week. Because of the limitation in the duration of this survey, and in the set of activities, we do not calculate the accuracy in anomaly detection for this dataset.

### Results of Evaluation on the Casas DataSet

As described in Section 3.1.1, the Casas dataset has data from three apartments. One of the apartments have six months of labeled activity data ('Casas Dataset 1'). The other two have

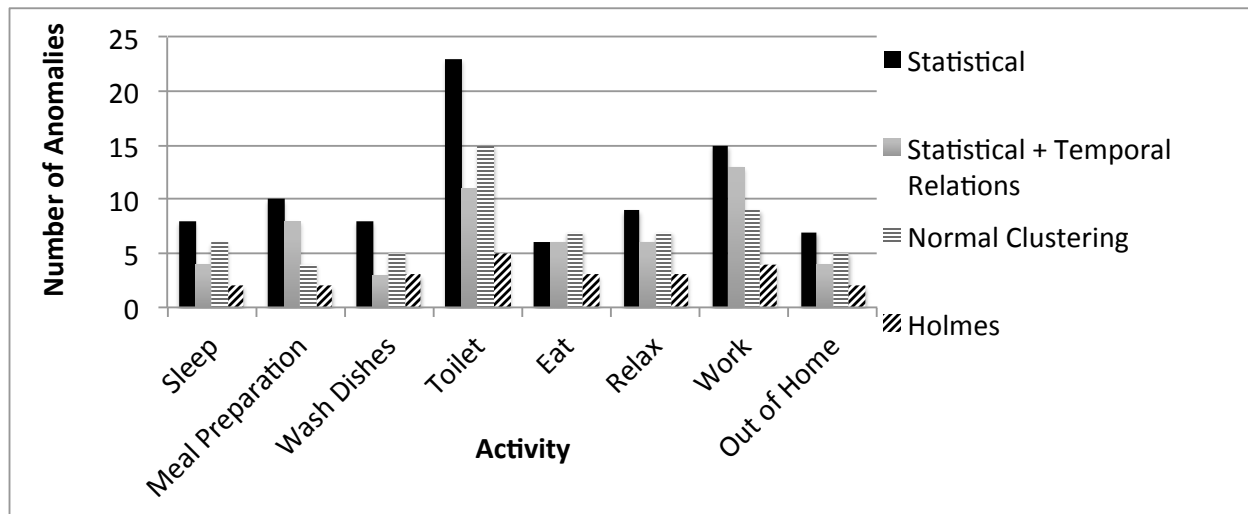


**Figure 5.12:** Total number of clusters for all activities with and without context-aware hierarchical clustering for the survey participants.

four months of labeled activity data each (‘Casas Dataset 2’ and ‘Casas Dataset 3’). These datasets do not have ground truth for anomalies. However, one of the main problems in the usability of anomaly detection systems is large numbers of false alarms. Our evaluation on the data we collected also shows high number of false positives. Therefore, in the absence of ground truth for anomalies, we compare the number of anomalies each system detects on the two public data sets. We assume that most of these detected anomalies would be false alarms, and the systems that generates less alarms is desirable.

For the ‘Cansas Dataset 1’, we have six months of activity data. Because of activity recognition errors in some of the activities (as discussed in Section 4.5.3), for evaluation of *Holmes*, we use the activity labels as given in the dataset. As before, we do a cross-validation with three full months of data as training and the remaining three months of data for testing. Figure 5.13 shows average number of anomalies generated per three months by different algorithms. *Holmes* produces the minimum number of anomalies for all activities.

For the activity ‘Out of Home’, the resident had different behavior based on day of the week which *Holmes* modeled and this helped in reducing total number of anomalies for that activity. For the activities ‘Toilet’, ‘Relax’ and ‘Work’, the collective features show more

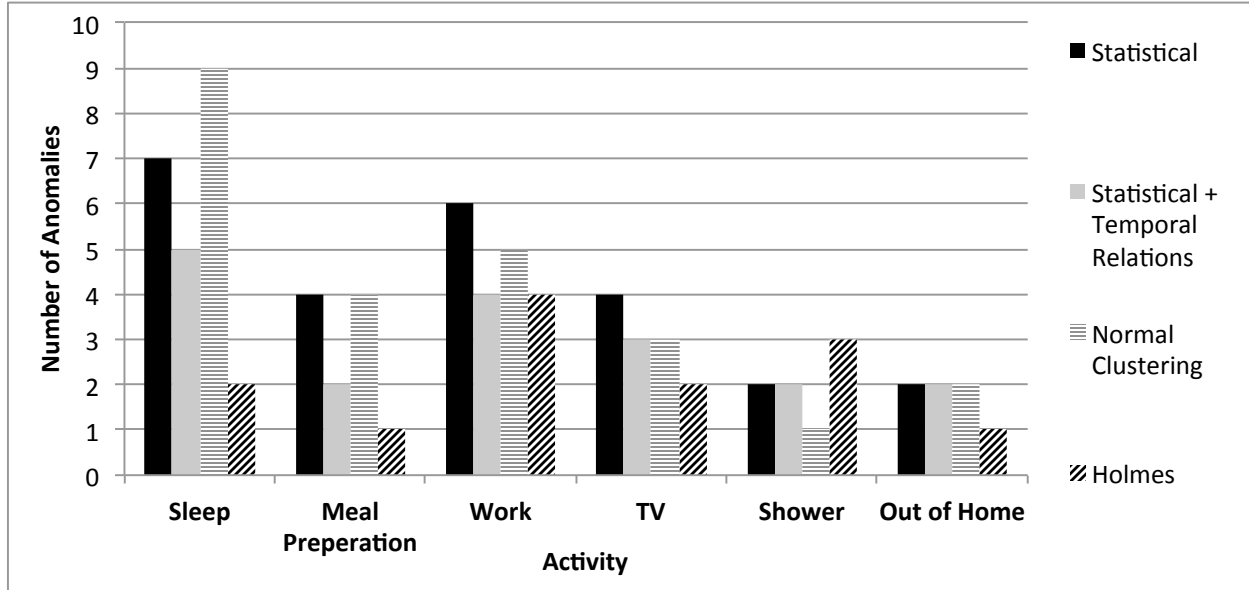


**Figure 5.13:** Average detected anomalies across all the folds of cross-validation for ‘Casas Dataset 1’. *Holmes* generates the least alarms for every activity.

regularity than the temporal features of individual activities. Therefore, *Holmes* performs significantly better than the statistical approach. For ‘Sleep’, both the temporal relationship approach and *Holmes* learns the habit of the resident that she often gets up from sleep in the middle of the night for toilet visits, and therefore produce less alarms.

For ‘Casas Dataset 2’ and ‘Casas Dataset 3’, we do a 4-fold leave-one-out cross validation; in each fold, we use three months data for training, and the remaining month for testing. We show the average (rounded) number of anomalies generated by each system over the four folds. Figures 5.14 and 5.15 show the comparison of number of anomalies in ‘Casas Dataset 2’ and ‘Casas Dataset 3’, respectively. The figures only show the activities for which at least one of the algorithms detect any anomaly. The figures show that *Holmes* generates significantly less false alarms in both data sets for most activities.

Figure 5.14 shows that *Holmes* detects more anomalies than other systems for showering. This activity usually takes place in a specific time of the day in every two / three days. However, in ‘Casas Dataset 2’, there are some instances where there are no instances of showering in a week. Because, other systems do not consider the interval between activity instances, they do not detect such cases as anomalies. Either the resident did not shower in



**Figure 5.14:** Average detected anomalies per month (rounded) across the 4-fold cross-validation for ‘Casas Dataset 2’. *Holmes* generates the least alarms for every activity but showering.

such cases or did not record when he took a shower. Overall, on average *Holmes* reduces number of alarms by 41%, 26%, and 31% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively for ‘Casas Dataset 2’. For ‘Casas Dataset 3’, on average *Holmes* reduces number of alarms by 57%, 31%, and 51% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively.

The context-aware clustering algorithm of *Holmes* helps to reduce the number of alarms, because some activities (e.g., sleeping, preparing meals and working) show different temporal characteristics (e.g., start / end time, duration) on specific days of the week in both data sets. Such instances may be few in numbers if considered together with all other instances. Statistical and normal clustering approaches do not isolate activity instances of such days, they generate alarms in such cases which *Holmes* avoids.

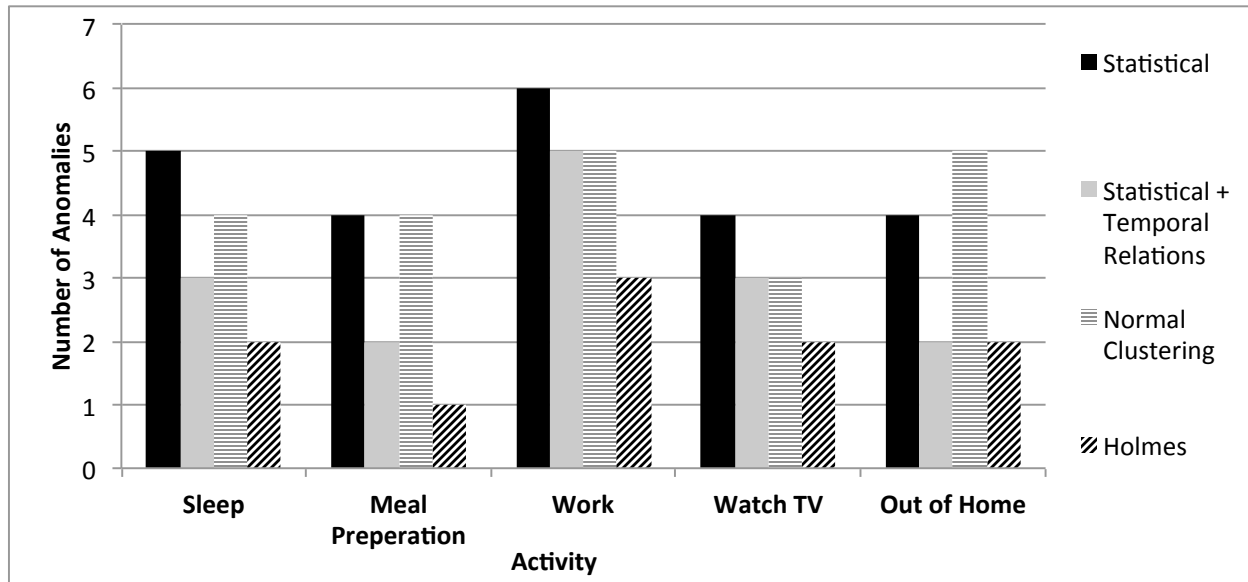


Figure 5.15: Average detected anomalies per month (rounded) across the 4-fold cross-validation for ‘Casas Dataset 3’. *Holmes* generates the least number of alarms for every activity.

### Results of Evaluation on *BeClose* Dataset

The data set from *BeClose* has four months of data from a single-resident home. The resident is an elderly person (age over 70) who uses *BeClose* so that the relatives can monitor the daily activities remotely. It does not include many activities. The activities included are sleeping, watching TV or relaxing in a couch, and going out of home. Based on the sensors deployed, the research team in *BeClose* generates the activity labels which we use for training *Holmes*. First, we investigate whether day of the week has any effect on the activities of the resident. After training *Holmes* with the four months of data, it generated two clusters for the activity ‘Out of Home’ based on day of the week. For two days of the week, the amount of time the resident spent out of home is significantly different from the other days of the week. For the other two activities, day of the week does not have any effect. For the activity ‘TV / Leisure’, collective features show more regularity than features of individual instances.

This dataset does not have ground truth for anomalies. Therefore, as in the previous section, we compare the number of anomalies generated by different systems. Here, we do a

Activity	Number of Detected Anomalies			
	Statistical	Statistical + Temporal Relations	Normal Cluster	Holmes
Sleep	4	4	3	1
TV / Leisure	4	4	4	2
Out of Home	2	2	2	0

**Table 5.3:** Average detected anomalies per month (rounded) across the 4-fold cross-validation for *BeClose* data set. *Holmes* generates the minimum number of alarms for each activity.

4-fold leave-one-out cross validation; in each fold, we use three months data for training, and the remaining month for testing. Table 5.3 shows the average number of anomalies generated across all four folds by different algorithms for different activities. It shows that *Holmes* produces the minimum number of false alarms for all activities in this data set. The statistical approach with temporal relations does not reduce the number of anomalies compared to the statistical approach mainly due to limited activities in this data set. The reduction in alarms by *Holmes* is mainly due to its use of features based on collective activity instances and day of the week based clustering approach.

## 5.6 Conclusions

Anomaly detection systems for in-home activities face many challenges including the inherent complexity in human behavior, correlations among activities, and outside factors (e.g., weather, visitor) causing specific variations in behavior. *Holmes* addresses these challenges in a novel way with the combination of using features based on both individual and collective activity instances, applying context-aware (day of the week) clustering and pattern mining algorithms, and learning semantic rules that explain deviation in behavior due to outside factors. Evaluation of such a system is challenging due to lack of true anomalies (e.g., disease, behavior change) in data. Our evaluation shows that *Holmes* reduces the number of false alarms compared to state of the art techniques.

# Chapter 6

## Conclusions

The contributions of this thesis should have a significant impact on home monitoring systems that require activity recognition, residents' regular behavior learning, and anomaly detection. Here we give a summary of each contribution, its impact, and some directions for future work to move this research forward.

### 6.1 Key Contributions towards Smart Home Deployments

#### 6.1.1 Robust and Easy-to-Use Ground Truth Collection System

We have designed, implemented and evaluated *Vocal-Diary* which is a privacy-aware, easy-to-use and robust ground truth collection system based on voice commands. *Vocal-Diary* achieves high accuracy in voice command recognition by integrating two-way acknowledgement and speaker recognition in the Microsoft Speech API (SAPI). *Vocal-Diary* is less invasive and easier to use than existing ground truth collection systems. *Vocal-Diary* can be used to collect ground truth for evaluating any new activity recognition system. Also, if for some homes it is not possible to deploy any sensors, but microphones, *Vocal-Diary* can be used to collect daily activity details from the residents which can be used to monitor them. At present, we are

observing a growing trend of personal voice assistants for smart phones. We envision that people will welcome the idea of personal voice assistants for smart homes with the use of in-home microphones. And the ideas in *Vocal-Diary* are applicable for such a personal voice assistant.

### 6.1.2 Easy-to-Train Active Learning Based Activity Recognition System

Our novel active learning based activity recognition algorithm, *AALO*, facilitates easy training by applying segmentation, itemset mining and clustering. The novelty in the segmentation step is in the way *AALO* preprocesses raw sensor data by identifying overlapping activities across multiple occupancy episodes which is a concern for any clustering based activity recognition approach. The novelty in the itemset mining and the clustering steps is in the way *AALO* captures the temporal, spatial and object use regularities in the activities of daily living to represent them in terms of these regularities. Evaluation results show that using very limited amount of user labeled data, *AALO* performs on a par with Hidden Semi Markov Model (HSMM) based activity recognizers. We envision that *AALO* will enable new applications to easily process the data collected from in-home sensors for recognizing different daily activities. We have also evaluated *AALO* in a variety of datasets which show that it can be used for data collected from a variety of home environments without requiring much modification to the algorithm.

### 6.1.3 Comprehensive Multi-Model Anomaly Detection System

Human behavior is very complex. To model such complexity a multi-model context-aware approach is necessary which few if any of the existing systems use. *Holmes* takes a comprehensive approach for modeling human behavior with the collection of novel features that include: hierarchically merging clusters from different days of the week ensuring that the merged

clusters do not become too generalized compared to the original ones, preservation of noise points found in training for future use during retraining, the use of a combination of features extracted from both individual activity instances and group of activity instances combined over a specific period, and use of sequential pattern mining and itemset mining algorithms to learn groups and sequences of activities that are part of a resident’s regular behavior (i.e., temporal and causal correlations among activities). The novelty in the anomaly detection arises due to the combination of features that include: combining point, collective and context anomalies to ensure reliability, use of semantic rules that represent logical deviation from regular behavior to reduce false alarms, and learning new semantic rules based on resident / expert feedback. Evaluation of *Holmes* on a variety of datasets shows that this comprehensive approach helps *Holmes* to perform better than the existing systems. The main ideas in *Holmes* will be useful for any domain where learning users’ behavior is necessary. For example, *Holmes* may be used to learn how someone behaves in social networks, or to learn regularity in the their online purchase pattern.

#### 6.1.4 Evaluation on Real Datasets

Most of the existing works in this domain have been evaluated in testbed / lab settings, or by a deployment in a home for a short time. However, we have evaluated our contributions through a variety of data sets most of which have months of data for real homes. The variety in the datasets also shows the generalizability of the algorithms. Getting real data is challenging. We have collected data in different ways such as public datasets, our own deployment, and an online survey. Evaluation results show that:

- *Vocal-Diary* increases precision in accurate ground truth collection by at least 40% and recall by at least 10% compared to a system that uses voice command recognition without any acknowledgement and speaker recognition.

- *AALO* performs as good as the state of the art supervised activity recognition algorithms (e.g., HSMM) even with significantly less amount of ground truth. *AALO* also performs better than the multi-instance based semi-supervised activity recognition algorithm (with settings that are practical).
- *Holmes* reduces false positives in anomaly detection by at least 46% and false negatives by at least 27% compared to state of the art systems.

## 6.2 Future Improvements

There are many notable extensions and improvements to the research that we have presented in this research which we summarize in this section.

### 6.2.1 More Generalized Segmentation Algorithm for *AALO*

The segmentation algorithm for *AALO* works on a per room basis which is fine for most types of homes and activities. However, if we want to use our algorithm in a studio-type home where there is only one room, or for some kind of activities that can span across multiple rooms, then there is no straightforward way to use *AALO*. Future research should address this limitation by designing more generalized segmentation algorithms. Also, the effect of faulty sensors on the segmentation step should be thoroughly investigated.

### 6.2.2 Multi-Resident Homes

*AALO* and *Holmes* work for single-resident homes only. This is a limitation that future research should address. It should be straightforward to extend *AALO* for multi-resident homes if each sensor firing is associated with the corresponding user who triggers it. Use of additional sensors (e.g., height sensors, weight sensors) may be investigated for doing such data association. If such data association is not possible, we can use the variation in temporal characteristics of how different users perform an activity. The complexity in using *Holmes*

for multi-resident homes is that one resident's activities may affect those of the other. Future research should modify *Holmes* to address this issue.

### 6.2.3 Better Training for *Vocal-Diary* in the Presence of Noise

Currently, *Vocal-Diary* only works when the resident gives command when there is no ambient noise. The utility of *Vocal-Diary* is in reducing the errors when ambient noise is misclassified as voice commands. However, we believe that with richer training sets that contain scenarios when the resident talks in the presence of ambient noise should be able to solve this problem. Also, significant progress has been made in the field of gesture recognition. Using gesture instead of voice may make a ground truth collection system easier to use. Further research along these lines is necessary.

### 6.2.4 Enriching a Generalized Set of Semantic Rules

There is good possibility that semantic rules learned from one home (resident) may be applicable for others. In the future, research should be done to learn semantic rules from different homes and learn the similarity between them. A database of semantic rules for different application scenarios will go a long way to reduce false alarms in monitoring systems which will make them more reliable. In addition, more research is necessary to identify the best way to collect feedback from the residents to learn semantic rules.

# Appendices

## Appendix A

# Online Survey to Select Participants for Providing Daily Activity Details

### A.1 Pre-screening Survey Sent to Potential Participants

## Sample Survey for Collecting Daily Activity Information

**1. We want to collect some details about your daily activities for two months (starting from September, 2013). Everyday you will need to fill out those details at a time convenient for you. The daily activities of our interest include: Eating, Sleep, Exercise, Entertainment. The details for each activity will be given in subsequent questions.**

**Note that:**

**\*\*\* You can opt out of the survey at any time during these two months.**

**\*\*\* If on a particular day, you do not feel like submitting the details, you can skip for that particular day.**

**\*\*\* You will remain completely anonymous i.e., even we will not know which details are provided by which participant. You will have the freedom to choose a code name for you and each day, along with the details you will provide your code name so that we can group responses from the same user.**

**\*\*\* You will have the right to revoke all data you provided at any point of time in future.**

**\*\*\* Your data will be used strictly for research purpose. Even the code names will not be published in the resulting publications.**

**Will you be interested to take part in such a survey? Note that, you do not need to provide details of all activities. You will have the choice to select the subset of activities of your choice.**

Yes

No

**2. Select which of the following details of 'Eating' activity are you willing to share?**

Approximate time of breakfast

What did you have for breakfast (Content)

Approximate time of Lunch

What did you have for lunch (Content)

Approximate time of dinner

What did you have for dinner (Content)

How many times did you have Coffee today?

How many times did you have other snacks today?

None of the above

**3. Select which of the following details of 'Sleeping' activity are you willing to share?**

When did go to bed at night?

How many hours did you sleep?

How was the quality of your sleep? (Sound / Difficult)

Approximately how long were you awake after getting to bed?

Approximately how long did you stay in the bed in the morning after you got up?

Did you take any nap during other time of day?

None of the above

**4. Select which of the following details of 'Exercise' activity are you willing to share?**

What kind of Exercise did you do today? (Walk / Run / Gym / Bike / Sports / Other)

How long did you exercise today?

None of the above

**5. Select which of the following details of 'Entertainment' activity are you willing to share?**

What kinds of 'Entertainment' activities did you do today? (TV / Dine out / Tourism / Social Gathering)

How long did you do the above activity / activities?

None of the above

**6. Will you also be willing any of the following details about your mood everyday?**

No

General Mood (Happy / Sad / Stressed / Normal)

More fine grained mood information like mood in the morning, afternoon, evening, etc.

**7. Will you be interested in classifying your day in the following way?**

Not interested to share

Normal / Abnormal

If not normal, what is the reason? (Guests / Exam / Deadline / Family affairs / On a trip / Important Occasion / Sick / Under medication)

**8. Will you be interested in sharing your weight once a week?**

Yes

No

**9. Will you be interested in reporting the following weather information in your area?**

No

Yes (Sunny / Cloudy / Rainy / Snow)

**10. Do you have any additional comments about the kinds of data we want to collect? Anything you did not like about it? Please feel free to provide us feedback so that we can refine our efforts. Thanks.**

Done

Powered by **SurveyMonkey**

Check out our [sample surveys](#) and create your own now!

## **A.2 General Consent Form Sent to Potential Participants**

## Informed Consent Agreement

**Please read this consent agreement carefully before you decide to participate in the study.**

**Purpose of the research study:** Monitoring daily activities for detecting irregularities in behavior is an important research problem. It is important to detect irregularities in eating and sleeping habits, exercise and entertainment routines, general mood, and in the correlation among these activities. A critical part of this research is to model human behavior in terms of daily activities that can be considered as 'normal'; these models represent regularities of your daily activities e.g., you usually have breakfast by 9 AM, you usually sleep for about 8 hours at night, you usually exercise for around 3 hours in a week. However, it is a challenging problem to figure out what features to use for modeling regular behaviors, and often multiple models are necessary. For example, one model may be needed for weekends, one for weekdays, or may be one for Monday and Tuesdays and one for the other days. And these models may vary for different persons. The goal of this study is to collect information about daily activities from at least 10 subjects for 2 months each. The daily activities we are interested are eat, sleep, exercise, and entertainment. This data will enable us to develop algorithms to model regular behavior on a variety of subjects.

**What you will do in the study:** You will select in a preliminary survey which activity details you are willing to share anonymously. After you select the activities, daily emails will be sent to you requesting you to fill out a survey providing your selected activity details. This process will continue for two months. You will have the freedom to skip a survey of any particular day. You will have the freedom to choose a code name for you, and each day, along with the details you will provide your self-selected code names so that we can group responses from the same user.


**Time required:** 2 months continuously.

**Risks:** There is no risk associated with this study. Daily emails will be sent to you to remind you to complete that day's survey.

**Benefits:** There are no direct benefits to you from this research study. The algorithms and techniques developed using this study will be invaluable for progress in this research problem.

**Confidentiality:** Materials will be stored in a password-protected computer. Data will be maintained for 2 years after collection unless you request removal. You will remain completely anonymous i.e., even we will not know which details are provided by which participant. You will have the freedom to choose a code name for them, and each day, along with the details you will provide your self-selected code names so that we can group responses from the same user.

**Voluntary participation:** Your participation in the study is completely voluntary.

IRB-SBS Office Use Only		
Protocol #	2013-0256	
Approved	from: 7/01/13	to: 6/30/17
SBS Staff		

**Right to withdraw from the study:** You have the right to withdraw from the study at any time without penalty. You cannot withdraw previous data as they were submitted anonymously.

**How to withdraw from the study:** Contact investigator Enamul Hoque ([eh6p@virginia.edu](mailto:eh6p@virginia.edu)) to have your data removed from the database.

**Payment:** You will receive no payment for participating in the study.

**If you have questions about the study, contact:**

Enamul Hoque

Dept. of Computer Science, Rice Hall 220

University of Virginia, 85 Engineers Way, Charlottesville, VA 22904.

Telephone: (434) 284-1091

Email address: [eh6p@virginia.edu](mailto:eh6p@virginia.edu)

John A. Stankovic

Dept. of Computer Science, Rice Hall

University of Virginia, 85 Engineers Way, Charlottesville, VA 22904.

Telephone: (434) 982-2275

Email address: [stankvoic@cs.virginia.edu](mailto:stankvoic@cs.virginia.edu)

**If you have questions about your rights in the study, contact:**

Tonya R. Moon, Ph.D.

Chair, Institutional Review Board for the Social and Behavioral Sciences

One Morton Dr Suite 500

University of Virginia, P.O. Box 800392

Charlottesville, VA 22908-0392

Telephone: (434) 924-5999

Email: [irbsbshelp@virginia.edu](mailto:irbsbshelp@virginia.edu)


Website: [www.virginia.edu/vpr/irb/sbs](http://www.virginia.edu/vpr/irb/sbs)

**Agreement:**

I agree to participate in the research study described above.

**Signature:** \_\_\_\_\_ **Date:** \_\_\_\_\_

You will receive a copy of this form for your records.

IRB-SBS Office Use Only		
Protocol #	2013-0256	
Approved	from: 7/01/13	to: 6/30/17
SBS Staff		

## Appendix B

# Form for Providing Daily Activity Details

### B.1 Google Form Sent out Daily for Two Months

# Survey to Collect Daily Activity Details

**\* Required**

**What is your user name? \***

Please write down the anonymous user name you have chosen for yourself for this study.

**Today's Date**

## Breakfast

**Where did you have breakfast yesterday? \***

- ☐ Home
- ☐ Work
- ☐ Other
- ☐ Did not have breakfast

**Approximately when did you have breakfast yesterday?**

Example: 11:00 AM

## Lunch

**Where did you have lunch yesterday? \***

- ☐ Home
- ☐ Work
- ☐ Dine out
- ☐ Other
- ☐ Did not have lunch

**Approximately when did you have lunch yesterday**

Example: 11:00 AM

## Dinner

**Where did you have dinner yesterday? \***

- ☐ Home
- ☐ Dine out
- ☐ Work
- ☐ Other
- ☐ Did not have dinner

**Approximately when did you have dinner yesterday?**

Example: 11:00 AM

**How many time did you have coffee yesterday****How many times did you have other snacks yesterday?**

## Sleep

**Approximately when did you go to sleep last night? \***

Example: 11:00 AM

**Approximately how long did you sleep last night (In Hours)? \***

In Hours

**How was the quality of your sleep**

- ☐ Sound
- ☐ Poor
- ☐ Not sure

**Did you take a nap any other time of yesterday?**

- ☐ Yes
- ☐ No

## Exercise

**What kind of exercise did you do yesterday? \***

You can choose more than one.

- ☐ Did not do any

- ☐ Walk
- ☐ Run
- ☐ Gym
- ☐ Sports
- ☐ Swim
- ☐ Biking
- ☐ Hiking
- ☐ Other:

**Approximately how long did you exercise yesterday? \***

- ☐ < 30 Minutes
- ☐ 30 Minutes - 1 Hour
- ☐ 1 - 2 Hours
- ☐ > 2 Hours
- ☐ Did not do any exercise

**Other****Which of the following activities did you do yesterday? \***

You can choose more than one.

- ☐ TV
- ☐ Dine out
- ☐ Movie theater
- ☐ Social gathering
- ☐ Grocery
- ☐ Shopping
- ☐ Outdoor visit / recreation
- ☐ None of the above
- ☐ Other:

**How was your overall mode yesterday?**

- ☐ Happy
- ☐ Sad
- ☐ Neutral
- ☐ Stressed
- ☐ Other:

**How was the overall weather yesterday?**

You can choose more than one.

- ☐ Sunny
- ☐ Cloudy
- ☐ Rainy
- ☐ Snow

**How would you classify yesterday overall? \***

- ☐ Do not want to share
- ☐ Normal
- ☐ Not normal due to family affairs or occasions
- ☐ Not normal due to presence of guests
- ☐ Not normal due to exams / deadlines
- ☐ Not normal due to sickness or being on medication
- ☐ Other:

Never submit passwords through Google Forms.



This form was created inside of UVa.

[Report Abuse](#) - [Terms of Service](#) - [Additional Terms](#)

## Appendix C

# IRB Approval to Deploy Sensors and Microphones

### C.1 General Consent Form Sent to Potential Participants

## Informed Consent Agreement

**Please read this consent agreement carefully before you decide to participate in the study.**

**Purpose of the research study:** Automatic recognition of daily activities (e.g., sleeping, eating, exercise) is one of the core components of many home health care and energy monitoring applications. Our goal is to develop a reliable activity recognition system that addresses the realistic scenarios that may occur due to long term use of the applications. To address these realities, we need to deploy our activity recognition system in real homes for a long duration (6-12 months). From the data collected during these deployments, we will be able to design novel algorithms for activity recognition and learn your behavior in terms of daily activities that will enable us to detect deviations from your regular behavior.

**What you will do in the study:** We will deploy some non-invasive sensors in your home. We will attach motion sensors in different rooms' (bedroom, kitchen, living room, and toilet) walls to monitor if the room is occupied by someone. We will place pressure pads in your bed and chairs / sofas in different rooms to detect when someone lies/sits on them. We will attach contact sensors in doors of different rooms to detect when they are opened / closed. We will attach contact sensors in kitchen cabinets, dishwasher, refrigerator to monitor when those are used; the sensors will not create any discomfort in using those objects. If you want that no sensor is deployed in a particular room, then we will not deploy any sensor there. There will be a laptop in a suitable location in home to collect data from the sensors. You cannot log in to this laptop, only the research team will have access.

In addition, there will be a microphone connected to the laptop. Before starting and ending an activity, you can talk to the microphone in a predefined format: 'System activity\_name start' or 'System activity\_name end'. The program running the microphone will then talk back to you confirming what it listened in the format: 'Have you started / ended activity\_name?'. In reply you will say 'System yes / no'. If you say 'yes', then the program will save the corresponding timestamp, activity name, start / end status, and speaker id. Note that the program will not store your voice; it will just store the above information. And if you do not talk to the microphone in the specific format, the program will not record and store anything. In this way, we ensure your privacy. Also, your interaction with the microphone is completely voluntary. If on any day, you do not give any command to the microphone, we will not send you any reminder. You will communicate to the microphone when you want.

**Time required:** 6-12 months continuously.


**Risks:** There is no risk associated with this study.

**Benefits:** There are no direct benefits for the participants of this research study. The algorithms and techniques developed using this study will be invaluable for progress in this research problem.

---

Revision date: 1/2/14

Page 1

IRB-SBS Office Use Only	
Protocol #	2013-0453-00
Approved	from: 1/9/14 to: 1/8/15
SBS Staff	

**Confidentiality:** Materials will be stored on a password-protected laptop placed in the participant's home. Data will also be uploaded to the cloud after being anonymized; only researchers involved in this project will have access to the data stored in the cloud. Data will be maintained for 10 years after collection unless the participant requests removal. The participants will remain completely anonymous i.e., their identity will not be exposed in the publications.

**Voluntary participation:** Your participation in the study is completely voluntary.

**Right to withdraw from the study:** You have the right to withdraw from the study at any time without penalty. After withdrawal, if you want us to delete all previous data collected from your home, then they will be deleted from the laptop and removed from the cloud immediately. Otherwise, we will utilize the data collected until you withdraw yourself from the study. Note that we anonymize the data before uploading to the cloud, and save the mapping between your identity and the anonymized data in an encrypted machine in the CS department server. If you withdraw from the study and want your data removed from the cloud, we can do so with the help of this mapping.

**How to withdraw from the study:** Contact investigator Enamul Hoque ([eh6p@virginia.edu](mailto:eh6p@virginia.edu)) to have your data removed from the database.

**Payment:** You will receive no payment for participating in the study.

**If you have questions about the study, contact:**

Enamul Hoque

Dept. of Computer Science, Rice Hall 220

University of Virginia, 85 Engineers Way, Charlottesville, VA 22904.

Telephone: (434) 284-1091; Email address: [eh6p@virginia.edu](mailto:eh6p@virginia.edu)

John A. Stankovic

Dept. of Computer Science, Rice Hall

University of Virginia, 85 Engineers Way, Charlottesville, VA 22904.

Telephone: (434) 982-2275; Email address: [stankvoic@cs.virginia.edu](mailto:stankvoic@cs.virginia.edu)

**If you have questions about your rights in the study, contact:**

Tonya R. Moon, Ph.D.

Chair, Institutional Review Board for the Social and Behavioral Sciences

One Morton Dr Suite 500

University of Virginia, P.O. Box 800392

Charlottesville, VA 22908-0392


Telephone: (434) 924-5999 ; Email: [irbsbshelp@virginia.edu](mailto:irbsbshelp@virginia.edu)

Website: [www.virginia.edu/vpr/irb/sbs](http://www.virginia.edu/vpr/irb/sbs)

---

Revision date: 1/2/14

Page 2

IRB-SBS Office Use Only		
Protocol #	2013-0453-00	
Approved	from: 1/9/14	to: 1/8/15
SBS Staff		

**Agreement:**

I agree to participate in the research study described above, **under the condition that none of the sensors selected below will be deployed.**

Please select the sensors that you **DO NOT** want to be deployed in your home:


Bed room motion sensor  
Pressure sensors in bed  
Living room motion sensor  
Pressure sensors in living room chairs / couch / sofa  
TV sensor  
Dining room / area motion sensor  
Contact sensors in dining table / chairs  
Kitchen motion sensor  
Contact sensor in refrigerator  
Contact sensor in freezer  
Contact sensor in microwave  
Contact sensor in stove  
Contact sensor in oven  
Contact sensor in dishwasher  
Contact sensor in kitchen Cabinets  
Toilet motion sensor  
Contact sensor in toilet sink  
Contact sensor in shower  
Contact sensor in toilet Flush  
Microphone (it will not record anyone's voice, functionality describe above) in living room  
Microphone (it will not record anyone's voice, functionality describe above) in bed room(s)  
Microphone (it will not record anyone's voice, functionality describe above) in kitchen  
Contact sensor in doorways  
Pressure sensor in floor mats  
Contact sensor in washing machine  
Other (please mention): \_\_\_\_\_

Signature: \_\_\_\_\_ Date: \_\_\_\_\_

You will receive a copy of this form for your records.

Revision date: 1/2/14

Page 3

IRB-SBS Office Use Only		
Protocol #	2013-0453-00	
Approved	from: 1/9/14	to: 1/8/15
SBS Staff		

# Bibliography

- [1] Smart home sensor networks: A market dynamics report. <http://onworld.com/smarthomes/>, 2011.
- [2] Gilles Virone. Assessing everyday life behavioral rhythms for the older generation. *Pervasive and Mobile Computing*, 5(1), 2009.
- [3] Denis Elbert, Holger Storf, Michael Eisenbarth, Ö. Ünalán, and Mario Schmitt. An approach for detecting deviations in daily routine for long-term behavior analysis. In *PervasiveHealth*, 2011.
- [4] Mark R. Hodges, Ned L. Kirsch, Mark W. Newman, and Martha E. Pollack. Automatic assessment of cognitive impairment through electronic observation of object usage. In *Pervasive*, 2010.
- [5] Tim Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *UbiComp*, 2008.
- [6] Michael Buettner, Richa Prasad, Matthai Philipose, and David Wetherall. Recognizing daily activities with rfid-based sensors. In *UbiComp*, 2009.
- [7] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive*, 2004.
- [8] Tim van Kasteren, G. Englebienne, and Ben Krose. Activity recognition using semi-markov models on real world smart home datasets. *J. Ambient Intell. Smart Environ.*, 2(3), 2010.
- [9] Huiru Zheng, Haiying Wang, and Norman Black. Human activity detection in smart home environment with self- adaptive neural networks. In *ICNSC*, 2008.
- [10] Matthai Philipose, Kenneth P. Fishkin, Dieter Fox, Henry Kautz, Donald Patterson, and Mike Perkowitz. Guide: Towards understanding daily life via auto-identification and statistical analysis. In *Ubihealth*, 2003.
- [11] Emmanuel Munguia Tapia, Tanzeem Choudhury, and Matthai Philipose. Building reliable activity models using hierarchical shrinkage and mined ontology. In *Pervasive*, 2006.

- [12] Todor Dimitrov, Josef Pauli, and Edwin Naroska. Unsupervised recognition of adls. In *SETN*, 2010.
- [13] Yongkoo Han, Manhyung Han, Sungyoung Lee, A. M. Jehad Sarkar, and Young-Koo Lee. A framework for supervising lifestyle diseases using long-term activity monitoring. *Sensors*, 12(5), 2012.
- [14] Taketoshi Mori, Akinori Fujii, Masamichi Shimosaka, Hiroshi Noguchi, and Tomomasa Sato. Typical behavior patterns extraction and anomaly detection algorithm based on accumulated home sensor data. In *FGCN*, 2007.
- [15] Dorothy N. Monekosso and Paolo Remagnino. Anomalous behavior detection: Supporting independent living. In *Intelligent Environments*, Advanced Information and Knowledge Processing. SpringerLink, 2009.
- [16] A. Lotfi, C. Langensiepen, S. M. Mahmoud, and M. J. Akhlaghinia. Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behavior. *Journal of Ambient Intelligence and Humanized Computing*, 3(3), 2012.
- [17] Vikramaditya R. Jakkula, Diane J. Cook, and Aaron S. Crandall. Temporal pattern discovery for anomaly detection in smart homes. In *IE*, 2007.
- [18] Vikramaditya R. Jakkula and Diane J. Cook. Detecting anomalous sensor events in smart home data for enhancing the living experience. In *AAAI*, 2011.
- [19] Jianxin Wu, Adebola Osuntogun, Tanzeem Choudhury, Matthai Philipose, and James M. Rehg. A scalable approach to activity recognition based on object use. In *ICCV*, 2007.
- [20] Taketoshi Mori, Aritoki Takada, Yasuhiko Iwamura, and Tomomasa Sato. Automatic human life summarization system in sensory living space. In *International Conference on Systems, Man and Cybernetics*, 2004.
- [21] Dimitrios Lymberopoulos, Athanasios Bamis, and Andreas Savvides. Extracting spatiotemporal human activity patterns in assisted living using a home sensor network. In *PETRA*, 2008.
- [22] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48(5), 2009.
- [23] Beclose. <http://www.BeClose.com/>, 2014.
- [24] J.K. Aggarwal and M.S. Ryoo. Human activity analysis. *ACM Computing Surveys*, 43(3):1–43, April 2011.
- [25] Tao Gu, Shaxun Chen, Xianping Tao, and Jian Lu. An unsupervised approach to activity recognition and segmentation based on object-use fingerprints. *Data Knowl. Eng.*, 69(6), 2010.

- [26] Beth Logan, Jennifer Healey, Matthai Philipose, Emmanuel Munguia Tapia, and Stephen Intille. A long-term evaluation of sensing modalities for activity recognition. In *UbiComp*, 2007.
- [27] Tâm Huynh, Mario Fritz, and Bernt Schiele. Discovery of activity patterns using topic models. In *UbiComp*, 2008.
- [28] Microsoft speech api (sapi). [http://msdn.microsoft.com/en-us/library/ms723627\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/ms723627(v=vs.85).aspx), 2014.
- [29] Bor-Rong Chen, Geoffrey Peterson, Geoff Mainland, and Matt Welsh. Livenet: Using passive monitoring to reconstruct sensor network dynamics. In *Proceedings of the 4th IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2008.
- [30] Octav Chipara, Chenyang Lu, Thomas C. Bailey, and Gruia-Catalin Roman. Reliable clinical monitoring using wireless sensor networks: experiences in a step-down hospital unit. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2010.
- [31] Robert Dickerson, Eugenia Gorlin, and John Stankovic. Empath: A continuous remote emotional health monitoring system for depressive illness. In *Wireless Health*, 2011.
- [32] Timothy W. Hnat, Vijay Srinivasan, Jiakang Lu, Tamim I. Sookoor, Raymond Dawson, John Stankovic, and Kamin Whitehouse. The hitchhiker’s guide to successful residential sensing deployments. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2011.
- [33] Yongkoo Han, Manhyung Han, Sungyoung Lee, A. M. Jehad Sarkar, and Young-Koo Lee. Wireless sensing systems in clinical environments: Improving the efficiency of the patient monitoring process. *IEEE Engineering in Medicine and Biology Magazine*, 29(2), 2010.
- [34] Saad Arrabi and John Lach. Adaptive lossless compression in wireless body sensor networks. In *Proceedings of the Fourth International Conference on Body Area Networks (BodyNets)*, 2009.
- [35] Adam T. Barth, Mark A. Hanson, Harry C. Powell Jr., and John Lach. Tempo 3.1: A body area sensor network platform for continuous movement assessment. In *International Workshop on Wearable and Implantable Body Sensor Networks*, 2009.
- [36] Tia Gao, C. Pesto, L. Selavo, Yin Chen, Jeong Gil Ko, Jong Hyun Lim, A. Terzis, A. Watt, J. Jeng, Bor-Rong Chen, K. Lorincz, and M. Welsh. Wireless medical sensor networks in emergency response: Implementation and pilot results. In *IEEE Conference on Technologies for Homeland Security*, 2008.
- [37] Y. Han, M. Han, S. Lee, A. M. J. Sarkar, and Y. Lee. A framework for supervising lifestyle diseases using long-term activity monitoring. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2012.

- [38] M. A. Hanson, H. C. Powell, A. T. Barth, K. Ringgenberg, B. H. Calhoun, J. H. Aylor, and J. Lach. Body area sensor networks: Challenges and opportunities. *IEEE Computer*, 42(1), 2009.
- [39] Fahd Albinali, Nigel Davies, and Adrian Friday. Structural learning of activities from sparse datasets. In *PerCom*, 2007.
- [40] Derek Hao Hu and Qiang Yang. Cigar: concurrent and interleaving goal and activity recognition. In *Proceedings of the 23rd conference on Artificial intelligence (AAAI)*, 2008.
- [41] Joseph Modayil, Tongxin Bai, and Henry Kautz. Improving the recognition of interleaved activities. In *Proceedings of the 10th international conference on Ubiquitous computing (UbiComp)*, 2008.
- [42] K. M. Kitani and Y. Sato. Recognizing overlapped human activities from a sequence of primitive actions via deleted interpolation. *International Journal of Pattern Recognition and Artificial Intelligence*, 2008.
- [43] T. Gu, L. Wang, Z. Wu, X. Tao, and J. Lu. epsicar: A pattern mining approach to sensor-based human activity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 23(6), 2011.
- [44] Rim Helaoui, Mathias Niepert, and Heiner Stuckenschmidt. Recognizing interleaved and concurrent activities: A statistical-relational approach. In *Proceedings of the 2011 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2011.
- [45] S. Inoue. Challenges for activity recognition with real data. In *Proceedings of the 13th international conference on Ubiquitous computing (UbiComp)*, 2011.
- [46] Tim van Kasteren, G. Englebienne, and B. Kröse. Transferring knowledge of activity recognition across sensor networks. In *Pervasive*, 2010.
- [47] Vijay Srinivasan, John Stankovic, and Kamin Whitehouse. Protecting your daily in-home activity information from a wireless snooping attack. In *UbiComp*, 2008.
- [48] T. S. Barger, Donald E. Brown, and Majd Alwan. Health-status monitoring through analysis of behavioral patterns. *IEEE Transactions on Systems, Man, and Cybernetics - Part A*, 35(1), 2005.
- [49] Maja Stikic and Bernt Schiele. Activity recognition from sparsely labeled data using multi-instance learning. In *LoCA*, 2009.
- [50] Pang Wu, H. Peng, Jiang Zhu, and Ying Zhang. Senscare: Semi-automatic activity summarization system for elderly care. In *MobiCASE*, 2011.
- [51] Brent Longstaff, Sasank Reddy, and Deborah Estrin. Improving activity classification for health applications on mobile devices using active and semi-supervised learning. In *PervasiveHealth*, 2010.

- [52] M. Novak, M. Binas, and F. Jakab. Unobtrusive anomaly detection in presence of elderly in a smart-home environment. In *ELEKTRO*, 2012.
- [53] M. Novak, Jakab F., and Lain L. Anomaly detection in user daily patterns in smart-home environment. *JSHI*, 3(6), 2013.
- [54] Derek T. Anderson, Maria Ros, James M. Keller, Manuel P. Cuellar, Mihail Popescu, Miguel Delgado, and Amparo Vila. Similarity measure for anomaly detection and comparing human behaviors. *Int. J. Intell. Syst.*, 27(8):733–756, 2012.
- [55] P. Sukanya and K. S. Gayathri. An unsupervised pattern clustering approach for identifying abnormal user behaviors in smart home. *International Journal of Computer Science and Network*, 2(3), 2013.
- [56] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. Technical report, Department of Computer Science and Engineering, University of Minnesota, 2007.
- [57] Orna Raz, Philip Koopman, and Mary Shaw. Semantic anomaly detection in online data sources. In *ICSE*, 2002.
- [58] J.G. Ko, Chenyang Lu, M.B. Srivastava, J.A. Stankovic, Andreas Terzis, and Matt Welsh. Wireless sensor networks for healthcare. 98(11):1947–1960, 2010.
- [59] Julie a. Kientz, Shwetak N. Patel, Brian Jones, Ed Price, Elizabeth D. Mynatt, and Gregory D. Abowd. The Georgia Tech aware home. *Proceeding of the twenty-sixth annual CHI conference extended abstracts on Human factors in computing systems - CHI '08*, page 3675, 2008.
- [60] Sumi Helal, William Mann, Hicham El-Zabadani, Jeffrey King, Youssef Kaddoura, and Erwin Jansen. The Gator Tech smart house: a programmable pervasive space. *IEEE Computer*, 2005.
- [61] Marco Messina, Yen Yang Lim, Elaine Lawrence, Don Martin, and Frank Kargl. Implementing and Validating an Environmental and Health Monitoring System. In *Fifth International Conference on Information Technology: New Generations (ITNG 2008)*, pages 994–999. Ieee, April 2008.
- [62] Anthony Wood, John Stankovic, Gilles Virone, Leo Selavo, Zhimin He, Qiuhua Cao, Thao Doan, Yafeng Wu, Lei Fang, and Radu Stoleru. Context-aware wireless sensor networks for assisted living and residential monitoring. *IEEE Network*, 22(4):26–33, July 2008.
- [63] Jiakang Lu, Tamim Sookoor, Vijay Srinivasan, Ge Gao, Brian Holben, John Stankovic, Eric Field, and Kamin Whitehouse. The smart thermostat: Using occupancy sensors to save energy in homes. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 211–224, 2010.

- [64] Diane Doumas. Advanced sensor technology enhances operational effectiveness in senior living. Technical report, WellAware, Glen Allen, VA, 2011.
- [65] BeClose. Return on Investment of In-Home Assistive Technology. Technical Report 866, BeClose, Vienna, VA, 2012.
- [66] Philips lifeline with auto alert systems. <http://www.lifelinesys.com/content/lifeline-products/auto-alert>, 2014.
- [67] Intel-ge care innovations: Care innovations. <http://www.careinnovations.com/products/quietcare-assisted-living-technology>, 2014.
- [68] Cisco expert on demand solution. [http://www.cisco.com/web/strategy/healthcare/solution\\_CollaborativeCare.html](http://www.cisco.com/web/strategy/healthcare/solution_CollaborativeCare.html), 2014.
- [69] Cisco healthpresence. [http://www.cisco.com/web/strategy/healthcare/cisco\\_healthpresence\\_solution.html](http://www.cisco.com/web/strategy/healthcare/cisco_healthpresence_solution.html), 2014.
- [70] Hobo data loggers. <http://www.onsetcomp.com/>, 2014.
- [71] Serguei A. Mokhov. Evolution of marf and its nlp framework. In *Proceedings of the Third C\* Conference on Computer Science and Software Engineering*, C3S2E '10, 2010.
- [72] Robert Dickerson, Enamul Hoque, Philip Asare, , Shahriar Nirjon, and John Stankovic. Resonate: Improving speech classification in home environments using reverberant environment simulation. In *IPSN*, 2014.
- [73] Burr Settles. Active learning literature survey. Computer sciences technical report, University of Wisconsin–Madison, 2009.
- [74] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
- [75] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.
- [76] *The Elements of Statistical Learning*, chapter 14.3.12 Hierarchical clustering. Springer, 2009.
- [77] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE*, 1995.
- [78] Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.