

# **A System for Recognizing Complete and Partial in Home Activities and Monitoring Activity Quality**

---

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

by

Ifat Emi

August 2019



# *Abstract*

## **A System for Recognizing Complete and Partial in Home Activities and Monitoring Activity Quality**


by Ifat Afrin Emi

Affordable home health care systems are extremely important for early diagnosis of disease and to track patient recovery. As part of these systems, the ADL score calculated from activities of daily livings (ADL) and instrumented activities of daily livings (IADL) provide valuable statistics about the functional and cognitive ability of patients and elderly citizens, which is required for deciding treatments and services. However, most of the existing available systems impose constraints on sensor values, the types of detected activities (no parallel/interleaved/joint activities), or the number of users, which reduce the robustness of the system in the real world settings. Moreover, in order to provide a holistic solution for monitoring activities, it is important not only to provide information about complete activities but also detect attempted incomplete/partial activity instances and report the errors.

In this dissertation, we have designed, implemented and evaluated a novel activity recognition and person identification system named SARRIMA, a rule-based general framework *QuActive* for modeling activity, a system for recognizing activity steps, and an activity quality monitoring system based on the *QuActive* framework for identifying partial/incorrectly performed activities and finding the errors within the activities (missing steps, wrong steps, wrong orders, and delays within and between activity steps). We show that by incorporating time difference while segmenting the occupancy episodes and feeding those segments in *Apriori* rule association machine learning technique, the system is able to work in homes with multiple people and detect interleaved and parallel activities with higher performance than supervised machine learning systems. Moreover, we provide proof that by modeling activities in terms of activity steps and using grammar rules, it is possible to capture the different variation of the same activity. The grammar rules also enable to find the missing steps or errors due to performing wrong activity steps or doing them in the wrong order. Finally, we provide evidence that using user's activity history and occupancy correlation information, it is possible to infer which user performed an activity event for a significant number (the number varies in different datasets based on user lifestyle and sensor settings) of activity instances even when there is no direct person identification information available. Our evaluation in different public datasets and collected data from lab shows that our system performs better than the state-of-the-art activity recognition systems and is able to provide activity quality information of both complete and incomplete activities.

## APPROVAL SHEET

This Dissertation  
is submitted in partial fulfillment of the requirements  
for the degree of  
Doctor of Philosophy

Author Signature: 

This Dissertation has been read and approved by the examining committee:

Advisor: John A. Stankovic

Committee Member: Hongning Wand

Committee Member: John Lach

Committee Member: Laura Barnes

Committee Member: Yuan Tian

Committee Member: \_\_\_\_\_

Accepted for the School of Engineering and Applied Science:



Craig H. Benson, School of Engineering and Applied Science

August 2019

## *Acknowledgements*

First and foremost, I want to acknowledge my Ph.D. advisor John A. Stankovic, who has guided me at every step of this research with his extraordinary insight, constant encouragement, and endless patience. He has taught me how to conduct research, how to think critically, how to express clearly, as well as, inspired me to be more confident, hardworking, effective, and positive. I am extremely fortunate to have him as my Adviser and learn from the best.

I am very fortunate to have a great dissertation committee with each member specialized in the different technical areas relevant to my work. The graduate Machine Learning course taught by Professor Laura Burnes has equipped me with the tools and knowledge that were extremely useful in my research. While working on several projects with Professor John Lack, I learned the importance of realism and have realistic expectations and assumptions on merging the gap between research experiments in lab vs. in the wild. Professor Hongning and Professor Yuan Tian were extremely kind and always eager to help with any problem.

I am thankful to all my lab and research mates. When I first joined the group Enamul, Rob, Nirjon, and Munir were very welcoming and mentored me by giving me helpful advice and showing the resources necessary for conducting research. I am extremely grateful to my teammates in different projects - Abu, Sarah, Asif, Mohsin, Meiyi, Sirat, Zeya, and Ridwan. Especially, collaboration with Abu has made a lot of challenges easier and work more exciting.

I want to take this opportunity to thank all the scientists and researchers on top of whose research I have built my work. Especially, those who encouraged us to explore this topic by making their datasets public. Nonetheless, the experience from data collection is extremely valuable in finding new insights and a higher understanding of the factors related to realism. Therefore, this research would not have been possible without all the volunteers who have agreed to wear sensors and use our system and let us collect the data.

Ph.D., i.e., the Degree in Philosophy is not only about performing experiments and writing papers, but also a commitment and journey in life toward self-exploration. It is not possible to overcome the ups and downs of this life without the help and support from friends and family. My eternal gratitude to my family who has always been my source of energy and strength. My father has installed the love of learning in me by explaining science like a story. On the other hand, the patience and kindness of my mother made me endure the moments when life was not easy. My siblings are my most trusted companion and best friends whose faith in my ability makes me want to be a better person. I would also like to thank my uncles, aunts, relative, and cousins, especially Rinti, Shejuty, Faisal, Nobel, Tusar, Sajib, Shoumik, and nephew Aronnok in Bangladesh who have kept me updated with my country and made me feel close to home.

I am very thankful for the wonderful Bangladeshi Community at Charlottesville. Anindya vai, Tania Apa, Tonima Apu, and Samia Vabi have all gave and continue to give guidance in different situations and I am very grateful to them. Thank you, Liza, Nazia, Sarah, Noushin, Jisa, Tani, Nishat, Bobby, Anonya, Saoda, Sadia, Tonny, Sathi, Tumpa, Fabiha, Hemlata, and everyone for your delicious food and the cheerful evenings. Love to Abrar, Sophie, Airah, Arish, Merab, Sreyash, Sahitto, and all my kid friends for bringing joy and happiness to my life. I am grateful to my dear friends all over the world, Mily, Bithi, Tonny, Abir, Mahmud, Suman, Anika, Baishakhi, Archi, Shanta, Yamina, Tania, Tony, Lopa, and Simi Apu who have always been my side despite the distance.

Last but not least, thank you, Allah, for controlling the variables of my life in such a way that made it possible for me to reach where I am now.



# Contents

<b>Abstract</b>	<b>iii</b>
<b>Acknowledgements</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Challenges and Scope . . . . .	3
1.2 Thesis Statement and System Overview . . . . .	5
1.2.1 Statement . . . . .	5
1.2.2 System Overview . . . . .	5
1.3 Contributions . . . . .	7
1.4 Thesis Outline . . . . .	8
<b>2 Related Work</b>	<b>11</b>
2.1 Home Monitoring Systems . . . . .	11
2.1.1 Research Testbeds . . . . .	11
2.1.2 Commercial Products . . . . .	12
2.2 Research in Activity Detection and Recognition . . . . .	12
2.2.1 Different Sensing Platforms . . . . .	12
Wearable Sensing . . . . .	12
Infrastructure based Sensing . . . . .	12
Image based Recognition . . . . .	13
Wireless Sensor Network . . . . .	13
2.2.2 Different Algorithms and Techniques . . . . .	13
Machine Learning Methods . . . . .	13
Grammar, Ontology, and Rule based Activity Recognition Systems . . . . .	14
2.3 Research in Detecting and Recognizing Activity Steps . . . . .	14
2.4 Notification and Reminder Systems . . . . .	14
2.5 Research in Activity Quality Monitoring . . . . .	15
<b>3 Background and Terminology</b>	<b>17</b>
3.1 Activity Recognition . . . . .	17
3.1.1 Physical Activity vs. Daily Living Activities . . . . .	17
Physical Activities and Gestures . . . . .	17
Activities Daily Living and Instrumental Activities Daily Living . . . . .	18
3.1.2 Activity Types: Sequential, Interleaved, and Parallel . . . . .	18
Sequential Activities . . . . .	18
Interleaved Activities . . . . .	18
Parallel Activities . . . . .	18
3.1.3 Activity Steps . . . . .	19

	Atomic Activity . . . . .	19
	Sub Activity and complex activity . . . . .	19
	Micro-activity . . . . .	19
3.2	Grammar and Rules . . . . .	20
3.2.1	AI: Rule-based Systems . . . . .	20
3.2.2	Formal Grammar . . . . .	21
	Syntax of Grammars . . . . .	21
3.3	Algorithms . . . . .	21
3.3.1	Frequent Itemset and Association Rule Mining . . . . .	22
	<i>Apriori</i> Algorithm . . . . .	22
3.3.2	Pattern matching with Regular Expression . . . . .	23
3.3.3	Machine Learning Algorithms . . . . .	23
3.4	Hardware and Software . . . . .	25
3.4.1	Sensing Modalities . . . . .	25
	In situ sensors . . . . .	25
	Wearable . . . . .	25
3.4.2	Software Environments . . . . .	25
3.4.3	Netbeans IDE with Java 1.8 . . . . .	26
3.4.4	Matlab . . . . .	26
3.4.5	Weka . . . . .	27
<b>4</b>	<b>SARRIMA: Smart ADL Recognizer and Resident Identifier in Multi-resident Accommodations</b>	<b>29</b>
4.1	Motivation . . . . .	29
4.2	Contributions . . . . .	30
4.3	System Assumptions and Overview . . . . .	31
4.3.1	Sensing Layer . . . . .	32
4.3.2	Occupancy Episodes . . . . .	33
4.3.3	Defining ADL Classes . . . . .	33
4.3.4	Activity Recognition . . . . .	34
4.3.5	Person Identification . . . . .	34
	User Differentiating Features . . . . .	34
	Relative Position of the Users . . . . .	34
	The Relationship among the Recognized Activities. . . . .	36
4.3.6	Parallel and Overlapping Activities . . . . .	36
4.4	Evaluation and Discussion . . . . .	37
4.4.1	Results: Activity Detection . . . . .	37
4.4.2	Results: Person Identification . . . . .	38
4.5	Conclusion . . . . .	41
<b>5</b>	<b>Activity Step Recognition: Challenges and Lessons Learned</b>	<b>43</b>
5.1	Introduction . . . . .	43
5.2	Challenges . . . . .	44
5.2.1	Activity Step Definition . . . . .	44
5.2.2	Sensing Technology . . . . .	45



	In-situ Sensing Platform . . . . .	45
	Hybrid Sensing Platform: RFID . . . . .	46
	Wearable Sensing Platform: Smartwatch . . . . .	47
5.2.3	Number of Classes of Activity Step . . . . .	48
5.2.4	Same vs. Similar Activity Steps . . . . .	48
5.2.5	Data Processing . . . . .	48
	Segmentation . . . . .	48
	Classification . . . . .	50
	Energy . . . . .	50
5.2.6	Human Factor . . . . .	50
	Biometric difference . . . . .	50
	Hand motion . . . . .	50
	User Energy Level . . . . .	51
	Personal Preference or Habit . . . . .	51
	Parallel Actions . . . . .	51
5.2.7	Appliance Arrangement . . . . .	51
5.3	Ideas and Lessons Learned . . . . .	52
5.3.1	Scope of the Problem . . . . .	52
5.3.2	Importance of Sensing Platform . . . . .	52
5.3.3	Necessity of Clean Data . . . . .	52
5.3.4	Advantage of Sensor Fusion . . . . .	53
	Data Segmentation . . . . .	53
	Class Reduction . . . . .	53
	Energy Salvation of Wearable . . . . .	53
<b>6</b>	<b>Activity Quality Framework</b> . . . . .	<b>55</b>
6.1	Motivation and Challenges . . . . .	55
6.2	Contributions . . . . .	57
6.3	QuActive Framework . . . . .	57
6.3.1	Definition and Properties of Activity Steps . . . . .	58
6.3.2	Timed Probabilistic Context Free Grammar . . . . .	58
6.3.3	TPCFG for Detecting Activity . . . . .	59
6.3.4	Example Grammar and Parse Tree . . . . .	61
6.4	System Design . . . . .	62
6.4.1	Sensing Layer . . . . .	62
6.4.2	Event Layer . . . . .	62
6.4.3	Activity Step Layer . . . . .	62
6.4.4	Activity Layer . . . . .	63
6.4.5	Feedback - A Notification Layer . . . . .	64
6.5	Evaluation and Results . . . . .	65
6.5.1	Datasets . . . . .	65
	Interleaved ADL . . . . .	65
	Multiresident ADL . . . . .	65
	Cognitive assessment activity data . . . . .	65
6.5.2	Data Collection . . . . .	66
6.5.3	Results and Discussions . . . . .	68
	Evaluation on the Datasets . . . . .	68

	Evaluation on Experimental Data . . . . .	69
	Notification Subsystem . . . . .	71
6.6	Conclusions . . . . .	71
<b>7</b>	<b>Activity Quality Monitoring</b>	<b>73</b>
7.1	Motivation . . . . .	73
7.2	Contributions . . . . .	75
7.3	Problem, Background & Scope . . . . .	75
7.3.1	Problem Description . . . . .	75
7.3.2	Activity Grammar . . . . .	76
7.3.3	System Inputs and Outputs . . . . .	78
	Knowledge base: What does the system know? . . . . .	78
	Inputs . . . . .	79
	Preprocessing Input: Activity Episodes . . . . .	79
	Outputs . . . . .	79
7.3.4	Number of people and activities . . . . .	80
	Number of Users . . . . .	80
	Activity Types and Occurrence Boundary . . . . .	80
7.4	Solution . . . . .	80
7.4.1	Selection Phase . . . . .	81
	Searching . . . . .	81
	Filtering and Selecting . . . . .	82
	Separating steps of Different Activities . . . . .	82
7.4.2	Decision and Validation Phase . . . . .	83
7.4.3	Result Adjustment and Storing Phase . . . . .	84
	Error Adjustment . . . . .	84
	Timing information for Quality . . . . .	84
	Storage for Efficiency and Statistics . . . . .	85
7.4.4	Verification by Expert . . . . .	85
7.5	Implementation Details . . . . .	85
7.5.1	Entity Classes . . . . .	85
	Utility Classes . . . . .	88
7.5.2	Major Functions . . . . .	88
	Validation: Pattern Matching . . . . .	88
	Storing and Searching . . . . .	90
7.6	Evaluation and Discussion . . . . .	91
7.6.1	CASAS Dataset: Cognitive Assessment Activity Data . . . . .	91
	Results: Recognizing Complete and Incomplete Activities . . . . .	93
	Discussion . . . . .	94
7.6.2	Collected Data . . . . .	95
	Results: Recognizing Complete Activities . . . . .	96
	Results: Recognizing Activities with Errors . . . . .	97
<b>8</b>	<b>Notification System</b>	<b>99</b>
8.1	Motivation . . . . .	99
8.2	Contributions . . . . .	101
8.3	System Description . . . . .	101

8.3.1	Operating Script . . . . .	101
8.3.2	Reminder Life Cycle . . . . .	102
8.3.3	Reminder Session . . . . .	103
8.3.4	System Architecture . . . . .	103
8.4	Solutions . . . . .	104
8.4.1	Alerts . . . . .	104
8.4.2	User Interaction . . . . .	104
	Interactions through Touch Screen . . . . .	104
	Interaction through Voice Commands . . . . .	105
8.4.3	Voice Command Recognition . . . . .	106
8.4.4	Cloud Connectivity . . . . .	108
8.4.5	Energy Efficiency . . . . .	109
8.5	Experiment . . . . .	109
8.5.1	Data Collection . . . . .	109
8.5.2	Analysis . . . . .	110
8.6	Discussion . . . . .	111
8.7	Conclusion . . . . .	112
<b>9</b>	<b>Conclusion</b> . . . . .	<b>113</b>
9.1	Summary . . . . .	113
9.1.1	Research on Activity Recognition . . . . .	113
	Novel Segmentation Algorithm . . . . .	113
	Techniques for Person Identification without Specialized Sensors . . . . .	114
	A Novel Framework for Activity Recognition based on Grammar . . . . .	114
9.1.2	Research on Activity Quality Monitoring . . . . .	114
	Defining Quality Parameters and Developing a System for Quality Mon- itoring . . . . .	114
	Comprehensive Studies on Understanding Activity Steps in Daily Life . . . . .	115
9.2	Future Work . . . . .	115
9.2.1	Studies in Detection and Recognition of Activity Steps . . . . .	115
9.2.2	Real World Deployment with Alzheimer's' Patients . . . . .	115
9.2.3	Grammar Learning from Activity Data . . . . .	116
9.2.4	Multiple People . . . . .	116
9.2.5	System Robustness . . . . .	116
	Missing Data vs Missing Activity Steps . . . . .	116
	Real Time Constraint . . . . .	116
	Handling Exceptional Cases . . . . .	117
9.2.6	Applications in Related Areas . . . . .	117
	Emergency Medical Responders (EMT) . . . . .	117
	Nursing Activity Monitoring . . . . .	117
	Surgical Procedure Monitoring . . . . .	117
	Worker Training . . . . .	118
	<b>Bibliography</b> . . . . .	<b>119</b>



# List of Figures

1.1	Conceptual integration of all the systems: SARRIMA, Activity Step Recognizer, QuActive Framework and System for monitoring the quality . . . . .	6
3.1	Action, environment, and activity . . . . .	17
3.2	. . . . .	19
3.3	Four types of grammar defined by Noam Chomsky . . . . .	21
3.4	<i>A priori</i> itemset mining and rule learning algorithm . . . . .	23
3.5	Regular expression matching with kleen star . . . . .	24
3.6	The first row shows Aeotec z-wave door sensor, motion sensor, and modem respectively by Aeon Labs. The next row shows Samsung Gear S smartwatch and an inertial measurement unit (IMU) with accelerometer, gyroscope, and magnetometer within the watch. . . . .	26
4.1	Overall system architecture of SARRIMA . . . . .	32
4.2	Training Framework for Defining Activity Classes . . . . .	32
4.3	Creating occupancy sessions . . . . .	35
4.4	Percentage of activity instances of Activity Classes recognized correctly in CASAS Spring 2009, CASAS Summer 2010, ARAS House A, and ARAS House B (time_threshold = 2 minutes) . . . . .	37
4.5	False positive of reported activity instances ARAS (House B) (time_threshold = 2 minutes) . . . . .	38
4.6	Accuracy of user identification from behavioral difference for particular activity classes where difference is observed in users way of doing the activities (ARAS Dataset). . . . .	39
4.7	Person Identification Accuracy (threshold=2 min) (a) House A (b) House B . . .	40
5.1	The above images show time series data of accelerometer x-axis and gyroscope x-axis corresponding to different gestures. The snapshots are taken from the 'Chonoviz' visualization software tool. The x-axis labels in each image show time in 1s intervals (except (l) which shows in 0.5s intervals). The y-axis show the normalized value of the accelerometer and the gyroscope in a fitted zoomed position. . . . .	49
6.1	Example parse trees showing different ways of performing the same activity ('Making Coffee'). . . . .	61
6.2	System Architecture for detecting and recognizing activity steps and high level activities based on QuActive Framework. . . . .	63

6.3	Recognizing interleaved (high level) activities using HMM, NBC [72], QuActive with and without (QuActive') location information incorporated in the grammar. The figure shows the percent of sensor data labeled correctly with respect to ground truth labeling. . . . .	67
6.4	Average accuracy in recognizing instances of independent, parallel, and joint activities using HMM [73] and QuActive on a multiresident dataset. . . . .	68
6.5	The average number of missing steps in performing activities by healthy, mildly cognitively impaired, and dementia patients. The stripped column values considers the effect of missing activities in calculating missing steps and the solid column values disregards missing activities. . . . .	69
6.6	Filtering the falsely recognized activity steps in subsequent layers of the system.	70
7.1	Complexity of solution space (identifying activity or activity attempt) with increasing difficulty of problem space (presence of different types of error) . . . . .	74
7.2	The overall process of recognizing activity instances and finding the quality parameters. . . . .	81
7.3	Definition of different entity classes . . . . .	87
7.4	Percentage of activities correctly recognized (true positive) and percentage of quality correctly assessed . . . . .	94
7.5	Percentage of incomplete activities correctly recognized (true positive) when steps are missing . . . . .	96
8.1	Example of a reminder entry in the Operating Script (OS) in JSON format . . . . .	102
8.2	Life cycle of a reminder . . . . .	103
8.3	System Architecture . . . . .	104
8.4	Some example symbols for a reminder (a) one pill (b) inhaler (c) two pills (d) two pills and the inhaler . . . . .	105
8.5	Example of multiple display pages for a reminder . . . . .	105
8.6	Voice command recognition process . . . . .	108
8.7	Error rates in recognizing intended voice commands with speech recognizer only (No training) and with using speech recognizer with leave-one-person-out (LOPO) training . . . . .	110
8.8	Reduction of error rates with personalized training . . . . .	111

# List of Tables

3.1	Example of Activities of Daily Livings and Instrumental Activities of Daily Livings	18
4.1	Effect of user identification using behavioral information and episode linking (threshold = 3 min) for a single day in ARAS House B . . . . .	39
6.1	TPCFG for activity ‘Making Coffee’ . . . . .	60
6.2	Example of activity steps within the activity ‘Preparing soup’ instructed to be performed by a user [72], [73]. . . . .	65
6.3	Activity List used for evaluation in different datasets and the collected data. . . .	66
6.4	Average performance of QuActive in recognizing activity instances from all users	71
7.1	Functionality of meta characters in extended regular expressions. . . . .	78
7.2	. . . . .	86
7.3	The ADL/IADL classes in the CASAS Dataset with cognitive assessment activity data. . . . .	91
7.4	Example of activity steps and some grammar rules of the activity ‘Preparing Oatmeal’ instructed to be performed by a user. . . . .	92
7.5	ADL/IADL classes observed in the Collected Dataset . . . . .	95
8.1	Examples of Command keywords, their purposes, regular expressions and related commands . . . . .	107





# List of Abbreviations

<b>ADL</b>	<b>Activities of Daily Living</b>
<b>IADL</b>	<b>Intrumental Activities of Daily Living</b>
<b>RE</b>	<b>Regular Expression</b>
<b>ERE</b>	<b>Extended Regular Expression</b>
<b>ML</b>	<b>Machine Learning</b>
<b>TPCFG</b>	<b>Timed Probabilistic Context Free Grammar</b>



*To my dear parents Fazilatun Nessa and Md. Nurul Houqe  
for their love, guidance, and support.*

*To my beloved siblings Farzana Afrin Hoque Jhimi and Md. Mahmudul Hoque Shounak  
for always believing in me.*



## Chapter 1

# Introduction

U.S. Health Care Costs Skyrocketed to \$3.65 Trillion in 2018, and according to the latest analysis from U.S. federal government the expenditure will get even worse in future years [1]. The major portion of this expenditure consists of hospital bills, physicians and clinic checkup, and drugs; whereas only 3% goes to home health care. This is very unfortunate, since by tracking the progression of sickness many deaths could be prevented as the top 9 leading causes of death in USA is due to some type of disease. Many of these chronic diseases are prevalent in elderly citizens. According to government profile [2], 28% of older American citizens live alone and 57% live with their spouse. Often, the spouse is the primary caregiver although the spouse himself/herself needs care. Having a 24 hour care provider is not possible for most families. Although there are home health care agencies that keep track of these people, their traditional procedure is one weekly call and a single monthly visit. Therefore, most of the time these elderly people are on their own. Even when elderly citizens or younger patients live with other family members, the adults have to go out for work and during major portion of the day there is no one to monitor them. Therefore, the importance of home health monitoring system is crucial in our community. With the advancement of technology, cheaper, and better sensors are available in the market, and new and efficient algorithms are being discovered everyday. Thus, we have the chance for making affordable home health care system for mass population and promise them a better future.

Most of the commercially available home monitoring systems today are still quite simple. Most of them involve fall detection from motion sensors or just pressing a button which automatically calls for help. Despite being simplistic in nature, these systems can play vital role if the person is mostly physically capable and has cognitive awareness to act upon in case of emergency. However, the systems cannot track the progression of disease or be effective when the person is hurt/unable to act for any reason. On the other hand, due to the benefits of health monitoring systems, the research community has explored this field from different directions. A major portion of this research focuses on detecting the in activities of daily living (ADL) for self-care and instrumental activities of daily living (IADL) for fundamental livings. The ability to perform ADLs and IADLs successfully is considered as a major criterion to access the condition of stroke patients and patients suffering from depression, Alzheimer, orthopedic, neurological or sensory deficits [3]–[5]. Besides, in the case of older citizens, these criteria have been found to be significant predictors of admission to a nursing home, use of paid home care, use of hospital services, use of physician services, insurance coverage, and mortality [6]. According to government profile of older Americans [2], on average 92% of the institutionalized and 40% of the non-institutionalized older citizens have difficulty in performing one or more ADLs and this percentage becomes higher as they grow older. Therefore, detecting and recognizing ADLs are important for detecting early symptoms of disease, the improvement of access to prescribed medication, providing exact medical history to physicians, and as an important

preliminary step in systems for assistance in performing ADLs. In this thesis, our major goal is to build a system that detects and recognizes in home activities, can be deployed in home, and perform better than the state-of-the art systems.

Many current activity recognition systems [7], [8] make too many simplifying assumptions about the environment and the number of users that either limit the types of recognized activities, or is tailored to perform well in simple situations such as single person homes and with no concurrent activities [9], [10]. It is necessary to consider interleaved, parallel, and co-operative activities for more robust and realistic activity recognition that also perform accurately in presence of more than one individual. The presence of multiple people creates complexities since the number of overlapping and parallel activities increase. It becomes challenging to find the correlation between sensors to specific activity, since a sensor can be triggered by any of the parallel activities. Again, difficulty arises because each person has their own way of performing an activity [11]. Therefore, as part of this thesis we investigate how modifying existing algorithm enables a system to operate in home with multiple persons. We also built our systems to be general so that in is able to recognize activities performed sequentially, intertwined, or parallelly.

In order to provide sufficient information about ADLs/IADLs, we need to look into the details of the activity process. To show the significance of the monitoring activity process, the researchers [12] performed an experiment where they brought healthy people, patients with mild cognitive impairment, and Alzheimer patients in lab and asked them to perform a series of tasks where each task constitutes of specific order of steps. They found activities of daily living (ADL) as a good predictor of early detection of cognitive impairment as people with higher degree of dementia tend to leave out more steps and/or make more mistakes in the order of steps. Alzheimer is one of the leading disease to death and an estimated 5.7 million Americans of all ages are living with Alzheimer's dementia currently at USA. Since it does not have any cure and can only be slowed down with early diagnosis and proper monitoring, we focused our thesis to find quality of activities from Alzheimer's patients ADL and IADL. Although, many researchers highlight the importance of behavior detection in homes and many claims to have indirect advantage of providing ADL statistics, as far as we know there has not been any system directly focused on this particular problem of tracking activity steps. Behavior monitoring research is available on specific activities such as observing sleeping patterns, or estimating locomotive activeness from motion sensors We also found work on closely related field of anomaly detection where the system is trained to observe a single residence activity and find specific patterns related to time, duration, and correlation among the performed activities. Later, any activity data that does not match with those patterns are reported as having an anomaly. However, none of the current activity recognition systems identify partially completed activities or the missing steps in the overall activity process, but rather recognize whether an activity has occurred or not. Again, finding activity quality in home monitoring is not as straight forward as the experiment shown in lab [13], since there is no fixed sequence of steps for activities, and a particular activity can be performed in a number of possible ways. Again, since some activity steps are optional or dependent on the preference of the user, it is not easy finding the actual missing steps as those steps might be omitted intentionally. Another problem is that the steps of an activity are not as clearly defined in the real world. Therefore, in this thesis we look into the problem of detecting activity steps from sensors and wearables, and define the relation of steps to activity. We also perform detail analysis on the quality of performed activities and find whether steps are missing/wrong/out of order as well as suggest the closest match having the correct step sequence.

Another important task of in home activity monitoring system is to identify which person performed the activity, since we would like to know the activity quality of the person of interest for whom the system would be deployed. Although, person identification is straightforward when using wearable sensors for activity recognition, not all elderly people or patients complies carrying additional devices while performing all the activities. Therefore, as part of our thesis we also looked in to person identification problem when only in situ sensors are used. We studied the limitation of using only binary sensors and how much improvement is possible when considering activity history data or limited number of specialized sensors.

In most existing monitoring system, the resulting information is not acted upon in any direct or real-time manner. However, by more intimately bringing the human into a feedback loop, there is great potential to use interventions and notifications to improve human activities. For example, by reporting emergency situations, such as *'You forgot to turn off the stove'*, the home health care systems can keep patients safe. Also, by reminding later steps of an activity when the patient forgets, the system can assist up to certain extent. Therefore, we also examine how to integrate a static reminder system Medrem [14] in our activity recognition and quality monitoring system. We inspect which additional parameters are required for event based notification and how to categorize the notification that ensures safety without creating annoyance, and serves the user in best possible way.

## 1.1 Challenges and Scope

In this section, we address a number of challenges in building a system for monitoring the activity quality and recognizing the activity. Although each of the later chapters provides additional details related to particular problem, we briefly summarize some of the key issues here.

- First of all, what is the quality of an activity? In this thesis, we are only interested in the quality of ADLs and IADLs performed in a home setting. When a physician determines that activity quality of a patient has degraded, he makes judgement based on many complicated factors, clinical knowledge, and applying judgment based on both logical thinking and intuition. However, how can a system know what the activity quality is? What are the parameters? How can it be measured? In order to address this challenge, we looked in to the literature from clinical community and came up with two major elements as quality metrics, activity steps and time. An anomaly on any or both of these factors are considered important in evaluating the quality of an activity. Most importantly, both of these elements can be measured by sensing devices.
- In order to address the first challenge we assume that all activities are composed of some specific steps. The second question is how to define an activity step? Within an activity where does each step start and where does it end? How fine grained should the steps be? The details of this answer is given in section 3.2. Here, we would like to mention the two key requirement for defining activity steps in our thesis,
  - Each activity step should be detectable by the sensing system.
  - Activity steps are considered as a unit, i.e., a step is either done or not done, but cannot be half done.
- Third, how to model the activity process in terms of activity steps. The activity process also varies depending on person, environment, or situation. Different activities often

have similar steps, and steps performed in a different order might result in the same or a different activity. Thus, the process of mapping steps to distinct activities that are capable of handling these variations is vital. Another challenge is addressing the deviation from usual activity processes. For example, if a certain step is missing or performed out of order, then is the activity incomplete, wrongly performed, or still a valid activity performed in a different way? It is a question on how to keep a general structure of a particular activity which is performed in different ways? (Chapter 6)

- How to identify the prospective/incomplete activity when one or more steps are missing, or performed out of order? How can we use the activity recognition framework for monitoring the quality of an activity? How to find which steps are missing or done out of order? What happens if we consider interleaved and parallel activities and steps from different activity occurring between the steps of the particular activity? What role does the timing parameters play in activity quality? Depending on different activity types, what modifications/considerations are required? For example, delay between steps within a single activity have different significance depending on whether the activity was performed in parallel with something else or not (Chapter 6 and 7).
- Another major challenge is recognizing activity steps. Different type of challenges are observed when detecting activity steps from in situ vs wearable sensors. Since the duration of an activity step can be very short, identifying distinguishable features time series sensor data becomes harder. Also, compared to the number of types of activities, there are larger numbers of different activity steps. Although for simplification, we assume that the a particular activity step is same irrespective of the high level activity where it occurs, and is general for different users and context, in reality variation arises dues to those factors (Chapter 5).
- Activity recognition in a single person home itself is very difficult. The presence of multiple people creates additional complexities. The amount of overlapping and parallel activities increase [15] as number of people in a home increases; it makes detecting activities from raw sensors more difficult, since a sensor can be triggered by multiple concurrent activities. Again, difficulty in recognizing activities arises because different persons perform an activity in different ways [11]. Chapter 4 discusses our solution to deal with multiple people.
- A final question is when to send the notification if the system detects some activity steps of a particular activity where the later steps are missing. Now, should the system send a reminder about missing steps? What if the user is actually planing to do the steps later (interleaved activity)? Then, sending reminders often may annoy the user. However, if the user actually has forgotten the steps, how long should the system wait before sending a reminder? To solve the problem, we have prioritize notifications based on safety critical, inconvenience, and poor quality of the resultant activity due to the missing steps. If a missing step is forgetting to turn off the stove, then an alarm is generated immediately, whereas forgetting to put the sugar in the coffee is not considered as important.



## 1.2 Thesis Statement and System Overview

### 1.2.1 Statement

*"It is possible to detect activities (ADLs and IADLs) and activity types (interleaved, parallel, or sequential) and have higher performance than existing state-of-art systems by detecting the sequence of activity steps and modeling the steps in terms of time probabilistic context free grammar. Moreover, this approach enables detecting partially completed or incorrectly performed activities and is able to report the quality of the attempted or completed activity."*

More specifically, this dissertation investigates the following suppositions:

1. We assume, ADLs and IADLs are performed in sequence of activity steps. The variation of different activity instances of the same activity occur when steps are performed in different order or when choosing alternative steps or dropping the optional steps while performing the activity. We hypothesis that it is possible to capture all such variations with grammar rules and represent an activity with a single time probabilistic context free grammar.
2. By detecting sequence of activity steps, we can detect which activity was performed irrespective of the way it was performed. Moreover, we can know whether multiple activities were performed in parallel or sequentially one after other by observing the distance between consecutive activity steps.
3. Since the grammar rules define the relationship among different activity steps with in an activity, we can also detect the quality of the performed ADL and IADL instances by tracking which steps were missing, repetitive, wrongly performed, and calculating the length of individual steps and delays in between consecutive steps.
4. We also hypothesize that user identification is limited when only passive binary sensors are used despite considering user behavior and activity history data. However, adding limited number of specialized sensors can make a huge improvement in the person identification process.

### 1.2.2 System Overview

In order to prove our hypothesis, we have built several systems: SARRIMA, QuActive, Activity Step Recognizer, and Activity Quality Monitor. A conceptual integration of all the components/subsystems of these systems is shown in Figure 1.1. Since each of our systems have modular components and similar subsystem structure, this integration is possible.

The bottom layer of all our systems is the Event/Gesture Detection Module. It takes data from binary/specialized sensors and wearable data streams and uses the sensor mapping information and data processing algorithms to detect specific sensing events or gesture segments. These event are used in the upper layers for detecting activity steps (QuActive Chapter 6) or might be directly used for detecting the high level activity directly (SARRIMA Chapter 4).

The next layer of our conceptual framework is the 'Activity Step Detection and Recognition' module. Although, activity can be recognized directly from the sensor events, detecting activity steps are vital when activities can be done partially or with error. Therefore, Activity Step Recognition module recognizes all possible steps that might have occurred during a specific time frame. In this thesis, we applied standard state-of-the-art machine learning algorithms

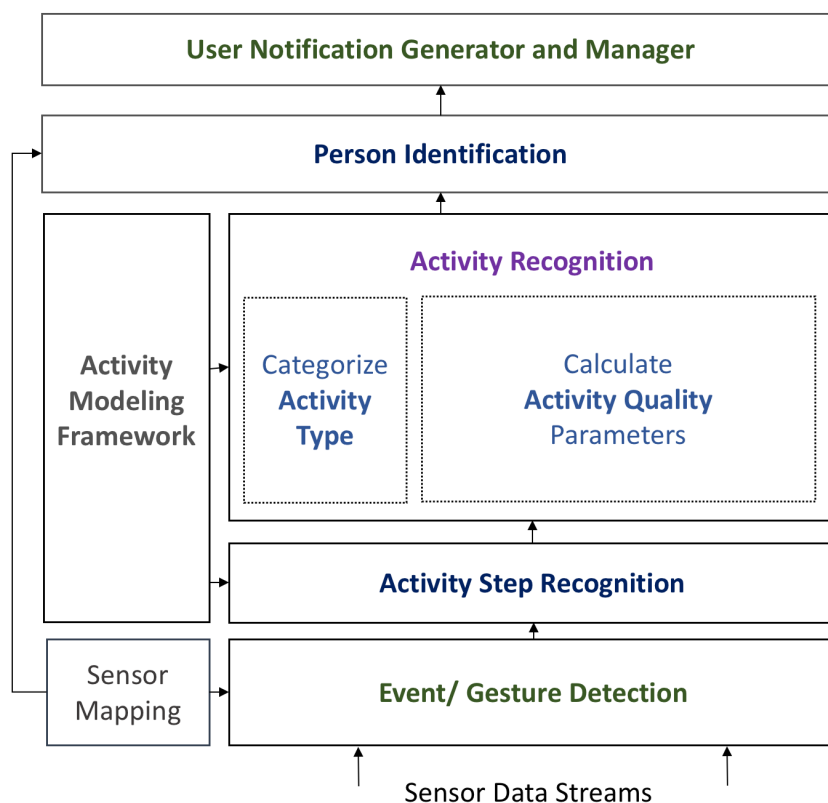


FIGURE 1.1: Conceptual integration of all the systems: SARRIMA, Activity Step Recognizer, QuActive Framework and System for monitoring the quality

for detecting the steps. However, empirical experiments were done to fine tune the parameter thresholds for better performance. Since the accuracy of overall system depends on the accuracy of detecting individual steps, the wrongly detected steps are filtered in the upper layers by matching with applicable grammar rules.

The Activity Modeling Framework (QuActive) models the different activity instances of the same activity class by using grammar rules, and captures both the structure and the variations within a specific activity. The details are provided in Chapter 6.

The core of the whole system is the Activity Recognition Module. While recognizing which activity was performed, it also finds the way a certain activity was performed as well as any discrepancies (missing/wrong step, delay, etc.) that occurred while performing the activity.

In order to provide proper feedback and also track personalized history it is important to find out what activity was performed by whom, and the person identification module assign individual to corresponding activities (SARRIMA).

The user notification system has been designed to generate reminders and notifications (Medrem [14]). Coupling the system as part of overall system makes it possible to generate notification based on specific activity quality events recognized by the underlying subsystems.

## 1.3 Contributions

The contributions of this dissertation are the following:

1. We modify a state of the art ADL recognition system AALO [16] for single person home and make it applicable to be used in a multi-person environment without sacrificing its capability to detect parallel and overlapped activities. Our system, SARRIMA, uses location and temporal information to detect ADLs and achieves average accuracy as high as 97% in all the tested public datasets with multiple people (CASAS, ARAS). SARRIMA performs better than other contemporary systems applied on the same datasets [17].
2. We present a solution to recognize the user of each activity from only passive binary sensors. However, since the accuracy of such solutions vary depending on the home, number and type of sensors, and the behavior and relationship of users - we perform a simulation to show how much improvement can be made by adding limited number of non-binary passive sensors. Very high accuracy can be achieved only by replacing or adding very few non-binary passive sensors [18].
3. This thesis presents *QuActive*, a novel activity modeling framework that utilizes fine grained information (activity steps) of the activity process. We show that the framework can capture variations of performing an activity in different ways with a single time probabilistic context free grammar (TPCFG). We show how different instances of the same activity creates different parse trees that can be generated by the same grammar definition. The concept of grammar is modified to be applicable to the activity recognition field and each of the grammar terms are described with respect of activity, activity steps, and correlation of steps within the activity (Chapter 6). The grammar also captures the sub-structure of in home activity where some steps are required to be perform in certain order whereas some can be performed in multiple places within an activity or in random order.
4. We performed a study on recognizing activity steps from in situ z-wave binary sensor and from wearable smart watches. We outlined the challenges of both approaches and provided a comparison of using each method vs the other. Finding steps from gesture events from data streams of wearable device opens the opportunity to capture lots of fine grained steps and micro behavior. But it is extremely challenging even for a single user due to the sheer number of possible steps that can occur. On the other hand, a binary sensor provides straight forward outcome of particular events, but not all type of events/activity steps can be captured by these sensors. While applying standard machine learning techniques, we realized that the problem is not trivial. Therefore, we have noted all our lessons learned and the various challenges we found in real home deployment and expect our experience will serve as important guidelines for the future researchers.
5. We implemented a system that incorporates a *QuActive* framework to recognize activity, monitor quality, and notify users. The *QuActive* framework is applied to three different public datasets of interleaved activities, parallel and co-operative activities, and dataset with activities monitoring cognitive decline. We show that even if activity steps are not detected correctly, by taking into account of all possible steps, combining inputs from both wearable and in situ sensors, we can achieve very high accuracy of activity recognition if we assume the activities were done correctly. Our evaluations on three datasets show that *QuActive* outperforms the state-of-the-art techniques for all of these datasets. The system has also been deployed in a real home in a semi-controlled setting. The results

show that *QuActive* recognizes more than 90% of the defined activity steps with a hybrid sensing platform (in situ and wearable), and the grammar detects 98.6% of the defined activities from the recognized activity steps.

6. We showed how activity quality monitoring can be achieved by defining quality parameters for ADL/IADL performed by Alzheimer's patient. We show how different episodes with error types can be separated and how the activity steps of multiple activity within the episode can be separated. We present our solution for searching and validating activity step sequences. Our evaluation shows the solution works in recognizing both complete and partial activity and is able to identify the reasons of discrepancies in case of incomplete activities.
7. We collaborated with developing a voice based medication reminder system, Med-Rem [14]. As part of this thesis, we show how to modify the medication reminder system as event base activity notification subsystem and integrate it as part of the activity recognition and quality monitoring system. The notification subsystem provides alerts about activities, informs user about missing steps, and stores user feedback.
8. As part of our experiments, we deployed both in-situ and wearable sensors and have collected data from lab and real home settings. We used video for tracking the ground truth and later annotated the datasets to label activities and activity steps within those activities. Since there has not been many datasets that have activity steps information, these annotated datasets are extremely valuable and can be used later for further exploration by the research community.

## 1.4 Thesis Outline

The rest of the dissertation is organized as follows:

- Chapter 2 introduces the state of the art research related to activity detection and recognition, research on activity step detection and recognition, research on grammar, ontology, and rule based systems, and research on home monitoring and notification systems.
- Chapter 3 describes different terminologies related to activity definition and activity taxonomy based on different context, and details about activity steps and related terms used for describing portions of activities. It provides insights on the terminologies on context free grammars, probabilities, and rules. Finally, a brief description is given on the data structures and algorithms used in our system on top of which we extended our research.
- Chapter 4 presents the details of different components of SARRIMA, a novel activity recognition system in multiperson homes, and its evaluation.
- Chapter 5 details the challenges on recognizing activity steps and provides a comparison and analysis of using in-situ vs wearable sensors for recognition of activity steps.
- Chapter 6 describes QuActive Framework for modeling activities and its performance in different datasets and collected data from a home setting.
- Chapter 7 provides the details on how activity steps are tracked from activity process, specially missing steps, repetitive steps, wrong steps, and wrong order of steps with in an activity and its evaluation results.

- 
- Chapter 8 describes the notification system and its performance. It also provides additional details about how the system works as part of the overall activity recognition and quality monitoring system, and necessary parameters to provide event based notification.
  - Chapter 9 concludes this dissertation with a summary of the contributions and provides a number of possible directions for future work.



## Chapter 2

# Related Work

In this chapter, we describe the ‘state-of-the-art’ systems and research techniques most relevant to our thesis. While reviewing the literature, we wanted to provide information about both the latest or current technology, as well as the research works that has inspired and used a lot in later works and pushed the frontier. The related work is divided in several sections. First, we discuss the available home health care systems (2.1) and research related to activity detection and recognition (2.2). Then, we describe the works that looked in to the details of the activity process (2.3). Finally, we present research related to notification systems (2.4) and quality monitoring systems (2.5). In each of the sections, we identify the unique aspects of our work, in comparison with the described state-of-art research.

### 2.1 Home Monitoring Systems

There are smart home systems for security, energy management, and other specific applications that are summarized well in a survey paper [58]. There are large scale health monitoring systems, such as CodeBlue that are not used in home but in hospital or care facilities. In this section, we are only interested in in-home systems for health and behavior monitoring.

#### 2.1.1 Research Testbeds

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]. 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]. Empath is an extensible, multimodal, largely passive behavioral monitoring system that is useful to caregivers in order to monitor their patient’s behavior, and thereby track their well-being and their response to treatment and therapies. The system not only served as a testbed, but worked well in real home deployments with real patients. However, empath mainly tracks two aspects of behavior - one is a sleep tracking module and another is mood tracking from voice data. Therefore, it is fundamentally different than our quality monitoring system that tracks activities of daily livings.

### 2.1.2 Commercial Products

Various 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. 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 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.

## 2.2 Research in Activity Detection and Recognition

### 2.2.1 Different Sensing Platforms

#### Wearable Sensing

A lot of work on activity recognition are done using wearable sensors or devices that can be easily carried. Examples include smart phone, smart watch, Fitbit, Ubitfit, and RFID devices [19]–[21]. A major advantage of this approach lies in multiperson scenerio where a person can be identification based on unique device ID. However, most of the activities recognized by this process are mainly physical activities or concentrates on the detection of a particular ADL. For example, Google activity recognition APP in mobile phone provides limited information about the user's activity, such as whether the user is on foot, in a car, on a bicycle, or still. There are apps that track walking, running, or sleeping. Fitbit and Ubitfit are used to track exercise. A wearable necklace is used to detect eating [22]. Therefore, wearble sensors are useful to detect particular type of activities, but have limitation in scenarios where a number of variety ADLs are needed to be recognized. Moreover, they are not comfortable; users often forget to wear them, and scaling the system to multiple persons is energy and cost consuming. Nonetheless, the popularity of wearables are rising and many claims that the benefits out-weights the disadvantages, specially as wearbles are becoming powerful in terms of computational capability and more energy efficient everyday.

#### Infrastructure based Sensing

Infrastructure based sensors, such as ElectroSense or HydroSense [23] are useful to detect particular type of activities that consume the same resource. The advantages of these sensors are that they are single-point, which reduce both the installation cost and overall system cost. However, these approaches require significant configuration and training effort from the end user, high cost sensors, and have not yet been evaluated in in-situ home environments. Most



importantly, these sensors cannot be used to recognize all the different classes of ADLs and IADLs, since the same resource is not used in all the activities.

### **Image based Recognition**

A lot of work in Activity Recognition is based on image processing and analyzing data from cameras [24]. However, in this approach the main problems are limited coverage area, obstacles, complexity of recognition due to user's angular variation while performing activity, and higher cost. requirement of a huge amount of data processing, Moreover, the system is costly in terms of data processing and required memory. The biggest concern is the violation of privacy - since most ADLs are private.

### **Wireless Sensor Network**

A lot of activity recognition systems or studies use wireless sensor installed in a surrounding environment. This approach is preferred by users since activities are monitored passively by detecting human-to-object interaction and not tracking the human directly. However, diverse physical activities that do not involve interaction with everyday objects can not be recognized. Moreover, there is a limitation on the number of objects that can be instrumented for detection.

## **2.2.2 Different Algorithms and Techniques**

Irrespective of what type of sensing platform is used, a significant proportion of research on activity recognition has focused on the learning techniques/detection algorithms used to infer resident activities from the sensor data.

### **Machine Learning Methods**

The existing research of recognizing activities of daily living with ubiquitous sensors uses different statistical and probabilistic approaches [25], [26]. The common supervised machine learning algorithms used in ADL recognition are Hidden Markov Model (HMM), semi HMM, Naive Bayes Classifier (NBC), and conditional random field (CRF) [13], [27]. Since NBC do not retain any timing information, some researchers use Dynamic NBC for activity detection. Again, the problem with HMM is when the sub-activities of a complex activity are also complex activities (such as cooking) the performance decreases, since in that case the hidden layers are not directly observable [28]. Both HMM and CRF are focused on the sensor sequence and are less flexible in incorporating variability of activities. The major drawback of a supervised method is that tremendous effort in data labeling is required during the training period, which is not always feasible for patients or elderly people, and specially for research with short term pilot studies. On the other hand, algorithms like item set mining disregard the sequence information of sensors and number of occurrences of a particular sensor which are useful information necessary for defining the detailed steps. Thus, although the algorithms are capable of detecting and reporting activities, no information is provided about the activity process. Moreover, for certain closely related activities (brushing teeth and shaving) the systems show poor performance in activity detection.

## Grammar, Ontology, and Rule based Activity Recognition Systems

The use of Context Free Grammar (CFG) in defining and recognizing human activities is not new. Previously, Li [29] presented a grammar-based Fall Detection framework that can recognize slow falls and better differentiate falls from other fall like activity. In paper [30] the authors use a CFG to recognize human action from video footage by co-relating sequence of human pose. Again, papers [31], [32] demonstrate how cooking activity can be better recognized from imote camera by associating it (using CFG) with accessing food, serving food, and cleaning activities. In their later works, the authors showed [8] how a Probabilistic CFG can be used to define and find the relationships among all the activities of daily living from video data. Probabilistic CFG has also been used in surveillance system [33] in order to determine unusual activities. Paper [34] uses a stochastic CFG to recognize a multitask activity (playing blackjack) from video data. However, none of the work addresses the issues of recognizing a variety of activities and monitoring activity process quality from a general framework.

## 2.3 Research in Detecting and Recognizing Activity Steps

The research work that focuses on garnering fine-grained information about activities are mostly done is vision and using image processing. The papers [35], [36] describe a dataset of fine grained cooking activities from video image processing. The main motivation of the works is to create necessary image processing capability for an assistive cooking robot to detect cooking steps. Paper [37] describes a smart kitchen where each appliance can wirelessly communicate with each other and partially automate the later steps of an activity if the initial steps are detected. However, the steps are static defined steps and no framework is defined to relate the different steps of an activity. Paper [7] describes fine grained ADL detection from RFID tags and defines models to associate different objects with different activities. However, the focus of the paper is to relate objects with activities; whereas our work concentrates on relating different steps of an activity and extracting more information about each step.

## 2.4 Notification and Reminder Systems

To avoid missing important tasks, people use different kinds of reminders, ranging from traditional methods like notes to technology enabled systems like text messages and smart phone apps. With the ubiquity of cell phones, particularly with the recent proliferation of smart phone usage, the use of these devices for medication alerts and tracking has received significant attention from different stakeholders including patients, caregivers, developers and researchers. Text messages are used for health intervention in several studies [38][39][40]. The text message based systems are not convenient for user interactions, and so these systems are inflexible in re-scheduling reminders, and tracking medication. Most of the limitations of the smart phone based systems, as mentioned earlier, are also applicable for the text message based systems.

A number of smart phone applications with different features are available in the app stores for providing medication reminders and tracking intakes [41] [42]. A functionality review of 229 of the apps, as reported in [43], shows that many of the apps lack important features like re-scheduling, medication pictures, and data export. For example, only 17% of the apps offered an option to re-schedule or postpone a reminder. Researchers have also designed, developed and evaluated smart phone based reminder and tracking systems. For example, Wedjat [44] is a smart phone based system that provides reminders and tracks medication intakes. It also

provides potential drug-drug/drug-food interaction information to the users. UbiMed [45] presents a solution that incorporates smart phone apps to provide reminders, and to support the tracking of prescribed medication for the aging and disabled population. As described earlier, smart phone based systems for medication reminder and tracking come with a number of limitations. A feasibility study [46] reveals some of the limitations.

Smart wearable devices like smart watches and wrist bands are usually enriched with many features like touch screens, microphones, sensors, Bluetooth and Wi/Fi. These devices are being used widely in healthcare applications including activity tracking, wellbeing monitoring, and reminders. Harmony [47] is a hand wash monitoring and reminder system that uses the sensors available in the smart watches to monitor the hand wash activities of the users. Whenever a user enters into or leaves areas like patient rooms and toilets, where hands should be washed before entering and/or after leaving, the watch detects the location from the signals of the Bluetooth beacons placed in the areas. If a user forgets to wash hands when it is required, the watch provides a reminder. A diary like system for diabetes patients is presented in [48] that uses both smartphones and smartwatches to log information from, and provide reminders to diabetes patients. SPARK[49] is a framework that combines smartphones and smartwatches together in monitoring symptoms of patients with Parkinson Disease. It also supports physicians in providing tele-interventions to the patients. Fabian et al [50] proposes to use pictures of the drugs on the display of the wrist device to reduce confusion of the patients when multiple drugs need to be taken. These systems use only the small display of the wrist device, and so can not provide detailed information related to a reminder using the wrist device only. Also the wrist devices used in the systems do not support rescheduling the reminders. Most of these reminder and tracking systems are aimed for specific group of patients or users. In contrast, MedRem is a general purpose medication reminder and tracking system that can be customized according to the patients' needs. It incorporates speech recognition and text-to-speech technologies along with intelligent interface design to overcome the limitations of the small display of the wrist devices as well as to provide useful features like reminder rescheduling and medication tracking.

## 2.5 Research in Activity Quality Monitoring

A lot of research claims detecting activities of daily livings for monitoring quality of life. However, they do not measure the quality directly. Several clinical studies access quality of life of particular patients by questionnaires or survey about their daily life [51].

Most of the research in Activity Quality Monitoring are tailored to particular activity. For example, paper [52] examines sleep quality and correlation of sleep quality with incontinence events of Alzheimer patient's. There are several works that study sleeping behavior of patients with sleep apnea [53]. A person at the primary phase of Parkinson's tends to make small and shuffled steps, and may also experience difficulties in performing key walking events, such as starting, stopping, and turning [54]. Therefore, there have been works focused on walking behavior and gait monitoring [55]. Paper [56] provides a comprehensive study on remote patient monitoring where the patients always have to wear sensors that constantly tracks their vital signs. However, those studies are helpful only for patients with severe injury. Moreover, they are not suitable for long term monitoring of daily life.

Although, we did not find any system that specifically monitors activity quality by considering fine grained information, we found some closely related work on behavior monitoring

and anomaly detection. The main idea in these research works is to train the system with activity data for certain amount of time (several weeks) and consider those pattern as normal behavior. Later, the system tracks activity and any deviation from those pattern is reported as an anomaly. Now, different researchers consider different aspect of data as behavior. Anderson et al. [57] 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. Jakkula et al. [58] use temporal mining to learn different temporal relations among different activities. Holmes [59] considers daily routine, activity duration, and change of routine in week days vs weekends. However, the activity anomaly research mainly focuses on the relationship of the activities within a day and in which sequence they occur and report anomaly from a broader perspective, but do not look in to the details of activity steps for determining the quality of a particular activity.

Review paper [60] looked into one hundred seventy-five unique studies that monitored the ADLs of elderly people and preferably measured some clinical outcomes such as ability to predict key events that require intervention. They found most studies reported on technical improvements in methods for detecting changes in ADL, few, if any, determined the benefits to the patient of telemonitoring for changes in ADL or correlation with any physiological changes. Therefore, we believe our work on activity quality monitoring will be valuable from clinical perspective and encourage more researchers in this important direction.

## Chapter 3

# Background and Terminology

### 3.1 Activity Recognition

The area of activity recognition is quite diverse and broad. There is a tremendous diversity of concepts that are classified as activities in the literature (section 2.2). Some define activity as 'performing/doing' something, whereas some define it as the 'result/state' of something being done. Therefore, activity recognition is performed from a user's perspective as well as from user to object interaction and detecting the changed state of objects.

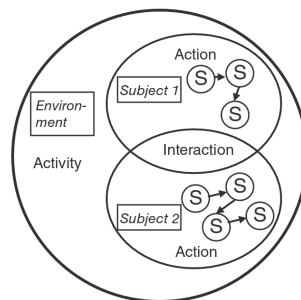


FIGURE 3.1: Action, environment, and activity

#### 3.1.1 Physical Activity vs. Daily Living Activities

Gesture and physical activity recognition are often labeled as activity recognition. However, there is some core difference between these activities vs. activities of daily living.

##### Physical Activities and Gestures

Physical activity is defined as any bodily movement produced by skeletal muscles that require energy expenditure. For example, walking, running, cycling, exercising, etc. On the other hand, a 'gesture' is defined as a movement of part of the body, especially a hand or the head, to express an idea or meaning. Often these types of activities are referred to as 'actions'. A lot of these actions are used while performing activities of daily living. For example, some activities are opening a door, opening a cabinet, picking up an item, push an item, pull an item, carry an item, throw an item, etc. Again, when the action involves more than one person, it is referred to as 'interaction'. For example, shaking hands, talking to someone, handing item to someone, etc. Figure 3.2 shows how action and interactions result in an activity.

### Activities Daily Living and Instrumental Activities Daily Living

Activities of daily living (ADLs or ADL) is a term used in healthcare to refer to people's daily self-care activities. Health professionals often use a person's ability or inability to perform ADLs as a measurement of their functional status, particularly concerning people post-injury, with disabilities and the elderly. On the other hand, instrumental activities of daily living (IADLs) are not necessary for fundamental functioning, but they let an individual live independently in a community. The table 3.1 lists some ADLs and IADLs from everyday life.

TABLE 3.1: Example of Activities of Daily Livings and Instrumental Activities of Daily Livings

Basic ADLs	Instrumental ADLs
Sleeping	Work at computer, work at desk
Eating	Preparing Meals
Toileting	Washing Dishes
Showering	Laundry
Dressing	Watching TV
Brushing	Cleaning the house
Moving	Study
	Write letters, cards
	Make phone call, talk on phone
	Drive car, bus or ride in car, bus
	Play board game, play card game

### 3.1.2 Activity Types: Sequential, Interleaved, and Parallel

#### Sequential Activities

Activities that are performed one after another are called sequential activities. Each individual activity is completed before starting the next one; therefore, no two activities occur together. For example, in the morning if a person gets out of bed, go to the washroom, brushes teeth, gets dressed, prepares food, eats food, and leaves the house, then the person has performed all these activities sequentially.

#### Interleaved Activities

Certain real-life activities may be interleaved. For instance, while cooking, if there is a call from a friend, people pause cooking for a while, and after talking to their friend, they come back to the kitchen and continue to cook.

#### Parallel Activities

When multiple activities occur at the same time, those activities are referred to as parallel activities. Users usually switch back and forth among multiple activities. For example, if someone watches TV while eating, or washes dishes while cooking food.

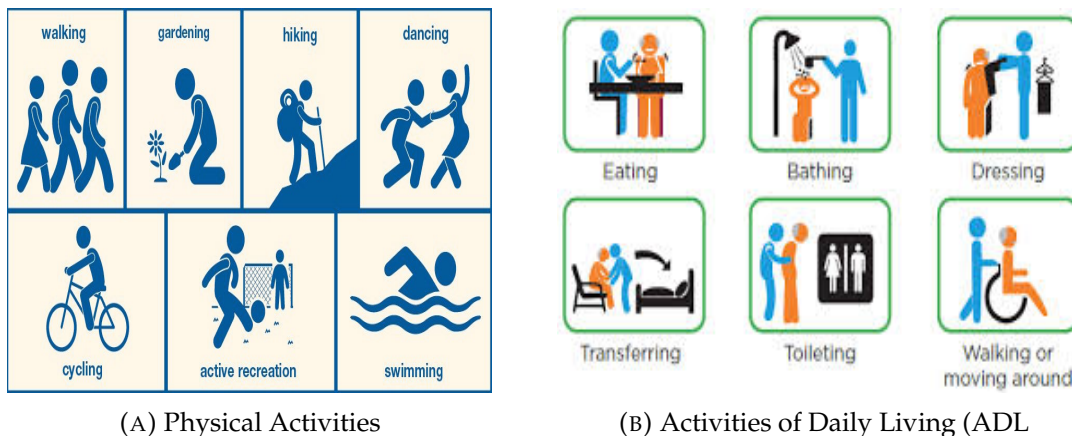


FIGURE 3.2

### 3.1.3 Activity Steps

The concept of activity steps comes from the desire to look into the details of an activity process. First, we provide similar definitions from the literature and then provide the definition and properties of an activity step that we consider for this thesis.

#### Atomic Activity

Atomic activity: is a body part movement sequence where the movement is limited to the finite temporal window and variation of movement limited to a small space. Example: a cycle of walking i. Four activities: walking, marching, line-walking, kicking while walking ii. Single letter utterance: 13 letters

#### Sub Activity and complex activity

A sub-activity is an activity making up part of a larger activity. When a large activity is made up of parts, where each part can also be identified as an activity is called a complex activity. For example, cooking a meal is a complex activity where sub activities might include cleaning ingredients, chopping ingredients, heating water, mixing ingredients, using the stove, etc.

#### Micro-activity

In paper a micro-activity ( $\mu Ac$ ) or an activity step is defined as the smallest activity step that cannot be decomposed any further. Therefore, a  $\mu Ac$  is equivalent to an atomic activity or a simple activity defined in the state-of-the-art literature, but in a broader sense which includes both objects and gestures. The following statements hold true for a  $\mu Ac$ :

- i An activity can be broken into one or more  $\mu Ac$ s. So, a  $\mu Ac$  can be an activity itself. For example, 'heating water' can itself be an activity or a  $\mu Ac$  of 'making tea'.
- ii  $\mu Ac$ s cannot be done partially, i.e., once started a  $\mu Ac$  has to be finished, or otherwise it is disregarded.
- iii  $\mu Ac$ s can occur in different activities. For example, the  $\mu Ac$  'using water' can be a part of the activity 'washing dishes' or the activity 'mopping the floor'.

- iv Although every activity is associated with one or more users, and every  $\mu Ac$  is associated with some activity, the  $\mu Ac$  itself might be independent of a user. For example, a user triggers the switch to boil water, but ‘water boiling’ itself is independent, and the user may do something else during that time.

In this thesis, we consider the definition of  $\mu Ac$  as an activity step. Besides, we also consider whether the step can be detected from the sensing perspective or not.

## 3.2 Grammar and Rules

### 3.2.1 AI: Rule-based Systems

In computer science, rule-based systems are used as a way to store and manipulate knowledge to interpret information in a useful way. They are often used in artificial intelligence applications and research.

A typical rule-based system has four basic components:

- A list of rules or rule base, which is a specific type of knowledge base.
- An inference engine or semantic reasoner, which infers information or takes action based on the interaction of input and the rule base. The interpreter executes a production system program by performing the following match-resolve-act cycle.
- Match: In this first phase, the left-hand sides of all productions are matched against the contents of working memory. As a result, a conflict set is obtained, which consists of instantiations of all satisfied productions. An instantiation of productions is an ordered list of working memory elements that satisfy the left-hand side of the production.
- Conflict-Resolution: In the second phase, one of the production instantiations in the conflict set is chosen for execution. If no productions are satisfied, the interpreter halts.
- Act: In the third phase, the actions of the production selected in the conflict-resolution phase are executed. These actions may change the contents of working memory. At the end of this phase, execution returns to the first phase.

Temporary working memory. A user interface or other connection to the outside world through which input and output signals are received and sent.

In AI, the most common method for defining rules are *Propositional Logic* and *First Order Logic*. Propositional Logic deals with propositions (which can be true or false) and argument flow. Compound propositions are formed by connecting propositions by logical connectives. The propositions without logical connectives are called atomic propositions. The main difference between propositional and first-order logic is that first-order logic uses quantified variables over non-logical objects and allows the use of sentences that contain variables so that rather than propositions it uses sentences with variables and quantifiers. Some researchers have used this logical reasoning for activity recognition, where they described variables in terms of sensors and used proposition and connectivity's to come to a conclusion about which activity was performed.



### 3.2.2 Formal Grammar

In formal language theory, a grammar is a set of production rules for strings in a formal language. The rules describe how to form strings from the language's alphabet that are valid according to the language's syntax. A grammar does not describe the meaning of the strings or what can be done with them in whatever context—only their form.

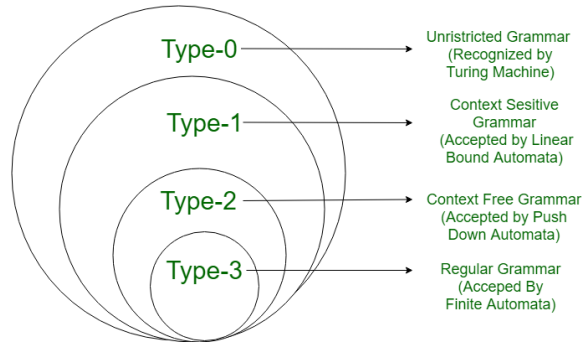


FIGURE 3.3: Four types of grammar defined by Noam Chomsky

#### Syntax of Grammars

A grammar is a type of language generator. It is expressed as  $\langle V_N; V_T; Start; R \rangle$ , where

- $V_N$  is a finite set of nonterminal symbols. Nonterminals are represented with words starting with a capital letter.
- $V_T$  is a finite set of terminal symbols. Terminals are represented with words starting with lower-case letters.
- $V_N \cap V_T = \emptyset$ .  $V = V_N \cup V_T$  is called the vocabulary and  $V^*$  is the set of all strings of symbols in  $V$  including the string of length zero.
- $Start \in V_N$  is the start symbol.
- $R$  is a finite nonempty subset of  $V_N \times V^*$  called the production rules. Each production rule is in the form of  $\alpha A \gamma \rightarrow \beta B \theta$

Depending on which symbols are allowed in each side of the production rules, Noam Chomsky described the hierarchy of formal languages with four types of grammars (Figure 3.3). For example, in context-free grammar (Type-2), only a single nonterminal can be in the left side of the production rule, whereas the right side of the rule can have both terminal and nonterminal symbols.

### 3.3 Algorithms

In this dissertation, we have worked on top of the algorithms that have been used in other fields. Although sometimes we modified a variation of the algorithm based on state-of-the-art papers, here we will describe only the basic ideas and classical versions of the methods.

### 3.3.1 Frequent Itemset and Association Rule Mining

Association Rule Learning (ARL) is a rule-based machine learning method for discovering interesting relations between variables in large databases. It searches for frequent items such that the set of items appear together frequently in a transaction or relation. This technique is often used in transactional and relational database applications. In contrast, with sequence mining, association rule learning typically does not consider the order of items either within a transaction or across transactions.

The idea that same activities usually trigger the same set of sensors in a home instrumented with in-situ sensors, several researchers have been interested in applying the concept of rule mining in ADL and IADL recognition [16]. We explored this idea in chapter 4. Here, we list some of the essential definitions for Association Rule Mining:

**Support:** It is a measure of how frequently the itemset appears in the dataset. Support tells about usefulness and certainty of rules. The support of  $X$  with respect to  $T$  is defined as the proportion of transactions  $t$  in the dataset which contains the itemset  $X$ . For example, 5% *Support* of some rule means total 5% of transactions in the database follow the rule.

$$Support(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}$$

**Support\_count(X) :** Number of transactions in which  $X$  appears. If  $X$  is  $A$  union  $B$ , then it is the number of transactions in which  $A$  and  $B$  both are present

**Confidence:** Confidence is an indication of how often the rule has been found to be true. For example, A confidence of 60% means that 60% of the customers who purchased milk and bread also bought butter.

$$Confidence(A \rightarrow B) = Support\_count(A \cup B) / Support\_count(A)$$

If a rule satisfies both minimum support and minimum confidence, it is a strong rule.

**Maximal Itemset:** An itemset is maximal frequent if none of its supersets are frequent.

**Closed Itemset:** An itemset is closed if none of its immediate supersets have same support count same as Itemset.

**K- Itemset:** Itemset which contains  $K$  items is a  $K$ -itemset. So it can be said that an itemset is frequent if the corresponding support count is greater than the minimum support count.

#### *Apriori* Algorithm

*Apriori* is a frequent itemset mining algorithm for association rule learning. It proceeds by identifying the frequent individual items in the database and extending them to larger and larger item sets as long as those itemsets appear sufficiently often in the database. Figure 3.4 shows the pseudo code of *Apriori* algorithm.

***Apriori* Property:** All subsets of a frequent itemset must be frequent. If an itemset is infrequent, all its supersets will be infrequent.

*Apriori* uses a breadth-first search strategy to count the support of itemsets and uses a candidate generation function which exploits the downward closure property of support.

### The Apriori Algorithm

- **Join Step:**  $C_k$  is generated by joining  $L_{k-1}$  with itself
- **Prune Step:** Any  $(k-1)$ -itemset that is not frequent cannot be a subset of a frequent  $k$ -itemset
- **Pseudo-code:**

```

 $C_k$ : Candidate itemset of size k
 $L_k$ : frequent itemset of size k
 $L_1 = \{\text{frequent items}\}$ ;
for ( $k = 1$ ;  $L_k \neq \emptyset$ ;  $k++$ ) do begin
   $C_{k+1}$  = candidates generated from  $L_k$ ;
  for each transaction  $t$  in database do
    increment the count of all candidates in  $C_{k+1}$ 
    that are contained in  $t$ 
   $L_{k+1}$  = candidates in  $C_{k+1}$  with min_support
end
return  $\cup_k L_k$ 

```

FIGURE 3.4: Apriori itemset mining and rule learning algorithm

### 3.3.2 Pattern matching with Regular Expression

In computer science, pattern matching is the act of checking a given sequence of tokens for the presence of the constituents of some pattern. Sequence patterns (e.g., a text string) are often described using regular expressions and matched using techniques such as backtracking. Sequence patterns are very popular in text editors, and other applications (DNA matching, genetic coding analysis, etc.) of these pattern searching are mostly a variation of string-searching algorithms.

**Finite Set of Pattern** There are many applications where a pattern can be represented finitely. For example, string searching, plagiarism checking, etc. The best finite pattern searching algorithms are -

1. Aho–Corasick string matching algorithm (extension of Knuth-Morris-Pratt)
2. Commentz-Walter algorithm (extension of Boyer-Moore)
3. Set-BOM (extension of Backward Oracle Matching)
4. Rabin–Karp string search algorithm

Most of these algorithms are made faster by using hash or pre-computation to skip ahead. Therefore, in the best cases, the algorithms complete within a linear time limit. The difference between these algorithms lies in how they pre-process the patterns.

**Regular Expressions:** The patterns that can not be enumerated finitely are represented usually by a regular grammar or regular expression. Regular expressions use meta-characters or quantifiers to repeat certain symbols and therefore are able to produce infinite number of sequence. Figure 3.5 shows a pattern matching algorithm which may contain the meta character '\*' kleen star. If a literal character is followed by '\*', that character can be repeated zero or more times. The algorithm shown in figure 3.5 is a variation of the dynamic algorithm version of String Edit Distance algorithm.

### 3.3.3 Machine Learning Algorithms

We compared our approach with existing literature which uses the same dataset for evaluation but uses a different approach. The baseline papers used different supervised machine learning

```

public boolean isMatch(String text, String pattern) {
    boolean[][] dp = new boolean[text.length() + 1][pattern.length() + 1];
    dp[text.length()][pattern.length()] = true;

    for (int i = text.length(); i >= 0; i--){
        for (int j = pattern.length() - 1; j >= 0; j--){
            boolean first_match = (i < text.length() &&
                (pattern.charAt(j) == text.charAt(i) ||
                 pattern.charAt(j) == '.'));
            if (j + 1 < pattern.length() && pattern.charAt(j+1) == '*'){
                dp[i][j] = dp[i][j+2] || first_match && dp[i+1][j];
            } else {
                dp[i][j] = first_match && dp[i+1][j+1];
            }
        }
    }
    return dp[0][0];
}

```

FIGURE 3.5: Regular expression matching with kleen star

models, such as the Hidden Markov Model and Naive Bayes. While experimenting with activity step recognizers we tried different popular machine learning methods used for gesture recognition from wearable devices in the literature. We summarize some of the methods here -

**Hidden Markov Model** The HMM is a popular tool for modeling data that can be characterized by an underlying process generating a sequence of observations, such as sensor events. HMMs are generative probabilistic models consisting of a hidden variable and an observable variable at each time step. In the context of modeling activities, the hidden variable is the activity and the observable variables are the sensor events or the features extracted from the sensor events. The HMM facilitates computation of the probability that a particular activity model produced the entire sensor sequence, as well as the state (activity) sequence that was most likely to have produced the observations (sensor events).

**Naive Bayes Classifier:** This is a probabilistic classifier, which is simple to develop and can be executed rapidly. However, it is based on the weak assumption of feature independence. The Naive Bayes classifier assumes that the acceleration data from the sensor has a Gaussian distribution whose mean and variance depend on class set. In the training stage, all the related mean and variance in different hypotheses are calculated and the Gaussian model for activities is built. During the testing stage, given a segment of raw data, the activity with the maximum probability based on the equation below is identified. In the following equation,  $H_1, H_2, \dots, H_n$  represents the features extracted from accelerometers, and  $C$  is the objective activity. Although it might work well for recognizing some activity classes, the assumption of feature independence usually results in reducing the overall recognition accuracy.

$$\operatorname{argmax}_c(P(C = c|H_1, H_2, H_3, \dots, H_n)) = \operatorname{argmax}_c \frac{1}{Z} P(C) \sum_{i=1}^n P(H_i|C)$$

**Decision Tree** This is a decision support tool using a tree-like model, which is used to describe decisions, their outcomes, and costs. This algorithm works by examining the discriminatory ability of features to create a set of rules which ultimately leads to a complete classification system. In the training stage, the construction of the decision tree is usually based on the feature

selection algorithm. In the testing stage, a tree traversal algorithm is used for classification. In [12], the Decision Tree classifier was used to recognize 20 activities with the time and frequency features and obtained the recognition accuracy of 84% which was the highest among the evaluated classifiers.

**Random Forest** Random forest uses multiples decision trees with different feature emphasis and various parameters to come up with an aggregated decision.

**K-Nearest Neighbors** It can be used to classify a large number of different activities. However, the on-line execution is slower than the Decision Trees classifier due to the distance evaluation requirements. The k-nearest neighbor classifier has been used for activity recognition in several studies

## 3.4 Hardware and Software

### 3.4.1 Sensing Modalities

Activities are recognized using different sensing modalities and some of the variations are already described in related works. In this section, we describe the devices we used as part of our data collection or the sensors found in the datasets we used.

#### In situ sensors

**Binary:** As part of our data collection procedure, we used z-wave door sensors and motion sensors developed by Aeon Labs. We also used custom made z-wave binary pressure pads that use commercial door sensors available in the market and converts the pressure to a binary value. All of these sensors are binary sensors. Therefore, they are simple and easy to install as well as consuming less power. Each of the sensors lasts almost a year using two AAA batteries. The door sensors were installed on home doors, the refrigerator door, microwave door, cabinets, dressers, water taps, stove knobs, and so on. The pressure pads were put on chairs and sofas. The motion sensors were placed at different places of each room and the hallways. The values triggered by the sensor was collected by a z-wave modem (Aeon Labs DSA02203-ZWUS Z-Wave Z-Stick Series 2 USB Dongle) developed by Aeon Labs. We connected up to 30 devices with a single modem. The modem is connected to a usb port of a laptop and our program reads the data from the usb port.

Although we did not use any non-binary in situ sensors during our data collection process, the public datasets we used for evaluation had some additional binary and non-binary sensors. For example, an electricity usage sensor provides the percentage of electric usage periodically.

#### Wearable

We used the Samsung gear s smartwatch as a wearable device. The smartwatch has an inertial measurement unit (IMU) with sensors integrated within the board. We used a sampling rate of 50Hz and collected time series data from the accelerometer, gyroscope, and magnetometer. The watch data was sent to a laptop via WiFi.

### 3.4.2 Software Environments

In our thesis, we have used the following software tools for building the system components and evaluating the data at different stages of development.



FIGURE 3.6: The first row shows Aeotec z-wave door sensor, motion sensor, and modem respectively by Aeon Labs. The next row shows Samsung Gear S smart-watch and an inertial measurement unit (IMU) with accelerometer, gyroscope, and magnetometer within the watch.

### 3.4.3 Netbeans IDE with Java 1.8

NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. We used Netbeans IDE that supports JDK 8 features, such as lambda expressions, repeatable annotations, compact profiles, etc. However, none of the exclusive features from Java 1.8 were used; therefore our program will also run on any IDE supporting Java 7 or 8.

### 3.4.4 Matlab

MATLAB (matrix laboratory) is a multi-paradigm numerical computing environment and proprietary programming language developed by MathWorks. MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, the creation of user interfaces, and interfacing with programs written in other languages, including C, C++, C, Java, Fortran, and Python.

### **3.4.5 Weka**

We used the Waikato Environment for Knowledge Analysis (Weka) for applying machine learning algorithms. Weka is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It is free software licensed under the GNU General Public License. Weka supports several standard data mining tasks, more specifically, data pre-processing, clustering, classification, regression, visualization, and feature selection. Weka API can also be used within a Java program by importing the package.





## Chapter 4

# SARRIMA: Smart ADL Recognizer and Resident Identifier in Multi-resident Accommodations

The ability of performing in-home activities successfully is used as an important factor in deciding treatments and services for patients and elderly citizens. However, most of the in-home activity monitoring systems are designed for single-resident house. The presence of multiple people creates higher numbers of parallel and overlapping activities, and introduces additional complexities in defining and recognizing activity instances. In this chapter, we present the design, implementation and evaluation of *SARRIMA*, a system that recognizes activity instances and assigns those activities to a person in two-resident homes using only passive sensors. We evaluate the efficiency of *SARRIMA* in two different public datasets (data from real homes) with multiple residents. We also show how the person assignment accuracy varies as a function of the similarity of behavior of the two people living together and of the types of passive sensors installed.

The rest of the chapter is organized as follows. We briefly describe the motivation of the work in section 4.1, followed by the contribution (chapter 4.2). Section 4.3 provides a high-level system description of *SARRIMA* and our approach for dealing with multiple people. Finally, the evaluation and discussion is presented in section 4.4 and section 4.5 concludes the paper.

### 4.1 Motivation

Systems for measuring Activities of Daily Livings (ADL) and Instrumental Activities of Daily Livings (IADL) play a significant role in home health-care. ADL refer to the daily self-care activities performed by an individual; examples include eating, sleeping, showering, dressing, toileting, and transferring. On the other hand, IADL refer to a more complex set of activities that are not fundamental, but very important for independent living - such as preparing dinner, cleaning the house, talking on the phone, and managing finance. The ability to perform ADLs and IADLs successfully is considered as an important criterion to access the condition of stroke patients and patients suffering from depression, Alzheimer, orthopedic, neurological or sensory deficits [3]–[5]. Besides, in the case of older citizens, these criteria are significant predictors of admission to a nursing home, use of paid home care, use of hospital services, use of physician services, insurance coverage, and mortality [6]. According to a 2013 government profile of older Americans [2], on average 92% of the institutionalized and 40% of the

non-institutionalized older citizens have difficulty in performing one or more ADLs, and this percentage becomes higher as they grow older. Therefore, detecting and recognizing ADLs are essential for detecting early symptoms of a disease, the improvement of access to prescribed medication, providing the exact medical history to physicians, and as a crucial preliminary step in systems for assistance in performing ADLs.

ADL detection systems are commonly designed for single-user residences. Although 28% of older American citizens live in single-resident homes, most of the older citizens (57%) live with their spouse [2]. Therefore, most homes require ADL systems that can perform accurately in the presence of more than one individual. However, the presence of multiple people creates additional complexities. The amount of overlapping and parallel activities increase [15] as the number of people in a home increases; it makes detecting activities from raw sensors more difficult since a sensor can be triggered by multiple activities. Again, difficulty in recognizing activities arises because different persons perform an activity in different ways [11]. The cost of scaling an existing ADL recognition system might grow exponentially if each individual has to be dealt with separately. Identifying people without using privacy an invasive device is also extremely challenging.

One way of tackling the challenges of multiple people scenario is to use RFID technology and wearable sensors for activity detection and consider each person separately [61], [62]. However, the expectation of elderly people or patients carrying additional devices while performing all the activities is often unreasonable. Moreover, this approach makes the user uncomfortable and does not work if the user forgets to wear/use the device. This approach cannot detect visitors and requires additional equipment each time a new user enters the system. Another way of handling multiple users is by using a camera [24] for both ADL recognition and user identification. However, in this approach the main problems are limited coverage area, obstacles, the complexity of recognition due to user's angular variation while performing an activity, higher cost, the requirement of a massive amount of data processing, and the violation of privacy - since most ADLs are private. Besides, the works in both vision and wearable sensors are mainly focused on physical activities or gestures from which the ADLs are inferred. Thus, activities that have similar physical movements require more nuanced analysis and more computational resources.

## 4.2 Contributions

In this thesis chapter, we present *SARRIMA*, a system that recognizes ADLs from passive wireless sensors installed in multi-resident homes. The system is evaluated in multiple existing data-sets [13], [27] of ADLs performed by multiple residents in real homes. The main contribution can be summarized as -

- We modify a state of the art ADL recognition system AALO [16] for a single person home and make it applicable to be used in a multi-person environment without sacrificing its capability to detect parallel and overlapped activities. The novel segmentation algorithm utilizes both the location and temporal information for segmenting occupancy episodes which enables the system to detect ADLs and IADLs when more than one person is present as well as in constrained apartment where several activities occur in the same room. The system achieves average accuracy as high as 97% in all the tested datasets. *SARRIMA* performs better than other contemporary systems applied on the same datasets [17].

- We present a solution to recognize the user of each activity from only passive binary sensors. The idea that a particular user have certain way of performing some activity lets us utilize the behavioral history data and identify the person based on different set of triggered sensors. In this paper, we introduce the concept of occupancy sessions, which utilizes the location information of users residing in different room at a particular time. The system identifies a significant percentage of activity user by cross referencing with other already assigned occupancy episodes in the same occupancy session. However, since the accuracy of such solutions vary depending on the home, number, and type of sensors, and the behavior and relationship of users - we perform a simulation to show how much improvement can be made by adding a limited number of non-binary passive sensors. Very high accuracy can be achieved only by replacing or adding very few non-binary passive sensors.

We tested the system with data from two-residence homes. However, we believe that the system will work in homes with more residence, although the accuracy might be lower. Nonetheless, there are not many homes where there are more than two residents and where ADL recognition is required.

### 4.3 System Assumptions and Overview

*SARRIMA* operates based on the assumption of spatial, temporal, and object regularity of ADLs, i.e., “particular activities of daily living are usually performed in some specific room area and generally occur at the same time of day and offer same objects are used to perform a particular activity”. Although not all activities are always performed in the same area (examples include talking on cell phones, having a conversation, or cleaning the house), most of the ADLs do. Therefore, those IADLs (like talking on cell phones) are out of the scope of this work. The temporal regularity is useful to differentiate activities like preparing breakfast or preparing dinner but is not a hard requirement. If certain activities (watching TV, drinking water) do not have this regularity, they will still be recognized by the system where the time of day parameter associated with that activity class will be considered NULL. If an activity class does not follow object regularity, then multiple class definitions are created by *SARRIMA* for the same activity class. Table 3.1 shows the activity classes recognized by *SARRIMA*.

In the *SARRIMA* deployment setting, sensors are positioned around the place where a particular ADLs take place. Therefore based on the spatial assumption, a specific set of sensors get triggered whenever that ADL is performed. This approach has already been applied on single-person residences [16], but the presence of multiple people introduces randomness and therefore complexity in defining the Activity Classes. Figure 4.1 shows the overall architecture of *SARRIMA*.

**Sensing Layer:** The purpose of this layer is to sample the sensors, process and organize the data in terms of sensor values, corresponding time-stamps information, and associated room IDs. The processed information is transferred to the next layer.

**Creating Occupancy Episodes:** The time duration intervals during when someone is present in a room are defined as ‘Occupancy Episodes’. The system assumes that activities in a room occur only during these occupancy episodes, i.e., when someone is present in the room. Occupancy episodes are determined from the sensor firing timestamps.

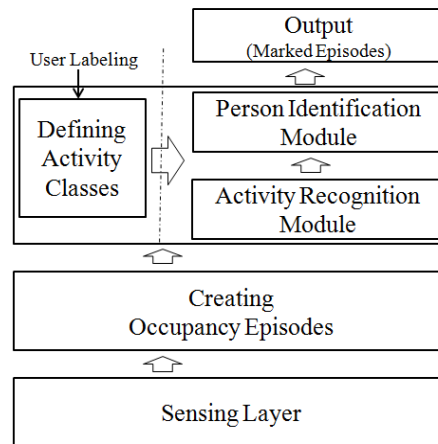


FIGURE 4.1: Overall system architecture of SARRIMA

**Activity Recognition Module:** This module recognizes what activities are performed in each of the occupancy episodes of each room based on the definitions of activity classes. Activity classes are defined in terms of the set of sensors which fire during performing the activity and the temporal statistics (start time and duration) of the activity class if the activity instances of that class show any temporal regularity.

**Person Identification Module:** This module identifies the user of an activity by exploiting the difference of users in performing the activity, the relationship among the recognized activities, and the relative position of each user in the home setting. The system is provided with the information of the number of users in that house and an associated ID for each user.

**Output:** The output of *SARRIMA* gives a list of  $\langle Activity, PersonID \rangle$  pairs for each room.

### 4.3.1 Sensing Layer

In our system setting, all the sensors are assumed to be non-wearable wireless sensors. Examples include (but not limited to) contact sensors, motion sensors, binary pressure pads, infrared sensors, and temperature sensors. The type and number of sensors may vary from house to house. The sensing layer of *SARRIMA* takes raw sensor data as input. It is also provided with the information of room ID associated with each sensor during the initialization of the system. If the sensor setting changes, the information needs to be updated, and the system needs resetting. Sensing layer processes the input data and creates a sequence of pairs of the form  $(s_i, t_{1i}, t_{2i}, v_i, r_n)$  where sensor  $s_i$  is deployed in room  $r_n$ , and has the value  $v_i$  from time-stamp  $t_{1i}$  to time-stamp  $t_{2i}$ . The data is sorted in the ascending order of start time.

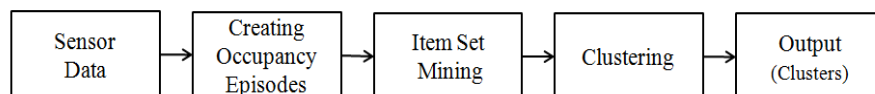


FIGURE 4.2: Training Framework for Defining Activity Classes

### 4.3.2 Occupancy Episodes

In a single resident home, sensors fire only in the room where the user is present. Therefore, occupancy episodes can be calculated by grouping the consecutive sensor firings from the same room and taking the time interval of the first and the last sensor firing timestamp. However, in a multiple resident homes the sensors firing in a particular occupancy episode will not be consecutive due to the presence of other users in other rooms. Therefore, the actual leaving and entering moment unrecognizable and ambiguity arise in determining the exact duration of occupancy episode. This ambiguity is resolved by considering the assumption: "A user is unlikely to leave while performing an activity and if no sensor fires for a certain amount of time, then the room is empty." Therefore, 'occupancy episodes' is created in two steps:

**Room-wise Separation:** In this step, *SARRIMA* separates the sensor events of different rooms into different files. Therefore, each file contains a collection of sensor events sequentially listed based on the starting timestamp ( $t_{1i}$ ) of the corresponding room.

**Consider time difference of consecutive sensor firing:** This time limit is defined as *timeThreshold*. If a sensor fires after that time, then a new occupancy episode starts. Therefore, the time difference between two consecutive sensor firings in an occupancy episode is always less than or equal to the *timeThreshold* of that room.

Thus, each occupancy episode of a room is a specific time duration and during that duration sensors fire depending on what activities were performed at that time. Therefore, an 'Occupancy Episode' is represented in the form (*roomId*, *startTime*, *duration*, *usedSensors*) where *usedSensors* is the set of sensors that fired during the episode. Based on the object regularity assumption, if a certain set of sensors in the list of *usedSensors* are frequent, then those sensors fired due to performing a particular ADL.

### 4.3.3 Defining ADL Classes

Before proceeding to the following sections, here, we shall briefly describe how *SARRIMA* uses the training framework (Figure 4.2) of AALO [16] to define activity classes. The activity classes of *SARRIMA* are defined in terms of the sensors used, and the temporal (if any) characteristics (start time and duration) associated with the activity class. The frequency item sets (the unique combination of sensor sets that fires together) are determined by applying the itemset mining algorithm *APRIORI* [63]. To determine the temporal characteristics of an activity class, *SARRIMA* applies the density-based clustering algorithm *DBSCAN* [64] which does not need to specify the number clusters in advance. This is important since the numbers of different activity classes in each room are not predefined. Now, for each frequent itemset  $FI_i$  of a room, *SARRIMA* runs *DBSCAN* separately on the set of tuples ( $startTime_{ik}$ ,  $duration_{ik}$ ); here  $k = (1, 2, \dots, \text{number of occupancy episodes})$  where  $FI_i$  occurs. Each attribute of each tuple is normalized before clustering. In *DBSCAN*, the number of clusters depends on the threshold parameter. *SARRIMA* calculates and uses the lowest threshold parameter that gives the maximum number of clusters but a minimum number of unrecognized instances for each  $FI_i$ . Each of the clusters signifies a particular activity class. The system outputs all the clusters with associated parameters to the user for labeling. Therefore after labeling, each activity class  $A_i$  is represented by the tuple ( $usedSensors_i$ ,  $meanStartTime_i$ ,  $meanDuration_i$ ,  $neighborhoodRadius_i$ ,  $label_i$ ,  $personID_{optional}$ ). Here,  $personID_{optional}$  indicates the user (if any) associated with the activity. If the activity class definition is general for all users, then  $personID_{optional}$  is null.

#### 4.3.4 Activity Recognition

SARRIMA uses the following steps to recognize activity instances of each occupancy episodes:

**Step 1:** It finds all the activity classes whose *usedSensors<sub>i</sub>* is a subset of the sensors fired during that occupancy episode.

- If there is no such class, then the recognized activity instance is NULL.
- If a single class matches the condition, then the Activity ID of that class is marked.
- Otherwise, go to Step 2.

**Step 2:** This step checks the temporal characteristics of the matched classes. If the *starttime* and *duration* of the occupancy episode is within the *neighborhoodRadius<sub>i</sub>* of the (*meanStartTime<sub>i</sub>*, *meanDuration<sub>i</sub>*) of an activity class, then the Activity ID of that class is marked. However, the duration feature has given less importance since the duration of an occupancy episode varies based on the value of *timeThreshold* used for calculating it.

Multiple ADL instance might be recognized for a particular occupancy episode. This happens because the user may do multiple activities in the same occupancy episode, or multiple users doing different activities can be present in the same occupancy episode.

#### 4.3.5 Person Identification

The purpose of this module is to identify the person who performed the activity. It is done in three main steps by incorporating the following information:

**Step 1:** User Differentiating features.

**Step 2:** Relative position of the users.

**Step 3:** The relationship among the recognized activities.

Each of the steps is described below:

##### User Differentiating Features

In this step, SARRIMA uses the information gathered from the sensors directly or indirectly to identify a user. The success of the identification process in later steps depends on this step.

**Step 1a: Behavioral Difference** We consider the behavioral difference as the unique way a person performs an activity. However, in this work, the difference is only considered in terms of the sensors fired, duration, or time of the activity performed. Usually, these differences are marked during the user labeling step of training. Therefore, the labeling is used to identify a user of an activity.

**Step 1b: Biometric Difference** The behavioral difference in performing activities are not prominent in all households. Therefore, additional specialized sensors might be required to identify a user. The most common low-cost passive sensors used for this purpose are height sensors, weight sensors, microphones, or videos. Since the success of user identification in later steps depends on Step 1, introducing specialized sensors might be necessary for certain households.

##### Relative Position of the Users

The main idea behind this step is to link the occupancy episodes of each user (i.e., separate the episodes of a different user) without directly identifying the user. For example, if SARRIMA

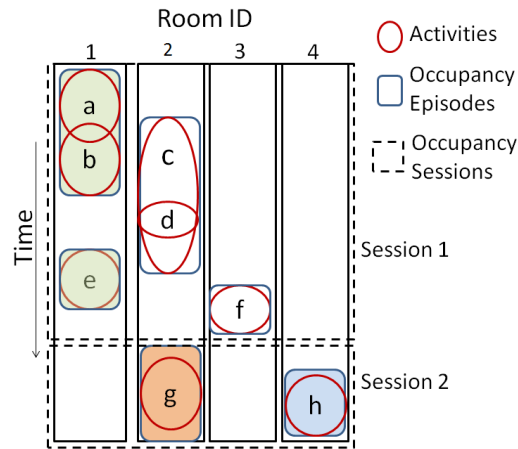


FIGURE 4.3: Creating occupancy sessions

detects two activities being performed in two different rooms at the same time, then those activities are being performed by two different users.

**Step 2a: Creating Occupancy Sessions** An occupancy session is a collection of back-to-back and overlapping occupancy episodes from all the rooms of the house. We introduce the term occupancy session to define a smaller time frame where all the occupancy episodes can be linked or separated from each other based on the associated user. The start time of an Occupancy Session is the start time of the earliest occupancy episode and the end time is the end time of the occupancy episode that finishes after all episodes in that session has finished. Now, as long as a new occupancy episode overlaps with one or more episodes occurring in a different room, it can be separated from those and therefore included in the same occupancy session. Whenever *SARRIMA* finds an occupancy episode that starts after all the previous episodes of the session have finished, it increments the current session number and adds the occupancy episode in a new session. The process continues until all the occupancy episodes are placed under some occupancy session.

**Step 2b: Co-relating Occupancy Episodes (Activity groups)** The occupancy episodes of a particular session can be separated or co-related based on their spatial and timing parameters. The rules for co-relating occupancy episodes in a particular session are described below. The description assumes a two-person setting for simplicity.<sup>1</sup>

- **Occupancy episodes in different room:** The occupancy episodes of a particular person should never overlap, since he/she cannot be in multiple rooms at the same time. Therefore, if the system detects two occupancy episodes occurring at the same time in two different rooms, then it places the two occupancy episodes in separate groups. For example, in figure 4.3, occupancy episode with activity *b* is placed in a different group than activity *c*, *c* in a different group than *e*, and *e* in a different group than *f*. Therefore, in a two user setting, the occupancy episodes with *a, b*, and *e* are co-related and the activity

<sup>1</sup>In a *n*-person setting, the problem of linking/separating occupancy episodes can be solved by converting it to a graph problem. Here, the episodes can be considered as nodes and the episodes occurring at the same times will have edges. The *n* activity groups can be found by dividing the complement of the graph into *n*-maximally connected sub-components.

instances performed by the same user, whereas the occupancy episodes with  $c$  and  $f$  are co-related and performed by the remaining user.

- **Occupancy episodes in same room:** All activities detected in a particular occupancy episode is assumed to be done by the user present in that episode. However, if the system detects only a single occupancy episode at a particular time, then most likely all the residents are in the same room. In that case, it is assumed that the activity can be done by any user and thus no assignments are made. For example, the user of activity  $d$  in Figure 4.3 cannot be determined with certainty.

The set of co-relating occupancy episodes in the same occupancy session is defined as an activity group. Theoretically, all the occupancy episodes of a particular session can be determined by detecting the user of any occupancy episode (from Step 1) within that session. However, if the initial information is wrong then all the episodes will be incorrectly marked. Therefore, SARRIMA marks an entire session if it has several marked occupancy episodes. Otherwise, no assignment is done for the unmarked episodes of that session.

### The Relationship among the Recognized Activities.

**Step 3a: Similar activities** For certain activity classes, if the existence of activity instances from the same class is detected within a certain time in two different groups, then the activities are considered to be performed by two different users. For example,  $A$  and  $B$  are groups in *occupancy session 1* and  $C$  and  $D$  are groups in *occupancy session 2*. If the system detects brushing teeth in both groups  $A$  and  $D$ , then SARRIMA merges the two sessions where groups  $A$  and  $C$  are merged into one larger group and groups  $B$  and  $D$  are merged into another larger group.

**Step 3b: Complementing activities** Some activities instances frequently occur together. For example, people go to toilet after waking up in the morning or before going to bed. Similarly, using a wardrobe can be detected before/after taking a shower. If these complementing activity instances are detected within certain time periods, then SARRIMA considers these activities to be performed by the same user.

The unmarked episodes are marked in the same way as it is done after Step 2.

#### 4.3.6 Parallel and Overlapping Activities

SARRIMA retains the detection capability of AALO in finding the parallel and overlapping activities. However, the user of the activities can not be identified in all the scenarios. If two or more activity instances occur at the same time, then they are called parallel activities. In a multi-user scenario, it is very common and the number of such instances increases as the number of user increases. SARRIMA detects parallel activities by detecting all possible activity instances in a particular time frame (occupancy episode). It assigns all activities to a user if he/she is the only one present. However, if multiple users are detected in the same episode, the system refrains from making any user assignment (unless it matches personalized activity class definition) and considers all the detected persons as a probable performer of the activity



## 4.4 Evaluation and Discussion

There are two publicly available datasets for recognizing ADLs from passive wireless sensors which have homes with more than one person [13], [27]. Each of the datasets has data from multiple homes and labeled ground truth for different sets of activities. The homes have different floor-plans, different types of sensors, and different demographics. Table 3.1 shows some example activities annotated in the datasets. The ARAS dataset [27] has a total of two months of data of 27 different type of activities; collected from two real homes - House A and House B. The sensor types included contact sensors, force sensors, photocells, pressure mats, distance sensors, sonar distance, IR, and temperature sensors. Each of the CASAS [13] datasets (“CASAS Spring 2009 multiperson dataset” and CASAS Summer 2010 multiperson dataset”) have three months of data of 11 different type of activities. Seventy-two sensors were deployed in each of the houses which includes motion sensors, door/contact sensors, and temperature sensors.

The algorithms of SARRIMA are implemented on MATLAB. We used cross validation technique where one-fifth of the data was considered as the training sample size.

### 4.4.1 Results: Activity Detection

SARRIMA assigns one or more activity instances to each occupancy episode based on the triggered sensor set and the temporal features of that episode. To minimize the number of false positives, occupancy episodes with very small durations (length  $\leq 10$ s) are filtered before making the assignment.

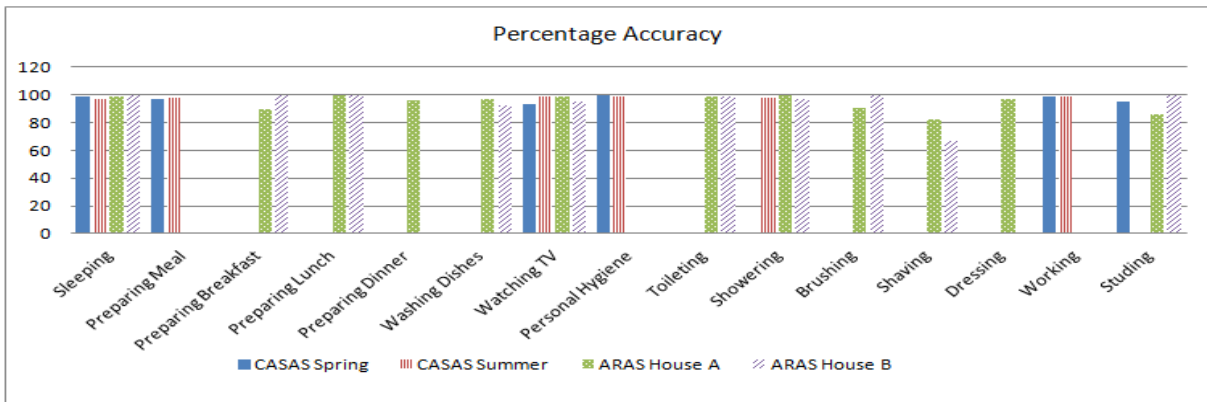


FIGURE 4.4: Percentage of activity instances of Activity Classes recognized correctly in CASAS Spring 2009, CASAS Summer 2010, ARAS House A, and ARAS House B (time\_threshold = 2 minutes)

Figure 4.4 shows the percentage of accuracy of recognizing instances of different ADL classes in four different houses. The missing columns in figure 4.4 indicates that no ground truth labeling is found for the corresponding activity class in that particular dataset. We see that SARRIMA achieved 97.34% and 98.15% accuracy on average respectively on the CASAS Spring and Summer dataset. The accuracy is higher than HMM and SHMM (92% on average) applied on the same dataset [13]. In ARAS houses, the activity classes have more fine grained definition. For example, ‘preparing breakfast’, ‘preparing lunch’, and ‘preparing dinner’ instead of ‘preparing meal’ or ‘toileting’, ‘brushing’, and ‘shaving’ instead of just ‘personal

hygiene’. Therefore, relatively lower accuracy is achieved when considering these activity class definitions (average accuracy of 87% for House A and 95.3% for House B). The reason is the activity class instances of one user is often confused with an closely related activity class of another user. However, the average accuracy of activity detection is more than 98% in both houses when general ADL class definition (similar to CASAS houses) is considered.

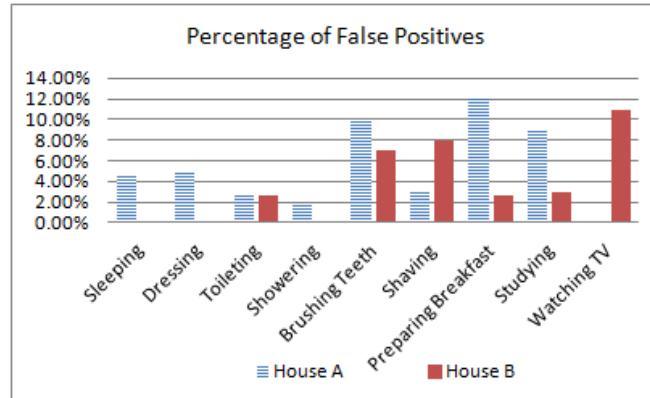


FIGURE 4.5: False positive of reported activity instances ARAS (House B) (time\_threshold = 2 minutes)

CASAS had less than 2% false positives for any activity instances. Whereas, higher number of false positives is observed in the ARAS dataset due to more fine grained definition of each of the activities (figure 4.5). We notice that less false positives occurs in House B for certain activity classes which were often performed together (eating breakfast or dinner). Whereas the activity ‘Watching TV’ has zero false positive in House A due to the choice of sensor (Infrared sensor to detect TV light instead of pressure pad to detect user in the couch) to detect it. The number of false positive in bathroom and kitchen reduces when higher *timeThreshold* is considered for creating occupancy episodes. However, in that case occupancy episodes have longer duration and therefore longer activity duration is reported for some activity instances.

#### 4.4.2 Results: Person Identification

Most activity instances of the CASAS dataset do not have user specific ground truth except for sleep, work, and bed\_to\_toilet. However, the two users in CASAS always performed the sleep, work, and bed\_to\_toilet in separate rooms. Therefore, SARRIMA recognizes the user of annotated user specific instances just by checking room location. Consequently, we achieve a 100% accuracy for this simple home living situation.

ARAS has more complex activity class definitions and all the activities of each user is labeled. Therefore, the following evaluation will focus only on ARAS datasets.

##### User Assignment from Behavioral Difference

SARRIMA uses the behavioral difference of users to define a personalized activity class. For example, in House A the two people sleep in two different room and have different times for using the toilet in the morning and at night due to difference sleeping schedule. On the other hand, in House B Person 2 always sleeps on the left side of the bed and ‘preparing meal’ is always done by person 1.

Figure 4.6 shows the percentage of activity instances assigned correctly to a user for particular activity classes based on behavior difference. Although the accuracy is as high as 100% for

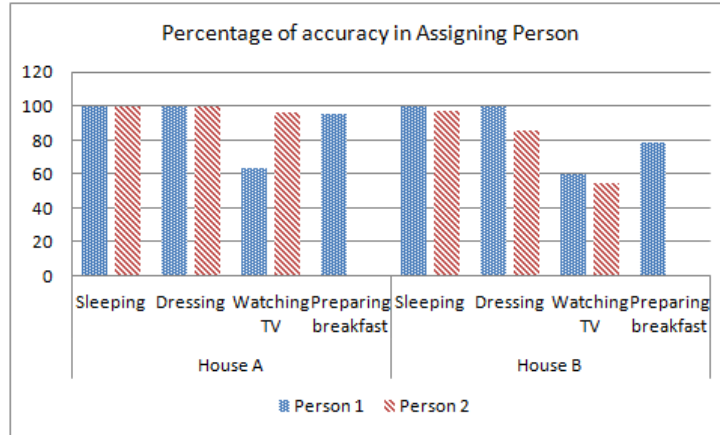


FIGURE 4.6: Accuracy of user identification from behavioral difference for particular activity classes where difference is observed in users way of doing the activities (ARAS Dataset).

some activity classes (Sleeping and Dressing), low accuracy is observed for the classes where user behavior is not consistent ('Watching TV' from different location in the room). Here, we want to note that the user behavior will not be same in every resident and thus there might not be any identifiable difference between two users performing the same activity. However, if differences exist in term of the sensor activated or the time of the activity, then the system recognizes the differences. The labeling of the classes helps to identify the differences of user behavior. The labeling is done only once after the framework generates the cluster. Therefore, not much extra user effort is required for this step.

TABLE 4.1: Effect of user identification using behavioral information and episode linking (threshold = 3 min) for a single day in ARAS House B

	User identified correctly in percent of episodes	
	(Behavioral)	(Correlating)
Bed	100%	-
Bath	0	47.3%
Living	37.5%	31.8%
Kitchen	0	42.85%
Hall	0	40%

**Linking Occupancy Episodes** SARRIMA correlates occupancy episodes of the same user based on the overlapping episodes of different rooms. This steps identifies additional activity instances for which no user behavioral difference is observed. Table 4.1 shows the episode information of a randomly chosen day from Aras (House B). The first column shows the percentage of occupancy episodes correctly identified in each room by SARRIMA based on behavior alone. The second column shows the percentage of additional episodes where a user is correctly identified by using the SARRIMA feature of linking the episodes with already identified ones. This shows the value of this feature of SARRIMA.

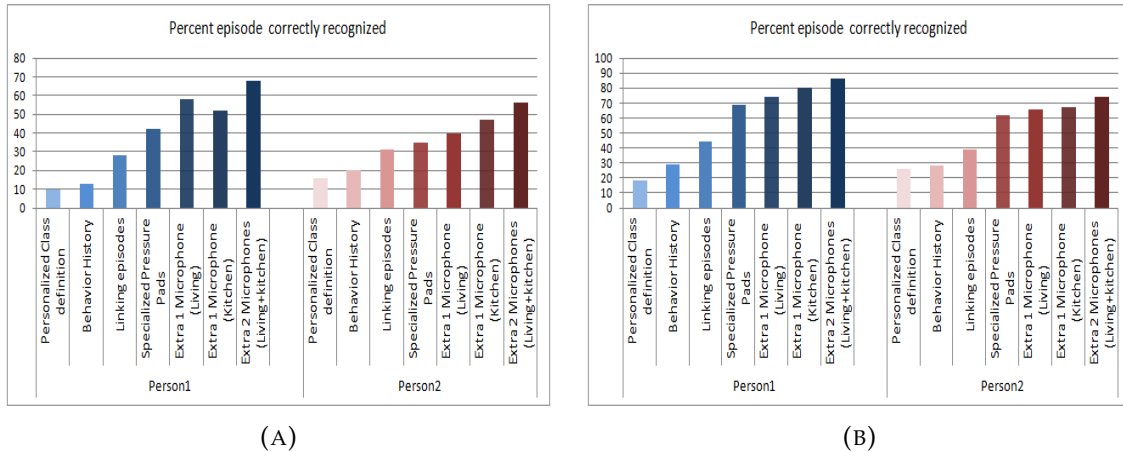


FIGURE 4.7: Person Identification Accuracy (threshold=2 min) (a) House A (b) House B

**Simulating Biometric Difference** As mentioned in section 4.3.5, the accuracy of person identification of linking episodes depend on already identified episodes. However, for most of the activity classes there was no identifiable behavioral difference and the datasets do not have sensors that can differentiate user based on biometric features. Therefore, we have simulated sensor readings to show how changing or adding a few sensors can have huge effect on accuracy.

*Weight sensor:* We have chosen to simulate weight sensor (specialized pressure pads) since ARAS dataset already have a few binary pressure pads. We have simulated the data using actual data and user annotation information, where the pressure pad shows actual user weight instead of a binary value. The pressure pads in the bedroom did not provide additional information since bedroom activities are already included in the definition of the personalized Activity class. However, the pressure pads on the dining room chair identifies the user who is eating a meal and the pressure pads in the living room identify the user of activities ‘Watching TV’, and ‘study’.

*Microphones:* A common biometric feature to distinguish between user is speech and recent works show [65] that users can be distinguished with over 85% even with utterance length of 2s and the accuracy gets higher with longer speech sample. Due to privacy concern, we only considered the position of the microphone in kitchen or living room. Again, since there is no certainty whether the user will talk or not. We filtered the occupancy episodes with less than five minutes based on the assumption that nobody talked during those episodes. SARRIMA does not have algorithms for speaker identification, but uses an external tool for this purpose [65].

Figure 4.7 shows the percentage of activities correctly assigned to each person of House A and House B. The accuracy is not high enough if only binary sensors are considered, however it increases significantly just by replacing the binary pressure pads with non binary ones and adding one or two microphones. We can observe from the figure that the effect of adding specialized sensors on accuracy varies from person to person and that using only a few of them can help to increase the accuracy to near 100%.

The ARAS datasets have been used in other research work for ADL and person identification. In the paper [27], the researchers used a hidden Markov model (HMM) and obtained an average accuracy of 61.5% for House A (with min=46.3% and max=88.4%) and 76.2% (with

min=31.1% and max=96.7%) for House B. In paper [17], the researches applied an incremental decision tree model and obtained classification accuracy 40% for House A and 82% for House B. The low accuracy in these previous works demonstrates the significant difficulties in person assignment. SARRIMA performs significantly better than these systems with adding few specialized passive sensors. Although, other systems might have performed very high with these extra sensors, their requirement in labeling all the activity instances during training period makes them difficult to use in actual homes.

## 4.5 Conclusion

In this chapter, we presented *SARRIMA*, a system for detecting activities of daily livings in the presence of multiple people. For the given datasets, *SARRIMA* is capable of detecting 97% of the activities on average which is higher than a HMM (92%), and it also reports parallel and overlapping activities. Importantly, *SARRIMA* identifies the user of the activity without using wearable sensors or RFID tags. The paper shows how *SARRIMA* achieves very high accuracy for person identification by using a history of personal behavior, linking occupancy intervals across rooms, and by including appropriate sensors in good locations when necessary.



## Chapter 5

# Activity Step Recognition: Challenges and Lessons Learned

In this chapter, we present our experience and lessons learned while designing the "Activity Step Detection and Recognition System", and collecting data in real home settings. Detecting activity steps itself is not essential unless utilized for other purposes. Therefore, the step recognition system is coupled with an activity recognition or activity quality monitoring system (chapter 6 and 7). However, the main goal of the chapter is not to describe the performance results of activity step recognition, but to highlight the unique challenges it offers (section 5.2). We elaborate on our experience and the design decisions we made while building the system. We also discuss our idea for handling or minimizing some of the challenges and the effectiveness of those techniques (section 5.3).

### 5.1 Introduction

While working on the problem of activity recognition we noticed that often activity recognition accuracy lowers when very similar activities are present in the set. Such as 'Brushing Teeth' and 'Shaving'. Such activities are very different from a human perspective, but almost identical from system/sensing perspective. Therefore, we hypothesized that if we gather enough fine-grained information about each activity by defining and recognizing activity steps, then we would be able to differentiate similar activities. Moreover, our intuition told us that such the research of activity steps would have an impact on future applications in different areas:

1. Differentiating activities of two persons if the steps are different.
2. Identifying sudden behavioral changes if activities are done differently.
3. Providing health-related information about Alzheimer disease if steps are missing or wrong.

Therefore, in this chapter, we look into the details of finding activity steps from sensor data. We used existing data sets for evaluation. But since the idea of activity step is new, it has not been explored in a large scale. Most of the publicly available data with in-home ADL/IADL provides only the ground truth of the start and the end of the performed activity, but not what steps occurred in between. Very few datasets have the notion of an activity step, but the ground truth of each step is not always labeled. Therefore, we could perform analysis up to a certain extent. To look into the problem more deeply, we decided to design experiments from scratch. Although, this gave us a lot of freedom in defining what activity steps to detect, or what sensors to use - the decision for choosing from the vast open opportunities was not easy. Therefore, in this chapter, we explain our reason for each design decision.

As our research goal was not focused on activity step recognition, but its utility at a higher level, we used traditional machine learning algorithms established in activity recognition and applied them for recognizing activity steps. When detecting activity steps from in situ sensors, we use our version of *Apriori* item set mining described in chapter 4. However, instead of outputting a particular activity, the algorithm outputs activity steps. Activity and gesture recognition from a wearable in the literature mostly uses HMM supervised machine learning [66]–[68]. In addition to applying HMM, we also applied other standard supervised methods such as Naive Bayes, Decision Tree, Random Forest and K-NN (chapter 3 section 3.2)

However, we were disappointed with the preliminary evaluation results. As we investigated the reason, we identified exciting observations. In our field, often the researchers collecting the data are not the same persons who perform the analysis. Therefore, when something goes wrong, they can only assume the reason, but could hardly find concrete evidence. As being part of both the data collection team and the data analysis team provided us with insight into why certain things did not work out as we expected. We looked back into the videos for collecting ground truth for finding evidence supporting our intuition. Although we plan to publish the data after de-identifying, the videos cannot be made public for privacy issues. Thus, we believe that our experience and observations are valuable to document for future researchers.

Later, some of the problems were resolved or mitigated by applying innovative fixes (section 5.3). However, many of the challenges still need attention from the research community.

## 5.2 Challenges

In this section, we describe the challenges we faced during the project. A lot of these challenges are also observed in high-level activity recognition from sensors, but the complexity level is higher when recognizing portions of activity, i.e., activity steps. Besides, there are some unique challenges we observed particularly in this area. We also mention our idea or insight on handling certain complexities, or the reason for choosing a particular trade-off option.

### 5.2.1 Activity Step Definition

*Q. How to find detailed information about performing an activity? What is the level of granularity in collecting fine-grained information of the activity process?* We all know activities

are performed in small steps. However, there is no strict scientific term defining an activity step. Some researchers have shared their own ideas about the topic. In chapter 3 section 3.1.3, we elaborated some of the related terms. However, the definitions are often molded in a way that serves the purpose of the researcher’s study objective. Since our ultimate objective is to use the activity steps for monitoring activity quality, we looked into the literature documenting activity process of dementia patients who has difficulty in performing steps or misses the activity step [13]. A physiologist was observing and keeping notes of the activity quality. One of the examples of an activity step was *Writing a birthday card left on the table*. As a computer scientist, we stumbled upon such definitions of an activity step. Because a human can understand when a person is having difficulty in writing, but with the current technology how can a system identify such random events? Therefore, from computer science and system design perspective, an activity step must be recognizable in terms of current technology, i.e., the underlying sensing system. Another condition we deem necessary is the atomicity. Because if



an activity step can be partially done, then the level of granularity has to be even finer; which introduces unnecessary complexity. Therefore, in this thesis, we define an activity step as,

*"An activity step is a portion of an activity considered as a unit which cannot be done partially and is recognizable by sensing technology."*

## 5.2.2 Sensing Technology

### *Q. What sensors are best for detecting activity steps of in-home ADL and IADL?*

As the above definition mentions, a system recognizing activity steps is as versatile as the underlying sensing platform's capability permits. Therefore, choosing the appropriate sensors is extremely important.

In the activity recognition literature, different sensing technology has been explored. Some of them are summarized in chapter 2 section 2.2. However, in many cases activities are inferred from the result. But if we want to detect partially finished activities or attempted activities, the results are not always evident. Therefore, recognizing activity steps requires more sophisticated capability.

If we look into the previous example of *Writing a birthday card left on the table*, a motion sensor or a pressure pad on the chair is able to detect if a person is near the table or sits on the chair. A smart wearable is able to detect hand motion. But detecting the writing itself is very tricky, especially for a general purpose system detecting all sorts of gesture events.

We experimented with different in-situ and wearable sensors in the lab and our personal houses before deciding which ones to use for data collection in a real home setting. Here we list our experience and explanation of different case studies.

### **In-situ Sensing Platform**

Passive in-situ sensors are popular in the field of activity recognition for preserving the privacy of the users, because they do not store any user-identifiable features. However, there is a vast number of available in-situ sensors, which ones to choose?

In our earlier experiments for detecting activities from in-situ sensors, we had used the X-10 sensing platform. However, X-10 sensors had limitations in the number of devices a modem can operate simultaneously. Moreover, missing sensor values make it difficult to have clean data and reduces the robustness of the system. In contrast, the off the shelf Z-wave binary sensors are much better for robustness issues. Therefore, we used the z-wave technology in our experiment. The z-wave sensors also looked prettier, which might seem a trivial fact, but is actually quite important when sensors are deployed in real homes of non-research people. However, there are some limitations of these sensors when we use them for recognizing activity steps:

**Effectiveness:** Not all activity steps can be recognized with passive binary sensors. For example, caregivers of patient's with mild cognitive impairment mention that often the patient forgets to put ingredients in their cooking. For instance, they might forget to add coffee in the cup. But how can a binary in-situ sensor detect it?

One way of solving the problem can be using item sensors, which works like small pressure pads for objects. We can detect whether the coffee container has been replaced with item sensors, but it would require putting the container exactly in the same place every time. Even healthy adults are not that consistent. If some other container is put instead, the item sensor cannot tell the difference.

Thus, *"an in-situ sensor is effective only when the state of the object being monitored is changed"*.

However, not all the in situ sensors are uniformly useful in serving their purpose. The usefulness depends both on sensor's power and how they are used.

1. **Door sensors and Pressure pads:** Door sensors were used for detecting opening and closing of refrigerator doors, microwave-oven, cupboard, drawers, dressers, sliding doors, water taps, and oven knobs. Pressure pads were put on top of the study chair, dining chair, sofa, and under the bed cover to detect whether someone is sitting on it. We found these sensors to be very helpful in satisfying our goal.
2. **Motion sensor:** We placed motion sensors in different places of the home to track user location. Although, the sensor values provide an overall idea about someone's presence when a person stays at a particular place for some time, but often misses if the person just passes by. Therefore, they are not helpful in providing the trajectory path of a user. In a single-user environment, if the person moves very slowly (assuming a person with movement problem), the motion sensor is helpful, but it still does not provide user-direction information. Besides, a motion sensor can tell the presence of a person and might give an idea about what activity is being performed if only a single activity occurs in that area. However, it is not helpful in providing information related to which activity steps were done and which were not.

**Instrumentation Difficulty:** Instrumenting the environment is another issue we had to face while working with z-wave sensors. First of all, not all objects can be instrumented; therefore not all object state change can be measured. In long term deployment, attaching sensors is also a problem, since they seem to fall off with repetitive usage of the object. Besides, deploying in-situ sensors in a home requires effort, and since every house is different, the deployment also requires thinking in terms of the sensor layout plan. The same strategy does not always work in all apartments. For example, we could attach a door sensor with stove knob in some homes but not all, because the way the stove knob was designed differently.

**Quantity:** Another limitation of using in-situ sensors is that there can only be a certain number of sensors that can be put in the environment. For example, the z-wave modem we used let us attaching 30 z-wave devices. In a sense, the number is quite high for a small apartment. However, in big houses 30 sensors might not be enough. Moreover, the cost of the sensing platform increases proportionally as the number grows. The deployment also becomes more laborious, since each sensor has to be attached individually. Thus, we have to limit the number of in-situ sensors.

Nonetheless, in-situ binary sensors are relatively cheap. The battery of each sensor lasts one year before needing a replacement. Most importantly, some of the events (from door sensor/pressure pads) are detected with absolute certainty. Finally, there is no extra effort needed for de-identification since it preserves privacy by default. Therefore, we used z-wave door sensors, binary pressure pads, and motion sensors (figure ??) in our experiment.

### Hybrid Sensing Platform: RFID

The main idea behind looking into RFID sensors was to instrument as many objects as possible with low cost. We found examples in the literature where RFID tags were attached in items, and a person carrying an RFID reader triggered unique codes by touching individual objects. In other words, an RFID tag triggers the same code each time it is near an RFID reader, and the code is unique for each RFID tag. Therefore, we purchased both low-frequency and high-frequency tags, and both low frequency RFID reader (we refrain from mentioning the

name) and a high-frequency RFID reader (Chafon 13.56MHz Mini USB HF RFID Mifare IC Card Reader). The reader is reported to have a range of 5cm-9cm. However, we finally decided to not use the technology for the following reasons:

**Difficulty Handling the RFID Reader:** RFID reader works by generating code when it is placed near to an object. Therefore, we first tried putting the reader around our hand like a smartwatch. However, it did not work due to the increased distance of the reader from the object being held. Next, we tried attaching the Reader with our gloves, which seem to somewhat serve our purpose.

However, the other problem was that the reader itself does not have processing power. Therefore, we had to connect it with a windows tab (the RFID Reader driver is written for Windows OS) via a USB cable. We needed to carry the tab around us which was very uncomfortable.

**Usefulness of RFID tags:** We experimented with two types of tags having respective low-frequency (range 125 – 134 kHz) and high frequency (13.56 MHz). We did not use ultra high-frequency RFID tags because if all the objects near the RFID reader sends code, then we cannot uniquely identify what object was used.

The low-frequency tags were very cheap, very thin, and could easily be attached with objects. However, the low-frequency RFID reader was very bulky. The biggest problem was, it took 12 to 15 tries to generate one code. Probably there were other better tags and readers, but the prospective did not look very promising.

The high-frequency tags have the shape and size of credit cards and was effective in generating codes. The main problem was attaching them to object since the cards were flat and hard.

**Tag and Reader Distance:** Although the Chafon RFID Reader is supposed to read tags within 5cm-9cm, we found it requires to be within 2cm near of the medium frequency tags. Therefore, if the object was handled in a way where the tag was on the different side no code was generated. The reader also did not generate any code when the object was handled really fast. Therefore, we had to move very consciously and hold object keeping in the mind that we want the code to be generated. That makes a user movement too restrictive and will likely not work in a real home.

Here, we want to mention that we are not implying that RFID technology is not practical. It is very widely used in different applications. It just did not serve our purpose. There are other RFID readers and tags available in the market which claims to provide better performance. But those were costly, and since our main objective is to help to build a system serving affordable health-care, we did not try them out. It also requires concentrated study specifically in that area and out of scope from our actual objective.

### **Wearable Sensing Platform: Smartwatch**

In recent years, there has been a tremendous advancement in the field of wearable technology. Small, powerful devices filled with various sensors are available in the market. Moreover, since most in-home daily self-care activities are performed with hand, using a hand worn wearable device opens up vast opportunity in detecting very fine-grained steps. Therefore, in our data collection sessions, we used a smartwatch in addition to in-situ sensors for detecting a number of variety of activity steps.

In our enthusiasm of exploring the opportunity given by off-the-shelf smartwatch device, we defined activity with a lot of details and labeled 54 different activity steps or gesture events

to be recognized from wearable sensors. Figure 3.2 shows the signals from the x-axis of the accelerometer (top) and the x-axis of gyroscope respectively (bottom) for different every day activity steps. However, while performing the data analysis and applying classification algorithms, we realized that we might have defined too many activity steps. Many of the challenges specific to wearable sensors are due to the reasons mentioned in later paragraphs. Therefore, we will not re-iterate them here and proceed to the next section.

### 5.2.3 Number of Classes of Activity Step

This particular challenge is closely related to the first challenge mentioned above, i.e., the definition of an activity step. In a particular home, the number of ADL classes are fixed and depending on datasets there are 5/6 ADL classes defined. Even though there is a higher number of IADL classes, the IADLs performed at home are limited. On the other hand, depending on how rich the underlying sensing platform is and how granular the activity process is divided into steps, there can be a large number of possible activity step classes. For example, 'stirring', 'chopping ingredients', 'opening door', 'closing cabinets', 'typing', 'rinsing an item', 'storing an item', 'sweeping the floor', 'wiping a surface', and so on. As the number of different classes increases, the probability of misclassification becomes higher. Moreover, it becomes very complex for gathering training data for all the different classes of activity steps.

### 5.2.4 Same vs. Similar Activity Steps

Ideally, the same activity step should give consistent signal across all the different types of activities. However, it is not always true for all the activity steps. For example, 'stirring' while making coffee is not always same as 'stirring' while cooking due to the difference in pot size and the thickness of the content. One might argue that these two types of 'stirring' should be considered as different activity steps. In that case, the complexity arises by creating huge number of activity steps classes. Moreover, finding all such different classes is neither possible nor practical.

On the other hand, some different class of activity steps are very close in terms of signal. For example, in figure 3.2, we see that completing different activities 'Adding Sugar' in a cup of coffee with a spoon has a similar signal with 'Sitting on a Sofa'. Although, to human eye the signal in the image might look distinguishable, from algorithms point they are very similar. Especially, when it is classifying all the different instances of 'Sitting on a Sofa' from all the different instances of 'Adding Sugar'.

### 5.2.5 Data Processing

#### Segmentation

Due to the sheer number of different activity steps possible there are steps that have very short duration (sudden spike in data) and long duration. It is confusing figuring out whether a sudden movement is an activity step with small duration or part of a continuous activity step with a larger duration? It makes the time series data processing harder in terms of framing and windowing, i.e. finding the size and starting position of the window. Again, a fixed length window is not a option since a larger window size will miss the many activity steps with smaller duration and a small size window only captures portion of long activity steps, the decision of how to vary the window size is very crucial.

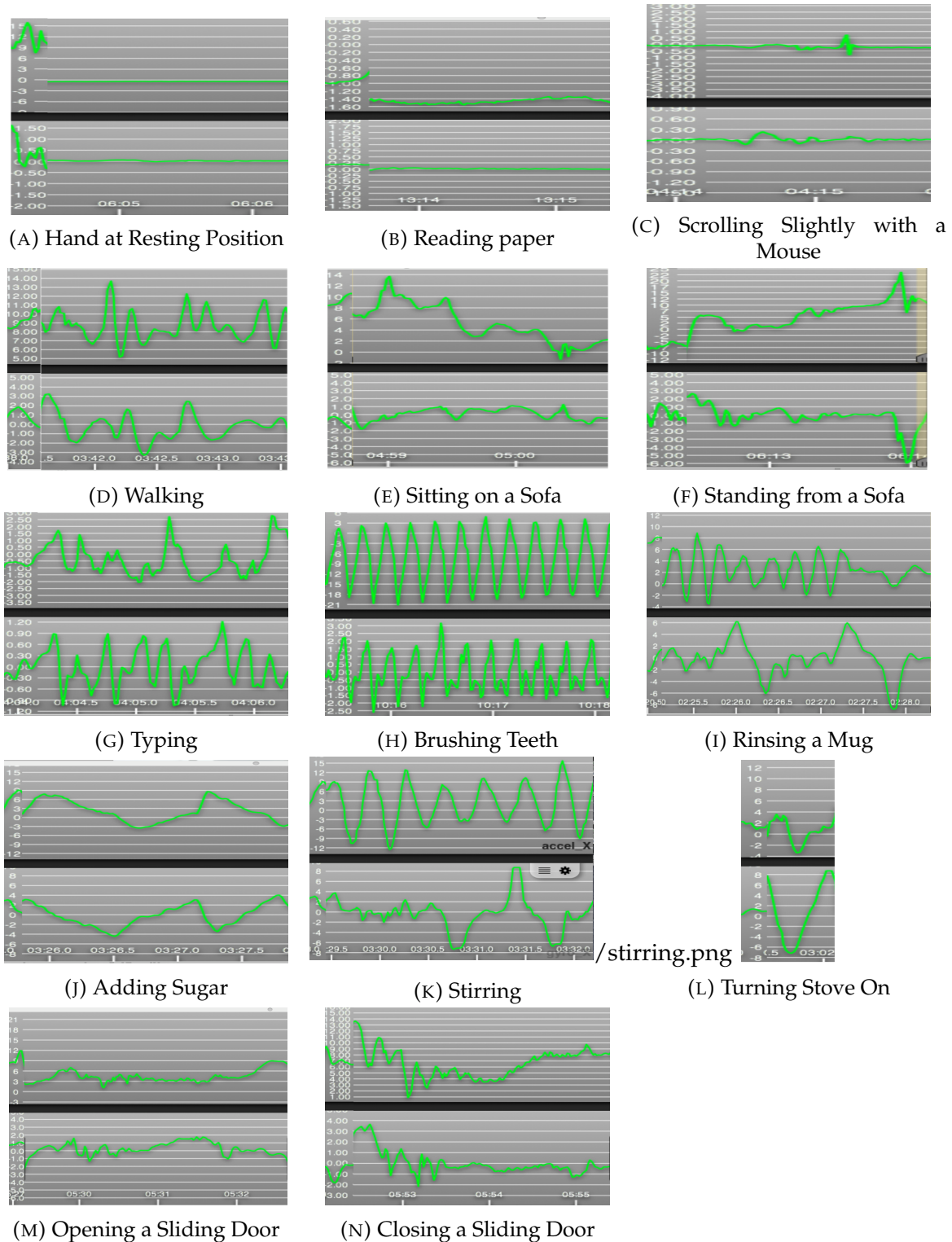


FIGURE 5.1: The above images show time series data of accelerometer x-axis and gyroscope x-axis corresponding to different gestures. The snapshots are taken from the 'Chonoviz' visualization software tool. The x-axis labels in each image show time in 1s intervals (except (L) which shows in 0.5s intervals). The y-axis shows the normalized value of the accelerometer and the gyroscope in a fitted zoomed position.

## Classification

Identifying distinguishable features to classify activity steps is very difficult due to the sheer number of possible classes. On top of it, the system requires separating meaningless gestures from actual activity step. For example, during the day people often move their hands subconsciously to do things like touching the hair, scratching nose, or popping finger joints. Often, these gesture signals create confusion recognizing the activity step and the activity being performed. Again, processing every spike detected from gesture signal is energy inefficient and creates a lot of false positives.

## Energy

The biggest problem of using a wearable device is energy consumption. Many smartwatch application developers deal with this problem by making the watch sleep and periodically collecting data. This method does not work in our case, since finding all the different activity steps needs the watch to collect data continuously. It creates a huge bottleneck in terms of energy consumption. Another way of dealing with energy problem is to turn off the sensors (such as, gyroscope) that consume more energy. This is also not possible for us since even using all the sensors it is hard to classify and recognized activity steps. A lot of researchers sample data at low frequency to preserve battery life. However, as mentioned above, some activity steps are very short in duration. To capture the signals generated from those steps, a higher sampling rate is required which drains the battery faster.

### 5.2.6 Human Factor

As part of data collection and data processing, we had the chance to observe and understand many human factors that otherwise go unnoticed. The challenges below are spotted when collecting data of human actions. Although we only used a smartwatch on the hand, many of the challenges are also applicable to other wearable sensing devices as well.

#### Biometric difference

Each person is unique in some way. However, a person's biometric difference, such as the difference in height, weight, and strength, changes the smartwatch signals in case of some activity or activity steps. For example, a tall person operates differently in a kitchen than a shorter person. The weight and strength also cause the signal to vary from person to person when the activity being performed requires a lot of movement and strength, such as cleaning the house. Age is also a factor since both kids and seniors operate differently than the young or middle-aged person. Therefore, training a system in the lab with healthy adults do not give the same performance when evaluating activities of an elderly person at home.

#### Hand motion

The smartwatch signal captures hand motion. However, the hand motion signal is different when the same activity step is performed faster or slower. Positioning the hand in different angles also creates a distinction. We also observed that some people keep the wrist fixed and moves the whole arm, whereas others move the wrist or finger a lot which changes the relative arm movement. This arm vs. wrist movement is noticeably observed in activities like typing

in a keyboard. But this variable also affects performing activities with tools such as stirring with a spatula or washing dishes with a sponge. Another interesting fact is movement in other body parts sometimes create motion in hand. For example, movement is found in the acceleration signal from the smartwatch of a resting person in sofa whose legs were bouncing back and forth.

### **User Energy Level**

While collecting data from the same person for several days, we noticed that based on her mood or energy level, the way she performs changes. The change is mostly in terms of speed. Since a tired person with low energy acts slower. Therefore, the difference is observed in activity speed from late night activities vs. activities performed before going to work. Another reason might be that in the morning, a person tends to hurry to reach timely for work and therefore acts faster.

### **Personal Preference or Habit**

Personal preferences or habit changes the wearable sensors single pattern in an interesting way. For example, while collecting data, we observed when a person retrieves something from the lower cabinets, some bend their torso, while others sit and reach out for the item from the cabinet. Therefore, the activity step 'retrieving an item' is not the same for those two cases. Such examples are not due to biometric differences, because we observed both tall and short person behaving either way. Thus, these ways of doing things are mostly due to habit or personal preference.

### **Parallel Actions**

Human is able to multitask seamlessly. The tasks we are talking about are not always the different ADL/IADLs. Therefore, from a system point of view many tasks are invisible. However, they do affect the activities at hand, some more than the others. For example, when a person is performing an activity while having a conversation with someone or laughing, for some activity the signal differs. It happens because while talking the person takes many small pauses from the activity and the speed also changes. Sometimes an angle in a specific sensor axis is observed when the user periodically moves and faces the person he/she is having a conversation with. These parallel actions are less observable when in a single person home. But a single person may also read or watch tablet while doing an activity.

There are likely to be features independent of scenarios described above, but finding those features require separate attention and collecting data having those various examples.

## **5.2.7 Appliance Arrangement**

The position of furniture, appliance, drawer, or shelf matters when detecting activity steps that uses them. For example, opening lower cabinets definitely produces different signal than the operating the higher ones. Similarly, retrieving items from drawers or shelves at different heights produce different signals. Although, these challenges are intuitive, but when defining activity such as 'opening a cabinet' or 'retrieving an item', the problems become evident.

Another complexity is observed when a person operates the appliance from a different location or position. Here, the location boundary is very small. For example, when someone is

only washing the dishes, he/she usually washes items standing in front of the basin. However, when the same person needs to rinse an item while cleaning or cooking, we noticed changes in the signal. We found the reason for this variation is because of the change in relative positioning. For example, the user just tilts his/her body a little bit to reach out to the sink and quickly washes the item. Thereby, creating a difference in motion signature

## 5.3 Ideas and Lessons Learned

In this section, we summarize our lessons learned after facing so many challenges. These lessons helped us to grow and think from a different perspective. We hope documenting the lessons helps others too.

### 5.3.1 Scope of the Problem

After spending a considerable amount of time and effort in identifying all the different kind of activity steps, we understood recognition of activity step, in general, is not a trivial problem. The huge number of possible activity steps and the challenges found in classifying the steps is tremendous. Since our ultimate goal was to detect the quality of the in-home ADL/IADLs, we tried to encompass everything that fits in the scope. However, for future researchers, it would be advantageous to select a small number of activities and identify steps from those activities only. Making this selection would require expert opinion based on the application. For example, a doctor or caregiver close to an Alzheimer patient could help select the activities where the patient has a problem completing it. Moreover, if the activity occurs in a certain place in the home or a certain time of day, then every other convoluting gestures can be ignored. However, the purpose of recognizing activity steps vary based on the target application. Therefore, picking a particular challenge from our given that is unavoidable in the target application and focusing on solving it would result in better performance in that area.

### 5.3.2 Importance of Sensing Platform

An activity step needs to be recognizable, and therefore the choice of sensor is critical. The decision of which sensors to use depends on what activity steps are defined. Inversely, defining the classes of activity step requires information about the sensing capability. Therefore, these two major decisions are coupled and interdependent. As we have listed in the previous section, each choice has its own advantages and disadvantages; therefore there is no right answer. However, the researcher should not make a choice randomly and consider carefully based on the application in hand.

### 5.3.3 Necessity of Clean Data

In our experiment, we let the users move freely according to their own will. Although we provided a list of high-level activities that require to be performed, we did not specify the activity steps or made them taking a pause from switching one step to another. The reason was to make the experiment as close to the real environment as possible. However, we later realized that only natural data is not good enough for training the system due to the all the variations that occur. Therefore, we had to conduct experiments with constraint environment define some specific activity step and collect data for each activity step separately. Training



the system by using both the clean data and data from the home setting performed better in identifying activity steps.

### 5.3.4 Advantage of Sensor Fusion

One of our key idea of solving the time-series data processing was to apply sensor fusion. Since we were already collecting data from both in-situ and wearable sensors, we planned to use this hybrid platform in our advantage.

#### Data Segmentation

To address the problem of finding variable window size, we imported the sensor values from in-situ sensing data and superimposed the values detected from the environment sensors, i.e., Z-wave contact sensors, pressure pads, and motion sensors on top of wearable time-series data. The discrete values/ value change from the sensor gives a good idea about the start and end time of the activity step. For example, a person might open the refrigerator to retrieve an item (short duration), store groceries (long duration), or to clean the refrigerator (very long duration). Again, the duration of storing the groceries or cleaning the refrigerator is not fixed in all different instances. However, since the door sensor attached on the refrigerator door triggers both when opening or closing the freeze, we use that time for windowing the time series data to find the activity step from the wearable device. This gives us an idea of when the activity is being performed and focus on recognizing activity steps from the time series data around that time. Therefore, a lot of unnecessary processing is avoided.

#### Class Reduction

Another advantage of using the in situ sensors is that it gives an overall idea about what activity is being performed. However, since it cannot tell all the activity steps, wearable data is helpful. By using sensor fusion, the possible number of activity steps that could occur can be eliminated and the remaining classes of activity steps become more manageable. For example, we previously mentioned that 'sitting on a sofa' gave a similar signature to 'adding sugar' activity step. We can easily find our what step actually occurred by looking into the value of the pressure pad sensor on the sofa and the door sensor attached to the cabinet containing the sugar case.

#### Energy Salvation of Wearable

Energy can be saved if wearable sensors collect data only after getting a command from the main system. The main system may send the command whenever it detects in situ sensors triggering. However, since we processed our data offline and received data from both types of sensor platform independently, we did not apply this technique. Nonetheless, the technique could be useful for future researchers

One disadvantage of using hybrid sensing is the additional cost and dealing with the disadvantages of both platforms. For example, wearables need to be adequately charged and deploying and maintaining lots of in-situ sensors still requires effort. However, it is up to the researcher and user of the system to decide whether the advantage outperforms the disadvantages.



## Chapter 6

# Activity Quality Framework

In order to notify users about potentially unsafe situations and to track mistakes or efficiency performing activities, it is crucial to monitor the quality of performing an activity and identify the missing/wrong steps. However, the state-of-the-art activity recognition frameworks ignore such details and impose constraints on sensor values, the types of detected activities (no parallel/interleaved/joint activities), or the number of users, which reduce the robustness of the system in the real world settings. Therefore, in this chapter, we present *QuActive*, a grammar-based general purpose framework for modeling activities and activity steps that retains the details of the activity process, quantifies activity quality, and notifies users about missing steps and unsafe situations. To show the versatility of *QuActive*, we evaluate the framework on three different public datasets that have interleaved activities, parallel and co-operative activities, and activities of cognitively declined patients with quality information labeled. In all cases, *QuActive* outperforms the state-of-the-art techniques applied to these data sets. Besides, we have deployed the system in a real home and collected data in a semi-controlled setting to evaluate the performance of the system in real environments.

The rest of the chapter is organized as follows. We briefly describe the motivation and challenges of the work in section 6.1, followed by the contribution (chapter 6.2). Section 6.3 describes the *QuActive* framework and our approach, and section 6.4 describes the *QuActive* system design and implementation. Finally, the evaluation and discussion are presented in section 6.5.

## 6.1 Motivation and Challenges

In today's smart world, wearable and in-situ sensors are being used to monitor humans and recognize many types of activities. In most cases, the resulting information is not acted upon in any direct or real-time manner. However, by more intimately bringing the human into a feedback loop, there is an excellent potential to use interventions and notifications to improve human activities. For example, by focusing on the activity steps of an activity, it is possible to detect the quality of a performed activity instance and dynamically react to improve that activity, if necessary. This human-in-the-loop Real-Time reaction is important in home health care systems to keep patients safe, in industrial process monitoring systems of factory workers to ensure the safety of workers and the quality of products, and so on. Without considering the activity steps of an activity, controlling the quality of the activity is difficult.

Most current activity recognition systems recognize whether an activity has occurred or not, but do not identify partially completed activities or the missing steps in the overall activity process. Hence, they cannot easily offer notifications and interventions in real-time to improve

performance. In addition, many current systems [7], [8] make too many simplifying assumptions about the environment, the number of users, etc. that either limit the types of recognized activities, or tailor the system to perform well in simple situations such as single person homes and no concurrent activities [9], [10]. It is necessary to consider interleaved, parallel, and cooperative activities for more robust and realistic activity recognition.

Our first hypothesis is that pushing the activity recognition constraints to a lower level solves the limitations of many existing systems. In this paper, we consider the fact that activities are composed of activity steps, where a user can perform only one activity steps at a time but can switch in between performing activity steps of a particular activity and do portions of other activities. For example, someone can chop ingredients for a meal, feed the dog, and then come back to use the ingredients for cooking. The activity steps provide important information about the high-level activities, such as whether the activity is complete or partially complete, the different ways a certain activity is performed, whether the activity has missing steps, how a missing step affects the overall activity quality, and how intermediate delays among  $\mu Acs$  might influence the overall activity quality. Therefore, this chapter addresses the problem of detecting activity steps in realistic settings and how to incorporate the human-in-the-loop by offering real-time notifications to improve performance. Since a step of an activity involves using objects resulting from specific gestures by a person, *QuActive* incorporates information from both wearable and in-situ sensors.

One challenge is how to model the activity process in terms of activity steps. The steps within an activity can occur in parallel or sequentially. The activity process also varies depending on the person, environment, or situation. Different activities often have similar activity steps, and steps performed in a different order might result in the same or a different activity. Thus, the process of mapping activity steps to distinct activities capable of handling these variations is vital. Another challenge is addressing the deviation from normal activity processes. For example, if a particular step is missing or performed out of order, then is the activity incomplete, wrongly performed, or still a valid activity performed differently? How to keep a general structure of a particular activity which is performed in different ways? How to identify the prospective/incomplete activity when one or more steps are missing? Finally, how and when to bring the user more intimately into the loop via notifications and interventions?

To address the mentioned challenges, the *QuActive* framework is created based on a Temporal Probabilistic Context-Free Grammar (TPCFG) to define the activity process (details in 6.3.2 and 6.3.3). The context-free grammar (CFG) follows the basic definition from literature [69] that includes terminals, nonterminals, and rules. However, the terms are tailored for defining particular activities and activity steps. The grammar outlines a general structure for each activity. Activities (nonterminals) and activity steps (nonterminals) are generated from rules. Rules are applied iteratively until terminal symbols (sensor values) are reached. Any future activity instance is recognized from the defined grammar representation. Again, to capture the variation of performing the same activity, multiple rules are added to represent the same nonterminal term. If an activity is performed in several ways, then a probability (P) is associated with each of the rules defining the same nonterminal. The timing parameter (T) is used to capture the time information [70] of each activity step as well as the time difference among two consecutive steps. Rules have notifications attached to them. Thus, *QuActive* utilizes the grammar structure to handle the challenges of modeling variations of activity process of the same activity. In addition, the *QuActive* system implementation defines parameters and thresholds for generating notifications.

## 6.2 Contributions

In this thesis chapter, we summarize the main contributions of this work:

1. This chapter presents *QuActive*, a novel activity modeling framework that utilizes fine-grained information about the activity process and uses that for notifications. *QuActive* is capable of monitoring activity quality and reporting prospective activity in case of missing steps and other realities in contrast to other state-of-the-art detection systems [18], [71].
2. We implemented a system that incorporates a *QuActive* framework to recognize in-home activities, monitors the quality of the finished or partially completed activity, and notifies users. The notification subsystem modifies the latest voice-based medication reminder system, Med-Rem [14], to an activity reminder system that provides audio alerts about activities, informs the user about missing steps, and stores user feedback.
3. The *QuActive* framework is applied to three different public datasets of interleaved activities, parallel and co-operative activities, and monitoring cognitive decline (missing steps and activity quality). *QuActive* outperforms the state-of-the-art techniques for all of these datasets.
4. The system has also been deployed in a real home in a semi-controlled setting. The results show that *QuActive* recognizes more than 90% of the defined activity steps and the grammar detects 98.6% of the defined activities from the recognized steps.

## 6.3 QuActive Framework

The core of the presented system is the *QuActive* framework. As mentioned before, the framework is based on Timed Probabilistic Context-Free Grammar (TPCFG). The rules of the grammar define activity steps in terms of the processed sensor information and activities in terms of the activity steps. Since grammar rules are applied iteratively, intermediate stages of recognized activities are defined as partial activities. The *QuActive* framework has the following advantages:

- **Manage Variation:** Multiple rules are added to represent the same activity that is performed in different ways. For example, making coffee ‘using a coffee maker’ or ‘using hot water and instant coffee packs’ have different grammar representations.
- **Handles randomness:** While making tea, the activity steps of ‘adding sugar’, ‘adding milk’, ‘adding tea’, and ‘pouring hot water’ do not require any specific order. However, ‘heating water’ must be done before ‘pouring hot water’, and ‘stirring’ is always the last step. These collections of ordered and unordered terms are handled in the *QuActive* framework.
- **Reusable:** Some activity steps of an activity process are observed in other activities. For example, ‘adding sugar’ occurs in ‘making tea’ or ‘baking cake’. Thus, activity step definitions are reused in defining new activities.
- **Extensible:** People may perform activities differently due to a change of habit. The activity process may also change when new technology or different appliances/objects are used. These changes can be handled just by adding new rules to *QuActive*, without requiring changes to the overall framework.

Section 6.3.3 presents the general structure of the *QuActive* framework in terms of TPCFG symbols irrespective of any activity class and section 6.3.4 describes an example grammar of particular activity class ('Making Coffee'). But before that section 6.3.1 defines an activity step and section 6.3.2 gives a mathematical definition of PCFG from the literature.

### 6.3.1 Definition and Properties of Activity Steps

An activity step is the smallest activity step that cannot be decomposed any further. The chapter assumes the definition and properties of *micro – activity* mentioned in chapter 3 section 3.1.3 as the definition of an activity step. We re-iterate the properties here for readers convenience.

The following statements hold true for a  $\mu Ac$  (activity step):

- i An activity can be broken into one or more  $\mu Acs$ . So, a  $\mu Ac$  can be an activity itself. For example, 'heating water' can itself be an activity or a  $\mu Ac$  of 'making tea'.
- ii  $\mu Acs$  cannot be done partially, i.e., once started a  $\mu Ac$  has to be finished, or otherwise it is disregarded.
- iii  $\mu Acs$  can occur in different activities. For example, the  $\mu Ac$  'using water' can be a part of the activity 'washing dishes' or the activity 'mopping the floor'.
- iv Although every activity is associated with one or more users, and every  $\mu Ac$  is associated with some activity, the  $\mu Ac$  itself might be independent of a user. For example, a user triggers the switch to boil water, but water boiling itself is independent, and the user may do something else during that time.

### 6.3.2 Timed Probabilistic Context Free Grammar

A context free grammar (CFG) is a type of language generator. It is expressed as  $\langle V_N; V_T; Start; R \rangle$ , where

- $V_N$  is a finite set of nonterminal symbols. Nonterminals are represented with words starting with a capital letter.
- $V_T$  is a finite set of terminal symbols. Terminals are represented with words starting with lower-case letters.
- $V_N \cap V_T = \emptyset$ .  $V = V_N \cup V_T$  is called the vocabulary and  $V^*$  is the set of all strings of symbols in  $V$  including the string of length zero.
- $Start \in V_N$  is the start symbol.
- $R$  is a finite nonempty subset of  $V_N \times V^*$  called the production rules.

A CFG where multiple rules define the same non-terminal can be extended to a probabilistic CFG [69] as an ordered five-tuple  $\langle V_N; V_T; Start; R; P \rangle$ , where

- The production rules are paired with a set of probabilities  $\{p_{ij}\}$  that satisfy the following rules.
  - For each production rule  $R_{ij} \in R$ , there is one and only one probability  $p_{ij} \in P$ .
  - $0 < p_{ij} \leq 1 \forall_{i,j}$

- For every  $i$  with  $1 < i \leq |V_n|$ ,  $\sum_{1 \leq j \leq n_i} p_{ij} = 1$ , where  $n_i$  is the number of productions with the  $i$ -th nonterminal on the left-hand side.

Timed PCFG is an extension of PCFG where timing information is incorporated into the grammar rules [70].

### 6.3.3 TPCFG for Detecting Activity

In this section, a general definition of TPCFG for an activity class is provided. Each activity class has a separate set of rules that follow this general structure. The terminals of the grammar are sensor values and rules are iteratively applied until the terminals are reached. The activities and activity steps are nonterminals. The following rules show how an activity can be composed of a sequence of steps.

Activity	→	ActivityStep TmDiff PartialActivity
Activity	→	ActivityStep
PartialActivity	→	ActivityStep TmDiff PartialActivity
PartialActivity	→	Activity

Here, the rules for ‘PartialActivity’ are necessary for generating an unambiguous grammar. After iteratively applying the rules, all ‘PartialActivity’s are decomposed until only the activity steps are left. The nonterminal ‘TmDiff’ indicates the time difference between two consecutive steps. A negative duration value of ‘TmDiff’ indicates an overlap between the two activity steps. For example, heating water while adding coffee can occur in parallel.

An activity step can be associated with more than one sensor, since several sensor events can occur at the same time. For example, several motion sensors can be triggered when a user enters a particular location.

ActivityStep	→	Event
Event	→	(Event, Event)
Event	→	(Sensor, Time, Value)

The above rules show how each activity step is associated with one or more sensor events. The comma separated tuple indicates that the sensor events are independent of each other. Each sensor event indicates the change of a particular sensor value at some specific time or during a specific duration.

Sensor	→	InsituSensor   Wearable
InsituSensor	→	motionSensorID   contactSensorID   tempSensorID   pressurePadID
Wearable	→	smartWatchID

Now, a sensor is either a wearable device or an in-situ sensor. The above rules show a sensor setting where a smartwatch is used as a wearable device, and motion sensors, contact sensors, temperature sensors, and pressure pads are used as in-situ sensors.

The value of the sensor is either a numeric value (0 or 1 of a binary pressure pad sensor) or values of a list of features extracted from the continuous sensor signal (e.g., features calculated from the accelerometer or the gyroscope data of a smartwatch).

Value	→	num   feature
-------	---	---------------

Although ‘Time’ and ‘Tmdiff’ both provide timing information, one is associated with a sensor event and the other indicates the time difference between two consecutive steps as mentioned earlier. The nonterminal ‘Time’ is associated with each sensor event. The *starttime* indicates the time when the sensor value changes and *duration* indicates how long the sensor value remained constant (or was above/below threshold in case of continuous data).

Time	→	(starttime, duration)
TmDiff	→	(starttime, duration)

Grammars defined for all the activity classes maintain this described structure. The grammar described here does not show the probability (P) for simplicity. The probability is associated with each rule when multiple rules define the same nonterminal.

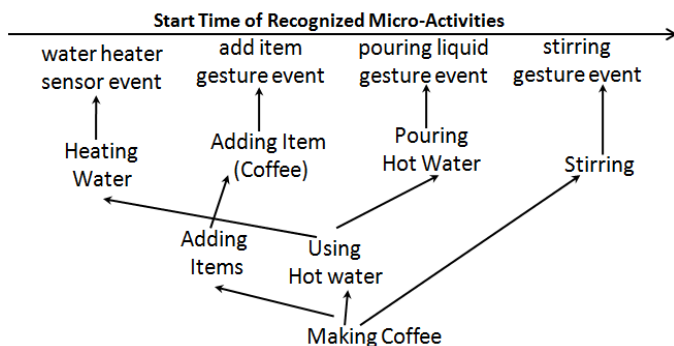
MakingCoffee( $p_{11}$ )	→	(UsingDrinkware) (UsingUtensil) (UsingHotWater) (AddingItems) (Stirring)
MakingCoffee( $p_{12}$ )	→	(UsingDrinkware) (UsingUtensil) (AddingItems) (UsingHotWater) (Stirring)
UsingDrinkware( $p_{21}$ )	→	(MovingObjectGestureEvent)
UsingDrinkware( $p_{22}$ )	→	(OpenCupboardEvent) (MovingObjectGestureEvent)
UsingUtensil( $p_{31}$ )	→	(MovingObjectGestureEvent)
UsingUtensil( $p_{32}$ )	→	(OpenUtensilDrawerEvent) (MovingObjectGestureEvent)
UsingHotWater( $p_{41}$ )	→	PouringWaterGesture
UsingHotWater( $p_{42}$ )	→	HeatingWater PouringWaterGesture
AddingItems( $p_{51}$ )	→	AddingItems*
AddingItems( $p_{52}$ )	→	AddingItem
AddingItem( $p_{61}$ )	→	AddingItemGesture
AddingItem( $p_{62}$ )	→	OpenCupboardEvent RetrievingItemGesture AddingItem
AddingItem( $p_{63}$ )	→	PouringLiquidGesture
AddingItem( $p_{64}$ )	→	OpenRefrigeratorEvent RetrievingItemGesture AddingItem

TABLE 6.1: TPCFG for activity ‘Making Coffee’.

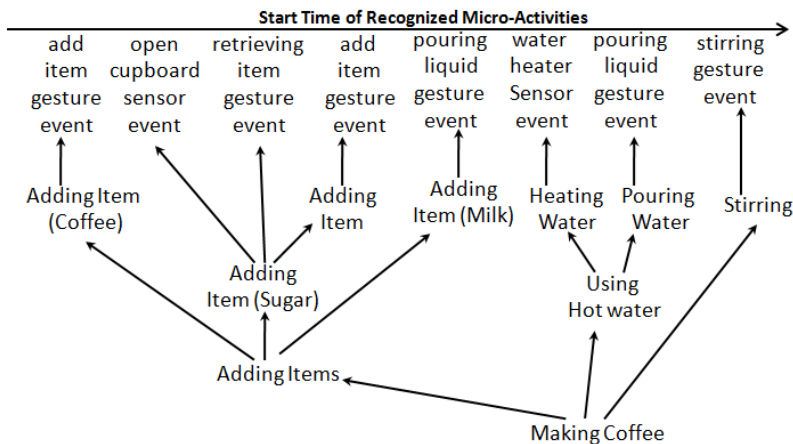


### 6.3.4 Example Grammar and Parse Tree

Table 6.1 shows one example of possible different rules for making coffee with start symbol as ‘*MakingCoffee*’. To make the rules clear, the timing parameter has not been shown. However, each event is associated with the  $Time(starttime, duration)$  information and each rule contains the  $TmDiff$  information between two consecutive terms on the right side of the rule. As we can see, multiple rules cover different situations such as when the items or utensils are retrieved from the refrigerator, cupboards, or drawers, as opposed to already being placed on the countertop. The associated probability represents the probability of that situation occurring. If a high probability rule does not match, other relevant rules are applied. One limitation of the system is that it cannot identify the exact added item. For example, if somebody adds only sugar instead of coffee, *QuActive* still recognizes the Activity ‘Making Coffee’. However, the limitation is associated with the sensing system and not directly related to the *QuActive* framework. In the future, if the sensing capability enables distinguishing each item, then similar rules can be added to make the grammar richer.



(A) The parse tree shows a way of ‘Making Coffee’ in which no sugar or milk is added in the coffee, and the coffee is added in between heating and pouring water. It also assumes that the drinkware, utensils, and ingredients (mug, spoon, and coffee) are already placed on the countertop, i.e., no object is retrieved from the cabinet or the drawer.



(B) The parse tree shows a situation where multiple items (milk, coffee, sugar) are added in the coffee and only one item (sugar) is retrieved from the cabinet.

FIGURE 6.1: Example parse trees showing different ways of performing the same activity (‘Making Coffee’).

Figure 6.1 shows two examples parse trees depicting ‘Making Coffee’ in two different situations. In the first tree, only coffee is added whereas in the second tree milk, and sugar are added to the coffee. However, in both situations, it is assumed that drinkware (mug) and the utensil (spoon) are already placed on the countertop. A different and larger parse tree will be generated in situations where the drinkware and the utensil need to be retrieved from their storage places. The intermediate levels of the tree show nodes related to partial/sub-activities. However, some order is maintained in all the different situations. For example, heating water always precedes pouring water (although other activity steps can take place in between) in order to identify the higher level activity ‘Using Hot Water’. Again, stirring is always performed last. Although timing information has not been shown explicitly, each low-level event retains time information which propagates up to the root (highest level activity) of the tree.

## 6.4 System Design

In this section, we describe the system architecture that uses the *QuActive* framework to recognize fine-grained activity steps, construct high-level activities, and extract activity quality parameters.

### 6.4.1 Sensing Layer

The sensing layer consists of both in-situ sensors and wearable sensors for collecting detailed activity information. Wearable sensors are placed on a user’s body to collect gesture information related to the activities. The system assumes a smartwatch as the wearable device containing accelerometer, gyroscope, and magnetometer. On the other hand, binary contact switches, binary pressure pads, motion sensors, and temperature sensors work as in-situ sensor nodes. Therefore, the sensing layer collects human motion that causes an activity as well as events related to the effect of resulting activities on the surrounding environment.

### 6.4.2 Event Layer

This layer preprocesses the sensor data and lists all sensor events. Whenever the status of a sensor is changed, an event is triggered. For example, a pressure pad triggers the event ‘occupied’ if somebody sits on it and triggers ‘empty’ whenever the person leaves. The environment sensors are assumed to generate discrete sensor events. On the other hand, the sensors in the smartwatch generate continuous data streams at a particular sampling rate. A threshold value is used to filter the normalized time series data where no significant motion is detected by the accelerometer and gyroscope. The filtered segments denote possible gesture events. Time information from environment sensors is provided to identify segments where gestures are more likely related to some activity step as well as to trim segments to find the approximate start and end time of a gesture. For example, by aligning the contact sensor events ‘opening refrigerator’ and ‘closing refrigerator’ with wearable sensor data, *QuActive* finds signals related to ‘storing groceries’ and ‘retrieving groceries’.

### 6.4.3 Activity Step Layer

This layer constructs the activity steps or possible steps from the information provided by the event layer. Although a user can leave in the middle of an activity, based on the properties

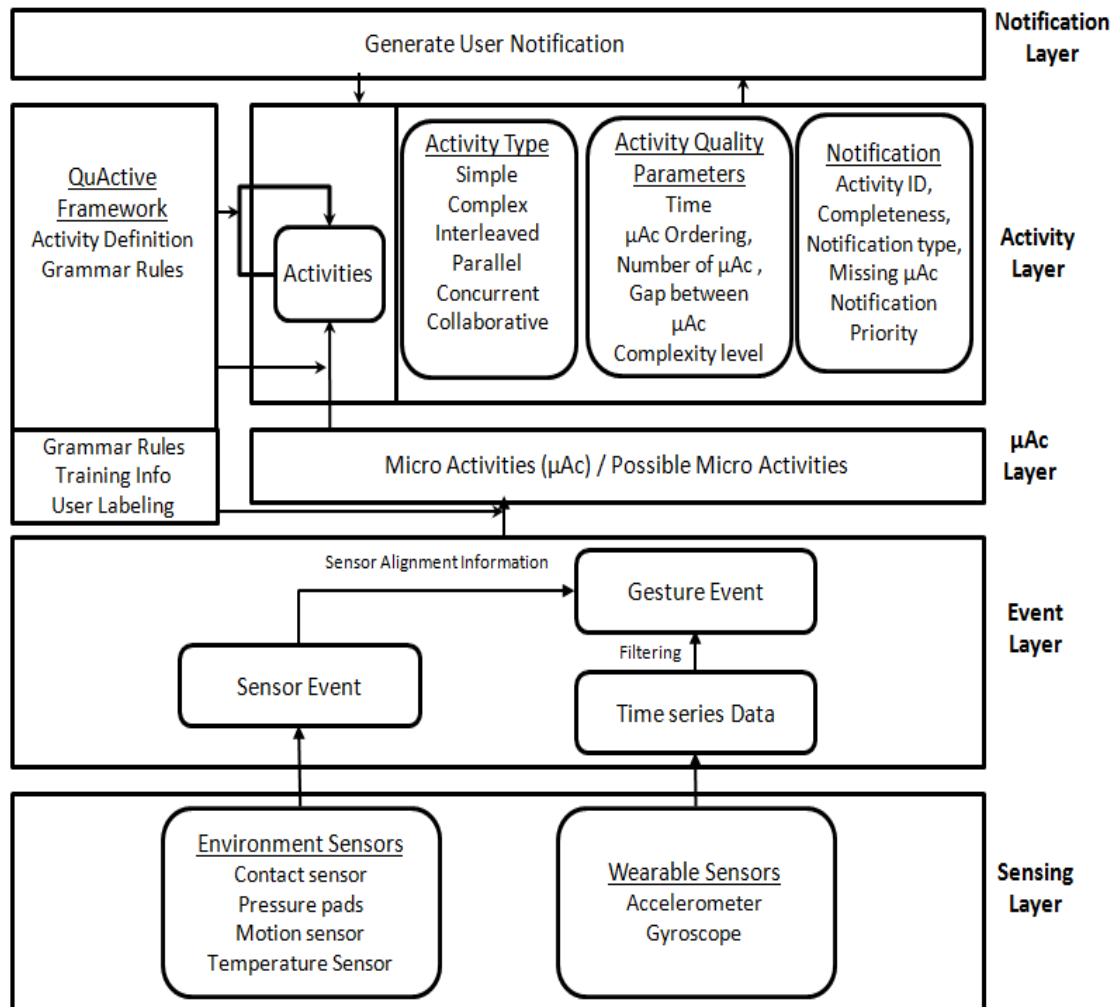


FIGURE 6.2: System Architecture for detecting and recognizing activity steps and high level activities based on QuActive Framework.

of the activity step - a user cannot leave in the middle of performing a step. Therefore, an activity step is either complete or not done at all. Any incomplete information is ignored. To detect activities from sensors and gesture events, the *QuActive* grammar is applied to event data. Grammar rules associating data with activity steps are created from training data and user labeling. Upon detecting problems (such as taking too long to perform a particular step), information is passed to the notification layer.

#### 6.4.4 Activity Layer

*QuActive* consists of grammar rules for each activity mapping to one or more activity steps. The rules are defined based on real-world observations as well as state-of-the-art definitions (particularly in ADL research in vision) relating activities with steps [36]. Each rule is assigned a default probability. Training is necessary to calculate the probability values from a particular real-world deployment.

The Activity Layer applies *QuActive* grammar rules to find the activities that occurred. Sometimes low-level activities are combined to a higher level activity. So, rules are applied iteratively until no new activities are observed. If an activity rule matches up to a certain level, where the steps are not part of any other activity, *QuActive* assumes that a certain activity was started, but not finished. Since the grammar preserves time information and activity step order, the activity quality parameters are extracted from these values. The activity complexity is determined based on how many iterations were performed to construct the activity. The activity and activity step timing information is used to classify the activity type whereas the quality parameters are used to create notifications. The priority of the notification is relevant to the severity of the problem in terms of safety (e.g., leaving the stove on), inconvenience (e.g., forgetting to put coffee in the coffee machine), or other issues (e.g., taking too long). It should be noted that *QuActive* distinguishes similar activity steps by correlating with different in-situ sensors triggered by the action or matching the activity steps with grammar rules relevant to high-level activities. However, if no specific in-situ sensor is triggered and two parallel activities with similar steps occur, then *QuActive* is unable to identify the exact activity step.

**QuActive Database:** The rules for mapping activity steps with each activity class are applied separately for every activity class. In this paper, this mapping has been done manually. However, if the sensor setting and activity list are similar, existing rules are applied for future activity instances. For example, during the evaluation of dataset 6.5.1, the rules created for dataset 6.5.1 were used. On the other hand, once an activity step is defined for a setting, it is not redefined for every activity. For example, in our experiment the 'Stirring' step is found in the activities 'Cooking' and 'Making Tea'. The 'Stirring' step is mapped to the stirring gesture event, whereas the gesture event detection is done by labeling the smartwatch data and applying a decision tree with five-fold cross-validation.

### 6.4.5 Feedback - A Notification Layer

The smart-watch based notification subsystem is an extension of the *MedRem* voice-based medication reminder system [14]. However, instead of only medication reminders, the system reminds users about activity problems. Moreover, *MedRem* is a stand-alone system whereas the *QuActive* notification system receives information about the activity parameters assuming that the watch has WiFi capability. In addition to providing reminders about activities, the sub-system generates notifications based on the notification ID. For example, missing activities trigger question like "Have you performed 'Activity X'?" whereas missing steps trigger question like "Have you missed 'step y' when doing 'Activity X'?" where the 'y' and 'X' are replaced with the corresponding parameters received from the Activity Layer. User's answer is stored and then sent back to the Activity Layer for further tracking. Therefore, the *QuActive* system intervenes with the user through informed notification.

## 6.5 Evaluation and Results

### 6.5.1 Datasets

We used the following datasets to evaluate the performance of *QuActive*. All these datasets have motion sensors, contact sensors, item sensors, temperature sensors, and water sensors installed in a smart home apartment.

#### Interleaved ADL

Dataset [72] contains activity information from 20 users. Each participant first performs the activities 1 to 8 defined in Table 6.3 independently, and then multiple activities concurrently. Therefore, the steps of multiple activities are intertwined. The details of the steps are provided to the participants. Table 6.2 shows an example of given instructions for the activity ‘Preparing soup’.

#### Multiresident ADL

Dataset [73] has information about parallel and co-operative activities. Here, each individual performs activities 1,2,5,9,10,11,12 (Table 6.3) independently, but two persons act in parallel. Therefore, steps of activities from different users are observed at the same time. In addition, activities 13 to 16 are performed jointly (co-operative activity), where the steps are either done individually (playing checkers) or together (moving furniture).

#### Cognitive assessment activity data

In dataset [12], 65 healthy and 14 cognitively impaired people are selected for the data collection process based on initial screening and questionnaires. Then, each participant is asked to complete the activities 1 to 8 defined in table 6.3 step by step. The dataset annotates the ground truth by labeling each sensor value with corresponding activities and sub-activities. Each activity is scored by expert clinicians from 1 to 8 based on the level of completeness. Moreover, the users are also diagnosed by the clinicians as healthy, as patients with mild cognitive impairment (MCI), or as patients with dementia based on the interviews, questionnaires, and performed tasks. All this information is used as the ground truth for the analysis.

Activity Steps: Preparing Soup
1. Retrieve materials from cupboard "A"
2. Fill measuring cup with water
3. Boil water in microwave
4. Pour water into a cup of noodles
5. Retrieve pitcher of water from refrigerator
6. Pour a glass of water
7. Return pitcher of water
8. Wait for the water to simmer in the cup of noodles
9. Bring all items to dining room table

TABLE 6.2: Example of activity steps within the activity ‘Preparing soup’ instructed to be performed by a user [72], [73].

Interleaved Activity List (dataset 6.5.1 and 6.5.1)	
1.	Sweeping the kitchen and dusting the living room.
2.	Obtaining medicine containers and a weekly medicine dispenser, filling the dispenser according to the directions.
3.	Writing a birthday card, enclosing a check and writing the address on an envelope.
4.	Finding the appropriate DVD and watching the corresponding news clip.
5.	Obtaining a watering can and watering all plants in the living space.
6.	Answering the phone and responding to questions.
7.	Preparing a cup of soup using the microwave.
8.	Picking a complete outfit for an interview from a selection of clothing.
Multiresident Independent, Parallel, and Co-operative ADLs (dataset 6.5.1)	
	1,2, and 5 (from the above list)
9.	Reading magazine in living room sofa
10.	Preparing dinner
11.	Setting dining table
12.	Hanging up clothes in closet
13.	Moving furniture
14.	Playing checkers
15.	Paying bills
16.	Gathering and packing picnic supplies
Activity List: Collected Data (dataset 6.5.2)	
17.	Study (sitting on study chair, using typing motion)
19.	Watching TV (sitting on living room sofa, occasional hand gesture for using remote)
20.	Making tea (using cabinets and refrigerator, heating water, gesture of stirring and putting items)
21.	Eating (sitting on dining chair, hand gesture of eating)
22.	Washing dishes (using tap, scrubbing dishes, rinsing dishes)
23.	Cooking (using cabinets, refrigerator, microwave, oven, and hand gesture for cutting, stirring)
24.	Dressing (choosing outfit from closet, motion for changing clothes)

TABLE 6.3: Activity List used for evaluation in different datasets and the collected data.

## 6.5.2 Data Collection

To evaluate the performance of *QuActive* in a real home setting, we have deployed the system in a real home and collected data from four users. All the users were healthy young adults from both gender groups (males and females). We used z-wave pressure pads, contact sensors, and

motion sensors for collecting environmental information. The pressure pads were placed on the living room sofa, dining room chairs, and study chairs. The contact sensors were attached to the cabinets, microwave, oven, refrigerator, freezer, and closets. The data was collected in a semi-controlled setting, and the users were instructed to perform the activities 17-24 (table 6.3). The experiment was not entirely controlled since we did not specify the exact steps to complete the activities or constrain their movement in any way. Users were free to perform the activities in their own way and were not required to start/stop each activity step from a resting position. Before each data collection session, the user wore an Android smartwatch on his/her dominant hand. The accelerometer and the gyroscope sensors (sampling rate 50Hz) in the watch were used to capture the hand gestures relevant to activity steps (chapter ??). Since each watch has a specific ID, it can be used to identify users in multi-user scenarios. However, in this experiment, multiple users did not perform activities at the same time. Therefore, the collected dataset has independent and interleaved activities from a single user setting. Table 6.3 shows some example steps observed during the activity process.

Another assumption made in this paper is that an activity step itself cannot be discontinuous, i.e., a step is either done or not. However, the activity steps within an activity can be discontinuous. The framework assumes that each ‘TmDiff’ parameter has a defined limit. *QuActive* assumes that the activity is not completed if the time is exceeded. The parameter value needs to be defined from training data from long term deployments. However, we did not have enough data necessary to define the parameter. We plan to accomplish this in the future. To collect the ground truth data, an observer video recorded the session, and the time series sensor data were annotated from the video using ‘Chronoviz’ software [74].

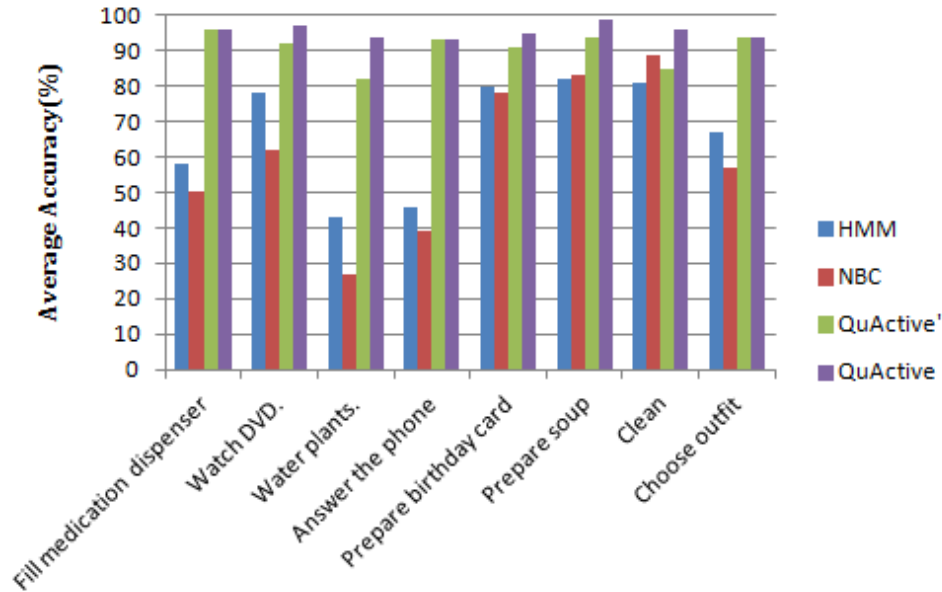


FIGURE 6.3: Recognizing interleaved (high level) activities using HMM, NBC [72], QuActive with and without (QuActive') location information incorporated in the grammar. The figure shows the percent of sensor data labeled correctly with respect to ground truth labeling.

### 6.5.3 Results and Discussions

#### Evaluation on the Datasets

Figure 6.3 shows a comparison of *QuActive* with NB and HMM classifiers [72] applied to the interleaved ADL dataset. Here, every fourth bar shows the performance of *QuActive* when the location information (using the location of the sensors) is incorporated as part of the grammar, and the third bars in each activity bar set shows when the location information is disregarded (*QuActive*). In paper [72], the authors show that Naive Bayes and HMM classifiers achieve an average accuracy of 66.08% and 71.01% respectively in detecting activities performed in an interleaved manner. The grammar in *QuActive* is constructed only with the activity steps defined from instruction disregarding the user’s location. Although location is not needed to detect most activities, it clarifies when the context is essential. For example, ‘Watering Plants’ and ‘Sweeping Living Room’ use different equipment, but the same closet sensor is triggered while retrieving/storing the equipment. Thus, adding the location information (‘kitchen to living’ or ‘in living’) in between the closet sensor trigger provides context about the prospective use of the equipment. With location information incorporated in the grammar, almost all the activities are detected perfectly. However, the accuracy in the graph shows less than 100%, since the exact start time and end time of the steps are not always aligned with the ground truth data. Therefore, there is less accuracy in terms of the percentage of sensor values labeled correctly with the corresponding activity.

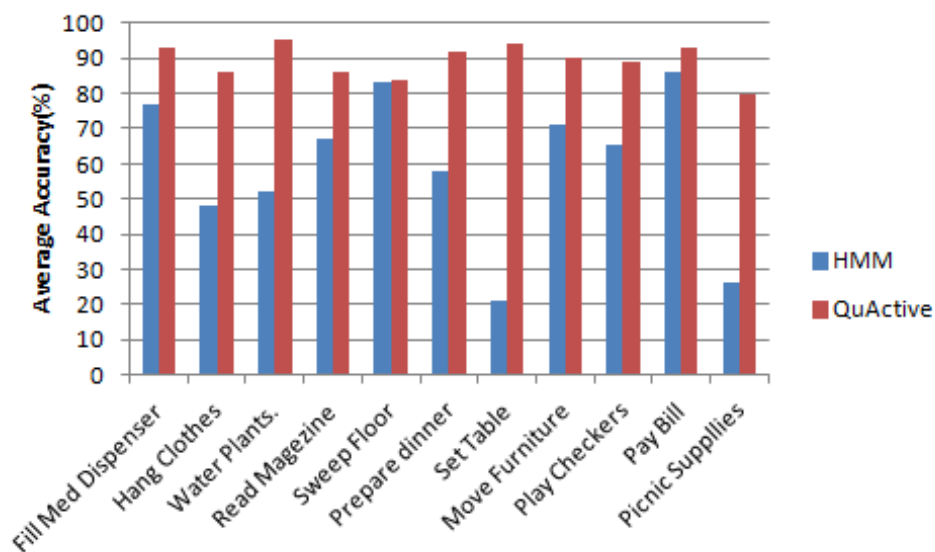


FIGURE 6.4: Average accuracy in recognizing instances of independent, parallel, and joint activities using HMM [73] and *QuActive* on a multiresident dataset.

Figure 6.4 shows the performance of *QuActive* in multi-person setting, where some activities are performed jointly, and some are done independently but in parallel. Here, the baseline is the average performance of the user-independent HMM classifier. Authors in the paper [73] show that a user-specific classifier increases the accuracy of activity detection for that user by about 20% on average, but the performance of the system decreases in detecting the activities of the other user. Thus, the average performances of user-independent and user-specific HMM models are almost the same. On the other hand, *QuActive* performs very well in this dataset,



because it filters irrelevant activity steps that do not match the grammar structure. In other words, unless the activity steps in other users' activity match the activity steps of a particular user, they do not affect the activity detection process of that user and thus yields higher accuracy.

In paper [12], the researchers show the activities of daily living (ADL) as a good predictor of early detection of cognitive impairment. They extract 38 features from the sensor dataset and show that using leave-one-out cross-validation accuracy of 86% is achieved in predicting the cognitive state. However, their process requires a lot of data from cognitive impaired and healthy persons for feature generation and training. Although we do not apply *QuActive* to determine the cognitive status of a person, Figure 6.5 shows how the value of the quality parameters defined in *QuActive* varies in healthy, MCI, and dementia patients in this dataset.

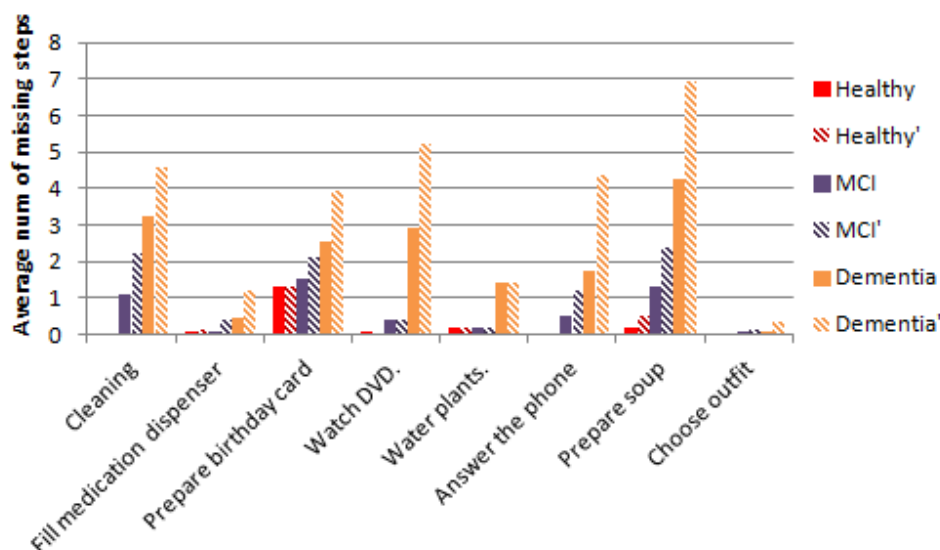


FIGURE 6.5: The average number of missing steps in performing activities by healthy, mildly cognitively impaired, and dementia patients. The striped column values considers the effect of missing activities in calculating missing steps and the solid column values disregards missing activities.

The solid columns in Figure 6.5 show the average number of missing steps per activity from the performed activity instances. The striped column shows the average number of missing steps considering the total activities, i.e., the effect of a missing activity is also considered. Similar differences are observed considering the activity duration and total duration between two consecutive activities. The figure shows the validity of considering missing steps and missing activities in identifying stages of dementia with a general purpose activity recognition framework. Thus, a grammar defined from the descriptions, that does not need huge training data and complex algorithms can also be a powerful tool in detecting how activity quality degrades over time.

### Evaluation on Experimental Data

We collected a total of 67 activity instances of activities 17 to 24 (Table 6.3) from four users in our experiments. Although only one user performed an activity at a time, she/he occasionally performed more than one activity in parallel. The experiment settings have both z-wave

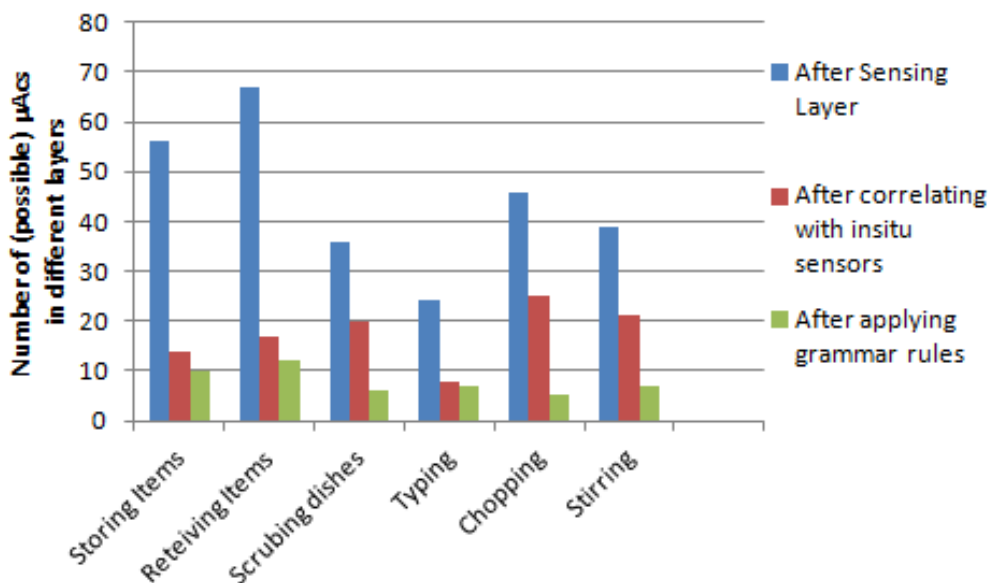


FIGURE 6.6: Filtering the falsely recognized activity steps in subsequent layers of the system.

sensors and smartwatch data for gesture recognition. *QuActive* recognizes all the steps that are defined only in terms of z-wave sensors, such as ‘Sitting on the sofa’, ‘Opening the refrigerator’ etc. However, in our experiment, only the in situ sensors do not give all information required for detecting all the activity steps.

For example, ‘Opening cabinet’ and ‘Closing cabinet’ are detected from the contact sensors, but differentiating between ‘storing item in the cabinet’ or ‘retrieving an item from the cabinet’ requires additional information which can be extracted from the hand motion. However, detecting activity steps from a continuous stream of sensor data itself is a challenging problem and the accuracy depends on the collected data and the threshold values for determining the cut-off point. In our experiments for detecting activity steps, we choose a lower threshold value to get a high percentage of true positives despite having a high false negative rate and therefore a lower recall. For evaluation purposes, we use a state-of-the-art supervised algorithm (Decision Tree C4.5) for gesture recognition and five-fold cross-validation irrespective of the user. However, coupling the gesture events with the in situ sensors filters a lot of the falsely recognized gestures. If the same gesture signal indicates more than one possible activity steps, the one matching with the defined grammar is recognized, and the rest are eliminated.

Figure 6.6 shows examples of a number of possible gesture events recognized from raw signal and how the number of irrelevant gestures are filtered at different stages, i.e., after associating with in situ sensors and finally mapping with grammar rules. Figure ?? (Appendix A) shows time series data corresponding to some example gesture events.

Table 6.4 shows the precision and recall of activity instances recognized correctly despite each user performing the activities in their own way. The accuracies of detecting high-level activities are 91% to 98%, which shows the promise of the *QuActive* system.

Activity	Precision	Recall
Making Tea	0.95	0.88
Washing Dishes	0.96	0.87
Cooking	0.91	0.84
Eating	0.93	0.95
Dressing	0.97	0.96
Study	0.98	0.99
Watching TV	0.98	0.98

TABLE 6.4: Average performance of QuActive in recognizing activity instances from all users

### Notification Subsystem

The notification subsystem has been implemented and evaluated separately from the activity recognition subsystem. In the implementation, once a notification is required, the notification subsystem delivers it to a smartwatch 100% of the time and properly records the user responses. The responses vary depending on the notification type. For example, if the notification is a reminder to add coffee to the coffee maker the user might respond, "OK done." Or if the notification suggests that they forgot to take their noon medication, the user might respond "I'll do it later." However, a user study that shows how effective the notifications are in actually improving health or performance of daily activities is beyond the scope of this thesis.

## 6.6 Conclusions

*QuActive* is a CPS monitoring and notification system for activities of daily living (ADL), developed based on a temporal, probabilistic, context-free grammar. It addresses the complexities of concurrent and parallel activities, and multiple person situations. It identifies activities irrespective of the activity errors, i.e., missing steps, delayed steps, and out of order steps in activities. Using several datasets, the performance of *QuActive* is shown to be (average accuracy of 95%) significantly above than the two baselines (accuracy of 66% and 71% respectively) from the literature. Another interesting observation is that despite having very low accuracy of gesture recognition events, by incorporating data from a number of in-situ sensors gesture events classification can be improved. Moreover, by applying structured grammar rule, high accuracy of activity recognition can be achieved by discarding misclassified activity steps.



## Chapter 7

# Activity Quality Monitoring

In this chapter, we address the problem of in-home activity quality monitoring focusing on a particular application, detecting the discrepancies observed in ADL/IADL performed by dementia patients. We provide a detailed definition of activity quality in terms of our objective and explain our solution and implementation details for handling the problem.

The rest of the chapter is organized as follows. We briefly describe the motivation of the work in section 7.1. We describe the scope of our work in section 7.3, followed by our solution and implementation details in section 7.4 and 7.5 respectively. Finally, the chapter concludes by providing the details of evaluation and discussion ( section 7.6).

### 7.1 Motivation

In 2018, an estimated 5.7 million Americans of all ages were living with Alzheimer’s dementia, which is the 6th leading cause of death in the United States [75]. Despite being very common, Alzheimer is tough to diagnose since it is a slowly progressing disease. However, the recent advancement in technology has enabled the research community from different domains to work on solutions which can at least detect the symptoms as early as possible and thereby slow down the progression of this terrible illness. One of the common detectable outcomes of this illness is the inability to complete familiar daily tasks. The behavioral scientists and psychologists have observed and identified several inconsistencies that Alzheimer patients exhibit while performing activities of daily livings (ADL) and instrumental activities of daily living (IADL). Due to the loss of memory and decreased judgment, these patients often forget steps while performing an activity, misplace objects, do things out of order, or repeat the same steps over and over again. Moreover, due to confusion they often take longer to complete an activity or certain activity steps which requires attention and judgment skills. Therefore, home activity recognition and health care systems which can detect the mentioned inconsistencies are extremely helpful in diagnosing the early signs of Alzheimer as well as tracking the progression of this deadly disease.

Although there are a lot of interesting works on the domain of activity recognition research not much is observed in finding the quality of an activity process. In this chapter, we specifically focus on the problem of finding missing activity steps, wrong steps, and other inconsistencies observed in Alzheimer’s patient in different scenarios (figure 7.1). In literature, the research approach of detecting and recognizing ADL and IADL are divided into two broad categories - data-driven approach and knowledge-driven approach. In the data-driven approach, an activity is recognized from wearable or in situ sensor data by using different learning algorithms. Even though data-driven approach is the standard and accepted practice, certain applications require the knowledge of expert or specialized learning algorithms to represent activity steps

in terms of knowledge base and relate the steps with specific activity rules (Context Free Grammar, AI ontology, etc.). Moreover, it requires an enormous amount of data, proper labeling, and effective training to learn from the data and develop practical specialized algorithms. Since the diagnosis of Alzheimer disease requires information about activity steps, this particular application requires knowledge driven or at least the hybrid approach for recognizing activities such that the activity steps can be clearly defined and the relationship among the steps well represented.

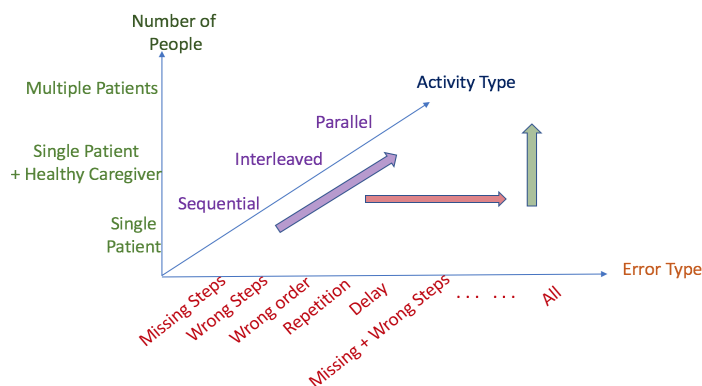


FIGURE 7.1: Complexity of solution space (identifying activity or activity attempt) with increasing difficulty of problem space (presence of different types of error)

In this chapter, we assume that the system is provided with the knowledge base and rule information about each activity. Now, if the system is given input of a stream of detected activity steps, will it be able to output which activity the user wanted to perform? Activity recognition from detected activity steps itself is very challenging due to similarities of different rules, or some of the same activity steps being found in more than one activity. However, the problem becomes even more complicated when we assume that not all activity steps are present, and/or some of the steps were done out of order or as a result of a mistake. Although, there is some work which shows straightforward ways of detecting missing steps from activities with sequential steps 3.2, none of them considers the variety of errors that can happen in real life complex scenarios or is vague about how exactly the solution will work in the presence of the combination of different types of errors (figure 7.1). The presence of different types of errors increases the complexity of detecting an activity even when the activities are done by a single person sequentially. Complexity increases if we assume that the activity steps from different activities can interleave or activities are done in parallel. In the presence of multiple users, the problem becomes even more complicated.

To address the mention challenges, in this chapter, we provide a solution for recognizing both complete and partial activities from the sequence of detected activity steps. Our solution shows how comparing a particular sequence with activity patterns defined by rules is able to find out the error that occur within the activity. We also provide details of system implementation and technique for storing and searching already observed sequences efficiently.

## 7.2 Contributions

The main contribution of this thesis chapter is

- We describe the activity quality monitoring problem for a specific application, accessing the activity quality of patients with dementia, and define parameters for clearly quantifying activity qualities.
- We provide a solution for addressing the challenge of recognizing incomplete activities due to missing/wrong step, wrong order, repetition of unnecessary steps, etc. We provide
  - Filtering technique for effectively separating activity episodes of different error types. Also, method for separating activity steps from different activities.
  - Validation technique by converting the infinite pattern matching problem from string domain to activity recognition. We adjust the solution by selecting and modifying the quantifiers to represent infinite patterns in activity recognition domain.
  - We provide effective storing different types of instances observed for the same activity and method for searching previously observed similar valid and invalid examples.
- We provided details of the implementation of our solution and describe the purpose of defining each parameters, and how the major functions work.
- We evaluated our solution in a public dataset with Cognitive Assessment Activity Data from healthy adults and patients with dementia. We also collected data from our lab and simulated the different error types to evaluate our system in different scenarios.

## 7.3 Problem, Background & Scope

### 7.3.1 Problem Description

The main focus of this paper is to detect the inconsistencies related to the activities of daily living (ADLs) and instrumental activities of daily living (IADLs) of Alzheimer's patient with dementia. The signs observed in ADL/IADL as a potential indicator of the disease can be divided into two main criteria - activity steps and delay. First, we list the anomalies related to activity steps -

1. **Missing an activity step:** The healthcare providers have marked missing step from ADL/IADL as a potential sign of illness, since they indicate lower functional (difficulty in performing an activity step) or cognitive (forgetfulness) ability. The family members of Alzheimer's patient mention patients leaving the stove on, leaving the water running, or forgetting to finish an activity, i.e., missing the steps that are at the end of an activity.
2. **Repeating steps unnecessarily:** Repetition is another common characteristics of Alzheimer patients because they forget already completing the activity steps. For example, putting ingredients (salt, pepper, sugar, etc.) again while cooking. This behavior can be hazardous in some cases. For example, taking a drug/ medication dose more than the prescribed amount that controls blood pressure or insulin level, or frequently eat despite having dietary restrictions.

One of the challenges in identifying repeated steps is that for some activities the repetition is part of the process, such as sweeping movements while cleaning the floor. Again, some steps occur for a certain number of times, but repeating more than some limit indicates an abnormality. For example, adding coffee in the cup while making coffee.

The most common form of repetition is observed when the patient is speaking since he/she repeats sentences due to forgetfulness. However, since we do not consider speaking as a part of daily in-home activities, recognizing the repetition in speech is out of the scope of this paper.

3. **Doing a step out of order:** Steps within an activity can be performed in different orders based on people's preference. However, in some cases, the order is very important. For example, sometimes patients need to measure specific physiological parameters measurements before/after having medications, or might need to have medicine in an empty/full stomach. In those situations, ordering of the steps will be necessary, where the steps may belong to same/different activities
4. **Wrong step:** Sometimes an Alzheimer patients start an activity and in middle of it start doing a different one, or performs an entirely irrelevant step within an activity. For example, putting the keys inside the refrigerator or boiling/heating the mobile phone which can be very dangerous.

Clinicians and researchers also identified an abnormality in activity duration as quality parameters related to sign of illness. Here, we consider finding which steps takes too long and whether there is a long pause between two consecutive steps.

1. **Taking too long in performing a particular step:** Since Alzheimer causes functional decline, the patients often take longer to perform activity steps. Again, due to cognitive declination, the seemingly simple task appears to be very complicated to them. For example, just merely dressing up becomes an complex activity with so many steps, i.e., opening the drawer/closet, selecting a cloth, unbuttoning the shirt/dress, making the dress inside out or outside in, putting it on, then buttoning the shirt, putting the right button in the corresponding holes, and so on. The family members of Alzheimer's patients report observing their struggle in buttoning the shirt or difficulty in tying shoelaces
2. **Delay between intermediate steps:** Delay between steps are also seen because either the patient temporarily forgets about what he/she was doing or needs time before the next step due to reduced functional ability. Again, while doing multiple activities, a conscious healthy person will switch back to the original activity and complete it before its too late, but Alzheimer's patient has difficulty in multitasking.

### 7.3.2 Activity Grammar

In order to find what activity was performed, the system needs to know what activity is or some form of information to compare or contrast the incoming input. In this paper, we use the concept of grammar to create a patterns for each activity and compare whether the stream of activity steps conforms with the defined patterns. For general languages, a 'grammar' is a set of rules that is used for proper conversation with each other. Similarly in computer science, 'grammar' is a mathematical model defining some rules for accurately writing a computer



language. According to Noam Chomsky, any grammar 'G' is formally defined as a tuple of  $\langle S; \Sigma; V; R \rangle$  where

- $S$  is the start symbol. From an activity recognition perspective, it would be the unique symbol representing the Activity class.
- $\Sigma$  is a finite set of terminal symbols. In our case, this is the set of all possible activity steps within an activity class that may occur in different variations of the activity.
- $V$  is a finite set of nonterminals. Nonterminals are used as intermediate symbols that can be broken into one or more terminal or nonterminals.
- $R$  is a finite nonempty subset of  $V_N \times V^*$  called the production rules. These rules are used to define how each of the symbols relates to each other. In the case of our application, the rules are useful in establishing the structure of activity steps within an activity class. Multiple rules are used to capture the variations of multiple ways of doing a particular activity.

The differences in different types of grammars are determined based on how constrained the rules are. For example, rules in any form of context-free grammar (CFG) ensures that the left side of the rule has only a single nonterminal symbol. Paper 3.2 shows that a time probabilistic CFG, a variation of CFG, is capable of modeling activity in terms of activity steps and recognizing the activity later when steps are detected.

However, depending on how the rules are defined in a CFG, it sometimes creates ambiguity. To avoid ambiguity a left derivative grammar or a right derivation grammar is used. Such languages are called regular languages and can be recognized by regular grammar, which is a subset of context-free grammar. CFGs are recognized using pushdown automata, and regular grammars are recognized by deterministic/nondeterministic finite state automata (dfa/nfa) or regular expressions. Any dfa can be converted to nfa and nfa can be converted to a regular expression. Therefore, there is always a regular expression that recognizes the regular language.

Regular expressions (regex) are used as a powerful tool for pattern matching in many areas, most importantly in text mining or string searching. In standard form, regular expressions have literal characters and meta characters, where each meta character has a special meaning. The metacharacters can be of different types, such as single character, quantifiers, and position characters. In order to compensate some of the limitations of the earlier versions of regular expressions, many current compilers use extended regular expressions (ERE) which have additional meta characters and allow counting of characters or specifying certain ranges. Again, many implementations allow grouping subexpressions with parentheses and recalling the value they match in the same expression (backreferences). Thereby these expressions have the power that far exceeds recognizing only the regular languages. However, which metacharacters are available in a particular environment depends on the regex engine.

In our studies, we have found that most of the everyday activities can be defined using a regular grammar. The exceptional cases which require stronger grammars mainly need keeping track of the number of steps or repetition counts. Therefore, we use extended regular expressions (ERT) to define each activity pattern and search the patterns from the input stream of activity steps. Table 7.1 shows the metacharacters (quantifiers) we used in our program and their functionality described in the string matching domain. The ERT for each activity is defined using the metacharacters and activity steps. We only used different quantifiers among the metacharacters. We built methods for supporting only the quantifiers that are useful for our purpose.

Instead of comparing literal characters we build methods for comparing activity steps. Moreover, we have included features for subgrouping and multiple alteration paths variable sizes by allowing recursion and backtracking. The following example shows how different

metacharacters are useful in capturing the variation and structure of activity steps in daily activities. In section 7.4.2, we explain how this powerful tool is used for finding the activity step related to anomalies observed in Alzheimer’s patients ADL/IADLs.

#### Usage of Metacharater in Activity Modeling

- i The Kleen star,  $*$ , captures the activities where steps are repeated without any specific count. Such as sweeping movement while cleaning the floor.
- ii The question mark,  $?$ , is used for defining optional activity steps.
- iii The count,  $\{n\}$ , is used to define activity steps that should maintain specific counts.
- iv The  $\{\min, \max\}$  is used to define activity steps that are repeated within a certain range.
- v Altercation  $[abc]$  allows us to chose a single activity step from all the given options within a square bracket, such as [‘add sugar’ ‘add honey’] while making coffee.
- vi Alternation and subgrouping  $(ab|bdef|egh)$  allow taking alternative pathways when the same activities are performed in different ways.

Since extended regular expression (ERE) are powerful in expressing and generating languages, we use this tool to create ERE corresponding to activities from a dataset description of activities, activity steps, sensor settings, and building layouts during our evaluation. An excellent future study can be learning the ERE from observed training examples, which requires a considerable amount of data to be general enough for handling all cases.

### 7.3.3 System Inputs and Outputs

#### Knowledge base: What does the system know?

The system knows:

1. The set of activities of daily living (ADLs) and instrumental activities of daily living (IADLs) that occur throughout the day in a particular setting.
2. Grammar or Extended Regular Expression (ERE) associated with each activity.
3. What activity steps occur in a particular activity class, i.e., the terminal set  $\Sigma_i$  of each activity class.

	Quantifiers	Functionality
Kleen star	$*$	Matches the preceding element zero or more times.
At least once	$+$	Matches the preceding element one or more times.
Optional	$?$	Matches the preceding element zero or one times.
Count	$\{n\}$	Matches the preceding n number of times.
Range	$\{\min, \max\}$	Matches the preceding atleast ‘min’ number of times and atmost ‘max’ number of times.
Altercation	$[abc]$	Matches either one of the characters with in the square brackets.
Subgrouping	$(ab bcde efg)$	Matches either one of the paths separated by $ $ with in the parenthesis.

TABLE 7.1: Functionality of meta characters in extended regular expressions.

## Inputs

The input of the system is a stream of activity steps detected over the day. There are properties associated with each of the identified steps, such as start time, duration, time gap from the previous step, and the sensor events or gesture from which the steps was identified. The actual process of steps detection is out of the scope of this paper. The lower level system/subsystem might have used any sensor setting (in situ sensors, wearables, or hybrid setting with multi-modal sensors) or any algorithmic or machine learning approach. The system will take the inputs from the activity step detection system/subsystem as facts and start processing from there. For example, if a step is missing, the system will not concern itself whether there was a detection error or not and report it as a missing step.

## Preprocessing Input: Activity Episodes

The input stream of activity steps recognized by lower-level system component is divided into smaller chunks based on the timing parameter, i.e., the time difference between two consecutive activity steps. We name these sequences as activity episodes 3.2.

**What timing threshold to use?:** Ideally, we would like to have a threshold such that all the delays between two consecutive steps of the same activity would be less than the threshold, whereas the delay between the end time of an activity and the start time of the next activity would be higher than the threshold. But in reality that is impossible. Some of the difficulties are:

1. The distribution of time difference between the steps within a particular activities vary widely depending on the activity type, person, or situation.
2. The delay between two consecutive activities are also not consistent and can be very short, especially at particular times of days. For example, a person's morning routine might be using the washroom, doing exercise, taking a shower, preparing a meal, having breakfast, washing dishes, getting dressed for work, etc. within an hour or two.
3. Depending on the available data, the distribution of the activity and activity steps can be very different. Thus, no fixed time threshold works on all activities.

In this paper we consider the optimal time threshold considered by paper [76], which ranges with in 2 to 3 minutes. Depending on the dataset, the exact value might shift a bit.

## Outputs

The system answers of the following questions-

1. What activity was performed, or attempted to perform?
2. What was the performed sequence?
3. Was it performed correctly?
4. What errors occurred?
5. What was the closet correct sequence?
6. Was there any timing anomaly?

These outputs are then stored and any change in parameters and statistics are updated after getting the result.

### 7.3.4 Number of people and activities

#### Number of Users

The system is designed mainly for one user (patient), but there can be other people in the environment. Hence, we tested our system with datasets of a single person who have issues completing ADL/IADL correctly. However, we simulated incorrect activities in the dataset having multiple people but assume that only one person's ADL/IADL has anomalies.

#### Activity Types and Occurrence Boundary

Depending on how closely in time the activities are performed, the following types of activity are observed in a single episode.

##### 1. Single Activity:

This is the ideal case where each activity episode has one activity instance. We proceed analysis assuming a single activity instance and check whether that activity has occurred without any errors, or it has missing steps, wrong steps, etc.

##### 2. Multiple Activities:

There can be more than one activity within a particular activity episode.

*Sequential Activities:* When people perform activities immediately after finishing the previous one, the two activities do not have enough of a gap and are found sequentially in the same episode.

*Parallel Activities:* When users perform more than one activity in parallel, i.e., steps of other activities in between the current activity, the steps of both activity instances are grouped together within a single episode.

Irrespective of the number of people in the systems, we assume there is no more than two activities at a time.

## 7.4 Solution

In this section, we explain how we solve the problem of recognizing activity and evaluating the quality of an activity instance. Figure 7.2 shows the overall process of our approach. Before entering the algorithm, the input streams of activity steps are divided into smaller chunks of manageable sequences called activity episodes (section 7.3.3). The algorithm takes each activity episode,  $E$ , as input and regenerates the output parameters of  $E$ . The whole process can be divided into three main phases:

- i Selection Phase: We perform some filtering and selection to determine whether there is a chance of instance of a particular ADL/IADL occurring within the episode and make necessary modifications of the input sequence if needed (box 1c - 1h).
- ii Decision and Validation Phase: In the next phase, a decision is made whether a particular instance of activity class  $A_i$  is found and performed correctly or not (box 2a - 2b).
- iii Result Adjustment and Storing Phase: Finally, the algorithm traces back the result to the original episode and merges information if necessary, checks for a time-related anomaly, and stores the result (box 3a - 3e).

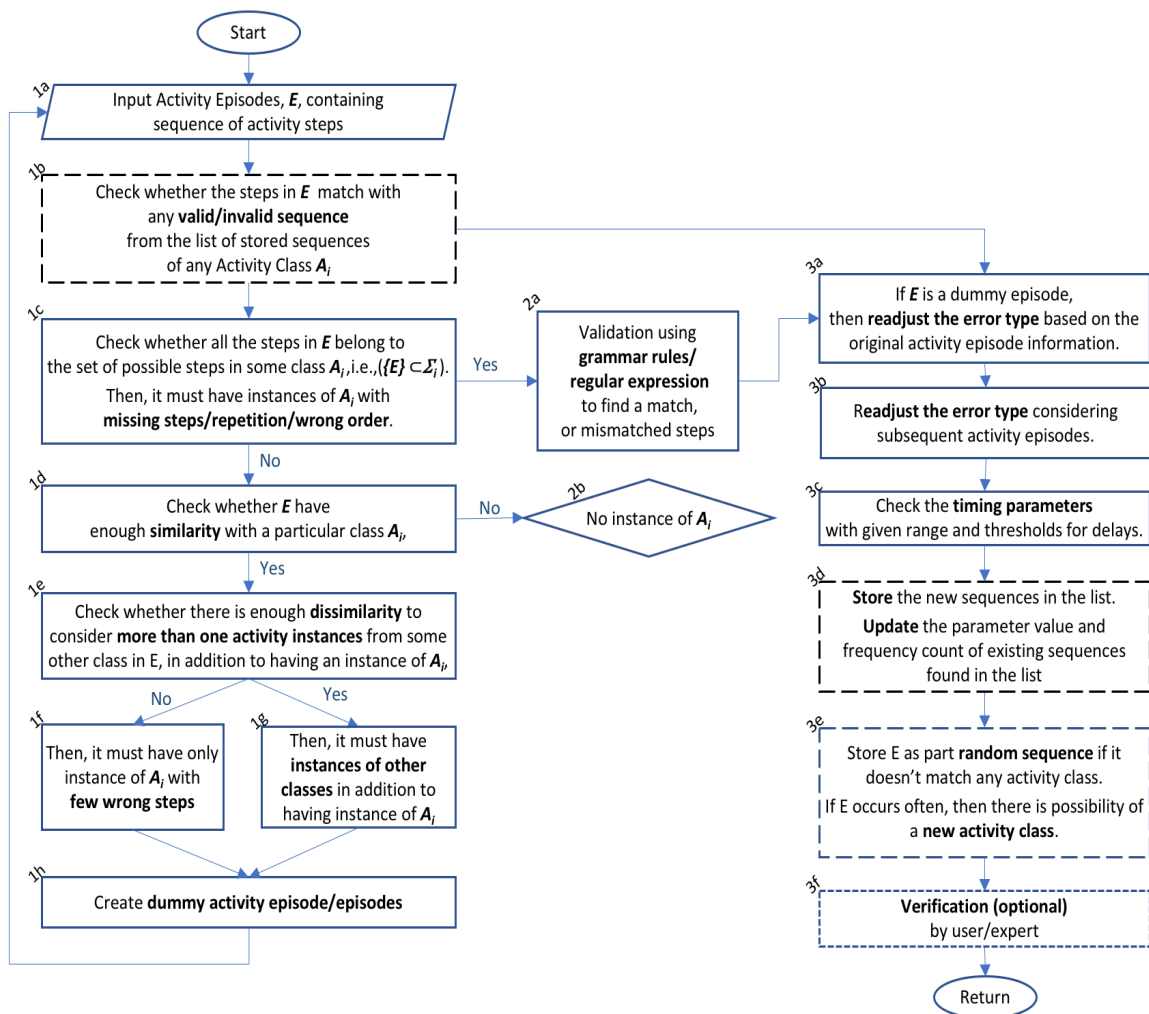


FIGURE 7.2: The overall process of recognizing activity instances and finding the quality parameters.

### 7.4.1 Selection Phase

#### Searching

The straightforward way of deciding whether a particular activity occurred in episode  $E$  is comparing with the already known activity sequences of that class. Therefore, in the first step (box 1b), our algorithm quickly performs a search in the stored list of activity instances to find the exact match. If no such match is found, then it means this particular sequence has never been observed. Initially, nothing is stored, and therefore no match will be found. As the system keeps on running, the same sequences are observed again. Thus, a lot of the processing steps can be skipped (goes directly to step 3a).

### Filtering and Selecting

If a particular ADL/IADL of class  $A_i$  occurs in activity segment  $E$ , then all the activity steps observed in  $E$  will belong to the set of possible steps found in class  $A_i$ . Thus, the condition  $\{E\} \subset \Sigma_i$  must be true (box 1c); here

- $\Sigma_i$  is the set of terminal symbols of the grammar defining activity class  $A_i$
- $\{E\}$  represents the set of activity steps in  $E$  without considering the order or count.

Although, the condition holds true for correctly performed activity instances, satisfying the condition doesn't always mean that the activity was done perfectly. The instances of  $A_i$  that have specific error types (missing steps/ repeated steps/ wrong order of steps) also satisfy the condition. Therefore, further validation is required (sends to step 2a). Here, note that even if  $E$  denotes some activity instance performed correctly, it still doesn't have to be equal to the nonterminal set  $\Sigma_i$ , because not all steps are mandatory in a valid activity instance.

However, an instance of  $A_i$  still might have occurred in  $E$  when the condition  $\{E\} \subset \Sigma_i$  isn't satisfied, due to the following reasons:

1. The user mistakenly performs a step from another activity (wrong step).
2. Activities from other classes are also being performed parallelly in addition to the activity instance of  $A_i$ .

Whichever of the above cases might have occurred; there should be many steps in  $E$  which belongs to class  $A_i$ . That is why in the next step (box 1d), we check the degree of similarity between the activity class,  $A_i$ , with the activity episode,  $E$ , using a similarity match function (section 4.3). If not enough similarity is found, the algorithm decides that no instances of class  $A_i$  are in  $E$  (box 2b). Now, if  $E$  has a very high similarity match, then which of the two cases (wrong step vs. multiple activities) mentioned above occurred?

If there are very few extra steps in episode  $E$  that does not belong to class  $A_i$ , then those few steps were likely just wrong ones since there will not be enough steps for another activity. In contrast, too many extra steps in episode  $E$  indicate the existence of multiple activities.

### Separating steps of Different Activities

The validation method assumes that each activity episode contains steps from a single activity instance. Therefore, before passing to that phase, we need to separate the steps of multiple activities or remove the extra wrong step/steps. This way we create dummy episodes that satisfy the requirement for entering the validation method. These dummy episodes are then passed to the algorithm for reevaluation, and the results are traced back to the original episode later.

#### Single Activity with Wrong Steps

In order to create an activity sequence where all steps belong to class  $A_i$ , we remove the wrong steps from  $E$  using the operation

$$E' = E - (E - \Sigma_i)$$

This deduced episode  $E'$  is called a dummy episode. Since the dummy episode satisfies the condition  $\{E\} \subset \Sigma_i$ , it is fed back to the start of the algorithm.

#### Separating steps of Multiple Activities

Suppose, there is activity instance of class  $A_j$  in addition to class  $A_i$ . Now, while creating dummy episodes we have to base it on the similarity between the two class.

***A<sub>i</sub> and A<sub>j</sub> are disjoint***

That means there is no common steps between these two classes. Therefore, the steps can be easily separated by creating the following dummy episodes:

$$E_i = E - (\{E\} - \Sigma_i)$$

$$E_j = E - \Sigma_i$$

Here,  $(\{E\} - \Sigma_i)$  is the set of steps in  $E$  that does not belong to  $A_i$ .

***A<sub>i</sub> and A<sub>j</sub> have common steps***

Now, depending on where the common steps belong there can be a few possibilities. Therefore, we create dummy episodes both with and without the common steps:

$$E_{i1} = E - \Sigma_j$$

$$E_{i2} = E_{i1} + (\Sigma_i \cap \Sigma_j)$$

$$E_{j1} = E - \Sigma_i$$

$$E_{j2} = E_{j1} + (\Sigma_i \cap \Sigma_j)$$

The algorithm feeds back all four of the dummy episodes for further processing, and any error is readjusted later (see section 7.4.3).

**7.4.2 Decision and Validation Phase**

The validation method takes activity episode,  $E$  and the grammar of activity class,  $A_i$  as input (box 2a). It assumes there is a single instance of  $A_i$  occurring in  $E$ . The method checks whether the step sequence preserves the structure of the activity class or not. In case of a mismatch, the method also finds what errors occurred based on the grammar definition. The basic idea is to modify the regular expression matching algorithms of strings that are similar to String edit distance. We use dynamic programming to store the intermediate results.

- Whenever literal characters (in this case activity steps) are found in expression, it is matched with the current step from the activity sequence.
- For quantifiers '?', '\*', or '+' there is a choice of continuing the matching with latest literal or skipping and proceeding to the next character.
- Quantifiers defining count and range require matching the latest literal character some the specified number of times.
- An alteration with single alternative steps is also handled by checking whether any of the steps specified within the square bracket have occurred.
- However, if alternations and subgroupings are encountered, then it is necessary to check the different alternative paths. Therefore, we use recursion to save the latest state of matching and explore a specific path. If the path ends up leading to a perfect match, the algorithm exits. Otherwise, our method backtracks to the latest saved state and chooses any available path which hasn't been explored yet.

The detailed algorithm is given in section 4.3.

If  $E$  matches with an activity  $A_i$  without any error, then no other classes are considered for a match and after completing the third phase of the process the algorithm exits. However, if the instance in  $E$  occurs with error, then the whole process is repeated, and other classes are checked for a perfect match.

Similarly, when the algorithm decides that a particular instance of class  $A_i$  does not belong in episode  $E$  (box 2b), it checks for the next class and repeats the same process. When no viable class is found, the episode is listed as a random sequence (box 3e)

### 7.4.3 Result Adjustment and Storing Phase

#### Error Adjustment

After activity detection is completed, the third phase of the process begins. If dummy episodes are created to separate steps of different activity instances, then adjustments are made by considering the information from the original activity episode and how the division was done.

**Single Activity with Wrong Steps** Now, if we think carefully, we see that there can be two types of wrong steps.

1. **Extra Step:** If  $E'$  matches a valid listed sequence perfectly, then the wrong steps within  $E$  are extra steps.
2. **Substitute Step:** If there are some steps missing from  $E'$ , then the wrong step within  $E$  are most likely done instead of the missing step.

Therefore, based on the output of errors found for  $E'$ , the error in  $E$  is readjusted.

#### Episodes with Multiple Activities

After the processing of the dummy episodes, the results in  $E$  is readjusted. Careful checking is required when both classes within the episode have common steps so that the same step is not reported as part of both activities. We also check whether the activity instances can be combined as the original activity episode  $E$  (i.e., which combination of the dummy episodes from each activity class occurred).

#### Combining Instances Together

Additional error adjustment is performed by examining subsequent episodes up to a specific time window. For example, checking is performed to see whether there is one activity instance with missing steps at the end, and another instance from the same activity class with missing steps at the beginning. Then, if those two instances are done within a certain time threshold, then the system combines those activity instances as one with no missing step but having a significant delay between steps.

#### Timing information for Quality

After all the discrepancies related to activity steps are sorted out, the algorithm proceeds in finding time-related abnormalities. These abnormalities are detected by comparing the timing parameters (start time and duration) of each activity step. There can be two types of delay:

1. **Delay of a particular step:** If the user takes longer than usual performing a particular activity steps, then it is considered a delay.
2. **Delay between steps:** While determining this type of delay, the system needs to consider whether the user was performing steps from some other activity or not.

In order to find the delay, the system has to provide with a range within which the duration should fall. Otherwise, initially, no delay related error is reported. Later, when enough examples are observed, the delay is determined by comparing with the history of data from which a duration distribution is calculated.

After the results are finalized, the newly observed sequences are stored. If a sequence already exists in the storage, then the statistic of the duration of the activity step and delay between consecutive steps are updated.



### Storage for Efficiency and Statistics

Once an activity sequence has been recognized, there is no need for going through the same process again. There can be up to so many variations, and often patients who perform activities with errors, tend to make the same mistakes. Therefore, storing the result provides a faster evaluation. To search the result efficiently, we use a dictionary.

#### 7.4.4 Verification by Expert

It is not always possible to receive user feedback, and therefore the verification part is optional in our system. However, if at least periodically (once in several weeks) an expert opinion can be received for the unseen sequences, then it would be extremely valuable. When generating expressions using a grammar, syntax analysis and semantic analysis both are required. For example, "I play chocolate" give a valid sentence structure but illogical meaning. Similarly, even if the grammar shows the activity sequence as valid, it may or may not be practical. Especially, since a lot of activities are user dependent, and a person close to a patient will be able to identify problems that others can not tell.

## 7.5 Implementation Details

In this section, we describe the implementation details of some of the major classes and important functions. The implementation was done using Netbeans IDE 8.2 with JDK 1.8.

### 7.5.1 Entity Classes

The goal of this system to recognize in-home ADL/IADLs. However, we have to differentiate the different instances of the same activity class. For example, each time a person eats, a new activity instance should be recognized. One way of implementing it would be creating a class for each high-level activity, and create a new object belonging to the class. For example, we can hard code classes named as 'Making Coffee', and create an object each time the system recognizes making coffee. However, that would make the system too constrained since different homes may have different activities and due to a difference in sensor setting the steps of the activity will also be different. It is not possible to know and create classes for every ADL/IADL in the world. Therefore, we separate the high-level activity class definition from the class that generates activity instances. The system has two different classes 'ActivityGrammar' and 'Activity'. If a dataset/home setting is designed to recognized 'N' different high-level activities, then the system creates 'N' objects of the 'ActivityGrammar' class corresponding to those high-level activity classes. On the other hand, any recognized instance is represented by an object of the 'Activity' class. However, each object of the 'Activity' class has a variable associated with the 'ActivityGrammar' class. Therefore, our system is general in terms of the underlying sensing environment and works irrespective of the ADL/IADL defined for a specific home setting.

**ActivityGrammar Class** The *ActivityGrammar* class defines the meta information of each high-level activity. One object of this class is created for a given list of ADL/IADL in a particular dataset/home setting during the initialization phase of the system. Each object has a unique id and corresponding class name. Variables symbols, terminals, rules, and probability are used to define the corresponding grammar of the class. The variable *regularExpression*

defines the same regular expression of the activity class, which is either provided directly instead of the grammar or is generated from the grammar rules. The time information variables *durationRange* defines the acceptable duration length of the total activity, whereas *delayTimeout* defines the timing threshold between two steps within this activity class.

**Activity Class** Whenever an activity instance is recognized, a new object of the 'Activity' class is created. The created object is associated with an object from the 'ActivityGrammar' class whose regular expression matched or is the closest match with the provided list of activity step sequence belonging to the 'Activity' class object. The *startTime* and *duration* variable stores instance specific timing information, i.e., when the activity was performed and how long did it take to finish. The *stepCount* is the length of the activity instance or the number of activity steps within the activity instance. The *errorCode* is used to indicate the error observed in that specific activity instance (table 3.2). If no error is found, the *errorCode* is zero, and a single digit code is assigned for any specific error type. If more than one type of error is found, then the codes are appended. The *deviationNumber* is the minimum number of steps required to insert/delete/change to convert an invalid activity step sequence to a valid one. If no error is observed, then the variable 'isNormal' is set to true. The 'activityOccurrenceIndex' is used for storing information of activity type, i.e., whether the activity was interleaved in between other activity instances, performed in parallel with one or more activities, or occurred by itself.

Error Codes	
No Error	0
Missing Step	1
Repetition	2
Wrong Step	3
Wrong Order	4
Deley within a Step	5
Deley between Steps	6

TABLE 7.2

**ActivityStep Class** 'ActivityStep' is another entity class similar to the 'Activity' class. Each object of 'ActivityStep' is a step detected by the underlying system components. Each object of this class also has time-related variables storing the start time and the duration of the step. The class related statistics of activity steps are stored within the *terminal* set of the *ActivityGrammar* object. Each time an activity step instance is encountered, the 'updateStatistics' function updates the relevant timing information of the 'ActivityStep' objects of the terminal set of corresponding high-level activity. Therefore, if the same activity step occurs in different high-level activities, the related statistics would be different.

**ActivityEpisode Class** The input stream of activity steps is divided into chunks of steps sequence, each chunk a named activity episode. The object of 'ActivityEpisode' represents each such chunk. The 'startTime' of an episode is the minimum 'startTime' among all the activity steps within the episode. The *duration* is found by subtracting the *startTime* from the maximum *endTime* as calculated from the sequence of activity steps. The 'length' indicates the number of activity steps in the given chunk of a sequence. Initially, the variable 'assigned' is false. After the episode is passed through the main algorithm and an outcome, i.e., the possible activity class, is found, then the variable is set to true. However, the main algorithm checks for other possible matches unless a match is found without any error. In that case, the variable *isNormal* will be set to true, and no further analysis will be performed on that specific activity episode. All the possible matches are listed under the outcome object belonging to the episode.

```

public class Activity{
    ActivityGrammar associatedClass;
    String name;
    int id;

    int startTime;
    int duration;
    int stepCount;

    boolean isNormal;
    int activityOccuranceIndex;
    int errorCode;
    int deviationNumber;
    List<ActivityStep> sequence;
    Set<ActivityStep> set;
}

public class ActivityStep {
    public int stepId;
    public String name;
    public int startTime;
    public int duration;
    List<Event> events;
}

```

(A) Instance classes

```

public class ActivityGrammar{
    public int id;
    public String name;

    //variables defining the grammar
    public Set<String> symbols;
    public Set<ActivityStep> terminals;
    public List<Rule> rules;
    public Map<Rule,Float> probability;
    public String[] regularExpression;

    //time-related properties
    public Pair<Integer,Integer> durationRange;
    public int delayTimeout;
}

class Rule{
    String LeftSide;
    List<String> RightSide;
}

```

(B) Metadata class

```

class Outcome{
    List<Activity> possibleActivities;
}

public class ActivityEpisode {
    public List<ActivityStep> sequence;
    public Set<ActivityStep> stepSet;
    int startTime;
    int duration;
    int length;
    boolean assigned;

    Outcome outcome;
    boolean isNormal;
    boolean hasSingle;
}

```

(C) Input Output classes

FIGURE 7.3: Definition of different entity classes

## Utility Classes

There are other utility classes for reading the grammar/regular expression and for reading the input. The input file may contain only the raw sensor value. In that case, the reader transfers the data to the activity steps recognizer which generates a sequence of activity steps. The *EpisodeCreator* class contains the functions for iterating through the continuous stream of activity steps and generating individual activity episodes (object of the *ActivityEpisode* class). These episode objects are considered as the unit of processing.

### 7.5.2 Major Functions

Since Java is an object oriented language, each functionality is wrapped around some class. The code system functionality is wrapped around a singleton class object of 'ActivityMonitor'. The 'ActivityMonitor' class has all the functions described in the solution section (figure 7.2). Here, we will only describe the process of validation and storing and searching of previously observed activity sequences.

#### Validation: Pattern Matching

The validation function takes two input parameters, an extended regular expression (ERE) corresponding to some high-level activity class and an activity episode (or dummy episode) assumed to have a single activity instance. The validation function first modifies the ERE and then compares the given sequence of activity steps within the activity episode.

**Modifying the Extended Regular Expression (ERE)** The input ERE is a string array where each element is a name of some activity step or a meta-character or a quantifier from table 3.2. However, some of the quantifiers can be defined in terms of other quantifiers. The purpose of modifying ERE is to reduce the type of different quantifiers. Suppose,  $S$  denotes a particular activity step, then the modification logic follows:

- The plus '+' quantifier is replaced by appending a single occurrence before a kleen star '\*'.  $S^+ = SS^*$
- The count { $n$ } is modified by inserting  $n$  occurrences of the preceding activity step.  $S\{5\} = SSSSS$
- The range { $min,max$ } is modified by inserting  $min$  occurrences of the preceding activity step, follow by ( $max - min$  optional occurrences of the preceding activity step  $S\{3,6\} = SSSS?S?S?$
- The alteration [ $S_1 S_2 S_3$ ] is modified by alternation without subgrouping.  $[S_1S_2S_3] = (S_1|S_2|S_3)$

Therefore, the matching algorithm needs to deal with only the quantifiers  $?,*,|$ , and parenthesis for indicating subgrouping. **Matching Logic** The matching logic is similar to the string

edit distance algorithm where instead of matching a char, we are matching activity step object. However, the special cases, i.e., quantifiers, they have to be handled separately. We use the dynamic programming version of edit distance algorithm. We use a two-dimensional Boolean array storing the match information, and a two-dimensional String array 'path' for storing the path from where it generated.

1. If `pattern.get(j) == sequence.get(i) : dp[i][j] = dp[i-1][j-1]`;
2. If `pattern.get(j) == '?'` : there are two sub conditions:
  - (a) If `pattern.get(j-1) != sequence.get(i) : dp[i][j] = dp[i][j-2]`  
In this case, `sequence.get(i)*` only counts as empty
  - (b) If `pattern.get(i-1) == sequence.get(i) : dp[i][j] = dp[i][j-1]`  
In this case, `sequence.get(i)*` counts as single step  
or `dp[i][j] = dp[i][j-2]`  
In this case, `sequence.get(i)*` counts as empty
3. If `pattern.get(j) == '*'`: there are two sub conditions:
  - (a) If `pattern.get(j-1) != sequence.get(i) : dp[i][j] = dp[i][j-2]` //in this case, `sequence.get(i)*` only counts as empty
  - (b) If `pattern.get(i-1) == sequence.get(i) : dp[i][j] = dp[i-1][j]`  
In this case, `sequence.get(i)*` counts as multiple steps  
or `dp[i][j] = dp[i][j-1]`  
In this case, `sequence.get(i)*` counts as single step  
or `dp[i][j] = dp[i][j-2]`  
In this case, `sequence.get(i)*` counts as empty

Similarly, the path array is assigned with the direction from which it comes, i.e.,

1. If `dp[i][j] = dp[i-1][j-1]`, then `path[i][j] = ↖`
2. If `dp[i][j] = dp[i-1][j]`, then `path[i][j] = ←`
3. If `dp[i][j] = dp[i][j-1]`, then `path[i][j] = ↑`
4. If `dp[i][j] = dp[i][j-2]`, then `path[i][j] = ↑↑`

If the `dp[pattern.size()][sequence.size()]` is true, then a match is found. Otherwise, the path array is used to backtrack and visit alternative paths for a possible match.

**Recursion and Backtracking** Recursion is performed for two reasons:

1. handling alternation and subgrouping defined by the ERE
2. exploring a alternative path by editing the input sequence for a prospective match when a mismatch for a particular activity step occurs.

The different variations of recursion change the two-dimension arrays `dp` and `path`. Therefore, we create a wrapper function and a separate recursive function. The wrapper function initializes the parameters, create the `dp` and `path` arrays, and pass them by calling the recursive function. On the other hand, each time the recursive function calls itself, it clones the partially populated `dp` and `path` arrays, and passes the cloned object. Another point to note is that the recursive call does not start examining the sequence and pattern from the initial position. Therefore, a variable `startI` and `startJ` is passed from where the called function starts checking and populating the arrays.

Whenever alternation and subgrouping occurs, the `pattern.get(j) == '('` is observed. Therefore, the algorithm recursively calls itself for each of the subgroups by initializing the 'startJ'

variable to the start of each subgroup, i.e., the pattern index immediately following '(' and '|'. A special variable 'alternating' is set and passed as a parameter. Therefore, the called function realizes that it is exploring one of the alternative paths. Thus, it discards anything from the pattern when '|' is found until the quantifier ')' is observed, and reset the 'alternating' variable to false.

Sequence editing is achieved by

1. If `sequence.get(i)` has an unnecessary activity step, call with `startI = i+1` and `startJ=j`.
2. If `sequence.get(i)` has a missing activity step, call with `startI = i` and `startJ=j+1`. Mark the activity step of `pattern.get(j)` as a missing one.
3. If `sequence.get(i)` has a wrong activity step, call with `startI = i+1` and `startJ=j+1`. Mark the activity step of `sequence.get(i)` as an incorrect step that occurred instead of `pattern.get(j)`. Modify `dp[i][j]=true` and mark `path[i][j]=''` indicating modification.
4. Sequence with wrong order observed both missing step at one place and extra step at other place while matching.

The algorithm also keeps track of the minimum edits. Therefore, whenever number of edits in a particular path exceeds the previous minimum, then path is not explored any more. Moreover, we initialize the *maximum<sub>edit</sub>* as a function of the sequence length and the pattern length, which guaranties that at a certain point of time, the memory stack will no have more than the *maximum<sub>edit</sub>* number of recursive functions and the exploration path always ends.

### Storing and Searching

We created a dictionary for storing the valid and invalid examples. Since only prefix matching is not enough, we decided to use HashMap for storing and searching. Moreover, there can only be a certain number of variations. Therefore, the overhead is not much and the search is very fast. The dictionary also stores statistical information for checking abnormality. The dictionary class maintains two separate lists for observed valid and invalid activity instances. The hashtable has the following <Key, Value> pair:

**Key:** A string is generated from the sequence of observed activity steps. This string is used as the key for searching.

**Value:** As a value, we create an instance of the output class. Each instance of the output class has the following associated information: - Activity class where the instance belongs.

- Standard activity instance with the particular sequence of activity steps.

- Time statistics (min, max, average, standard deviation) for the delay between steps and duration of each step.

- Occurrence Count

If a particular sequence is observed for the first time, then a key and an object of the output class is created. After that, whenever the same sequence is observed, only the 'Occurrence Count' is incremented and time statistics updated.

## 7.6 Evaluation and Discussion

In this section, we describe how we evaluated our solution using the following data:

1. **CASAS Dataset: Cognitive Assessment Activity Data** This is a publicly available dataset with ADLs/IADLs with data from a total of 400 participants. The participant population includes both healthy adults and patients with mild or moderate dementia.
2. **Collected Data:** We collected data in real home-setting from four different users. Although the users were all healthy adults, we could simulate the different error types in the data based on activity step labeling.

Number	Activity Class
1	Sweep the kitchen and dust the living room
2	Obtain a set of medicines and a weekly medicine dispenser, fill as per directions.
3	Write a birthday card, enclose a check and address an envelope.
4	Find the appropriate DVD and watch the corresponding news clip.
5	Obtain a watering can and water all plants in the living space.
6	Answer the phone and respond to questions pertaining to the video from task 4.
7	Prepare a cup of soup using the microwave.
8	Pick a complete outfit for an interview from a selection of clothing.
9	Check the wattage of a desk lamp and replace the bulb.
10	Wash hands with soap at the kitchen sink.
11	Wash and dry all kitchen countertop surfaces.
12	Place a phone call to a recording and write down the recipe heard.
13	Sort and fold a basketful of clothing containing men's, women's and children's articles.
14	Prepare a bowl of oatmeal on the stovetop from the directions given in task 12.
15	Sort and file a small collection of billing statements.
16	Setup hands for a card game, answer the phone and describe the rules of the game.

TABLE 7.3: The ADL/IADL classes in the CASAS Dataset with cognitive assessment activity data.

### 7.6.1 CASAS Dataset: Cognitive Assessment Activity Data

There are not many activity recognition datasets that have the notion of steps. Although some datasets with labeled activity steps are available in vision research, we did not find any dataset with ADL and IADL where the activity steps are labeled except the 'Cognitive Assessment Activity Data' collected by Washinton State University from Kyoto testbed. The dataset contains data from 400 participant where the population had healthy and dementia patients. We chose this dataset because

- The dataset had activities defined in steps sequence.
- It had real patients with dementia. Therefore, it has activities with abnormalities such as missing steps and delay.

The testbed was instrumented with passive binary, and non-binary sensors and the volunteers were brought to the testbed area for participating in the study. The table ?? lists the activities each participant had to perform during a data collection session. Before each activity, someone from the data collection team would describe the steps required to be performed for that particular activity. An observer continuously monitored the ADLs/IADLs being performed

Preparing Soup Instructions	Defined Rules
1) Participant moves to the kitchen	START
2) Participant removes materials from cupboard	PrepareOatmeal -> (retrieve pot from the cupboard)
3) Participant locates pot	(turns on stove)
4) Participant turns on stove	(fill pot with water)
5) Participant fills pot with water	(put oatmeal into pot)
6) Participant puts oatmeal into pot	(stir oatmeal)
7) Participants times one minute	(put item in the bowl)
8) Participant stirs oatmeal	(turn stove off)
9) Participant puts oatmeal into bowl	(put multiple items in the Bowl)
10) Participant turns stove off	\$
11) Participant puts sugar in oatmeal	retrieve pot from from cupboard ->
12) Participant puts raisin in oatmeal	(open cupboard)
13) Participant throws raisin box in trashcan	(locate item)
(close cupboard)	(retrieve item)
	\$
	fill pot with water->
	(turn on the tap)
	(fill pot)
	(turn off the tap)
	\$
	retrieve item->
	(retrieve pot)
	\$
	put multiple items in the bowl->
	(put item in the bowl)
	(put multiple items in the bowl)
	\$
	put item in the bowl ->
	(put oatmeal in bowl)
	\$
	put item in the bowl ->
	(put sugar in bowl)
	\$
	put item in the bowl ->
	(put raisin in bowl)
	\$
	put item in the bowl -> €
	\$
	END

TABLE 7.4: Example of activity steps and some grammar rules of the activity 'Preparing Oatmeal' instructed to be performed by a user.



and collected ground truth. The data files are numbered per participant which lists triplets <time, sensor, value>, and the fourth column (optional) is the label of activity start/stop, activity step start/stop, and instruction start/stop.

We created grammar rules for each activity just by looking at the description of the provided steps without looking into any data. The first column of table 7.4 shows one example of providing instruction for performing the ADL “ which clearly defines the activity steps, and the second column shows the defined grammar rules. Here, notice the first rule where the activity step ‘turn on stove’ takes place before ‘fill pot with water’ and ‘turn of stove’ takes place after ‘put oatmeal into a bowl’. Even when these two steps are interchanged, it doesn’t matter. Therefore, we also defined additional rules that take care of these alternative scenarios. However, since in this particular dataset the ordering is strictly defined, those extra rules were not necessary.

After accumulating the instructions of all the ADL/IADL, we listed the classes of activity steps that occur in the dataset. Although our algorithm assumes that the input would be activity steps, in this case, the input is raw sensor values. Therefore, we assigned sensor/sensors from the given sensor layout by the authors and using *Apriori* itemset mining algorithm for each activity steps. Although detecting activity steps is not the main focus of this chapter, this preprocessing step needed to be done before we could proceed to evaluate our core solution. One of the difficulty with correlating sensors with the given activity steps was that there was no clear one to one/ one to many mapping from the activity steps to the sensors. In fact, some of the activity steps were too abstract and could not be related to any sensor, for instance ‘stir oatmeal’. On the other hand, several activity steps were too close that same sensors get triggered by those activity steps, such as, ‘retrieves broom from supply closet ’ and ‘retrieves watering can from supply closet ’ only let us know the person was near the supply closet but not what item he/she retrieved. Therefore, we revisit the grammar definition and omit the activity steps that could not be detected from the sensor settings and combine the activity steps from different activity to a single recognizable step and try out both activity steps when a particular sensor is same for both steps.

### Results: Recognizing Complete and Incomplete Activities

The figure 7.6.1 shows the result where blue solid columns are percentage of activity instances in the dataset performed correctly and orange solid column shows percentage of activity instances in the dataset performed partially or incorrectly. We represent our evaluation result with striped column where blue, orange, and solid white boxes indicate reported percentage of complete activity instances, percentage of incomplete activity instances, and percentage of instances not recognized. We used percentage because not every activity was performed by everybody, and therefore the total number of instances in the dataset for each ADL/IADL classes vary.

We find many interesting observation such as for sweeping or filling medicines our solution performed poorly in detecting activity instances that were not performed correctly. Any instance it recognized was reported as a correct instance. Because sensors in the dataset cannot still detect difficulty in performing activity. It could only detect ‘using supply closet’ for fetching broom or ‘opening medicine cabinet’ from door sensors. The activities ‘preparing soup’ and ‘preparing’ oatmeal shows the highest percentage of error, probably because these two activities have the most number of steps defined by the researchers of this dataset. In case of ‘Preparing Oatmeal’ our algorithm performed poorly mainly because the grammar we defined

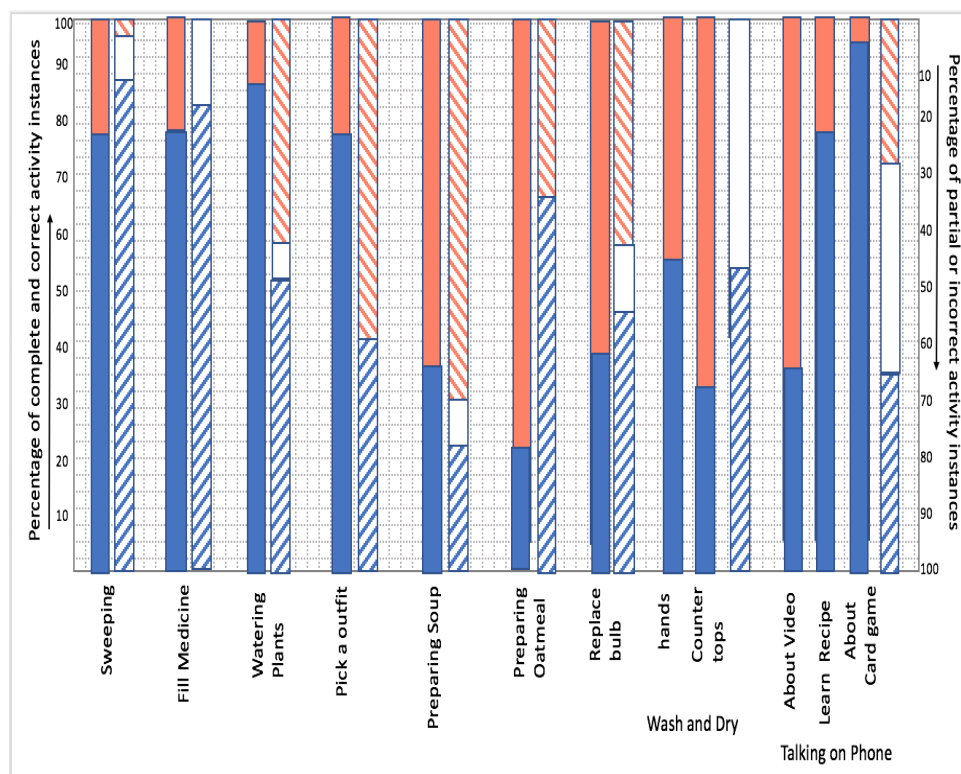


FIGURE 7.4: Percentage of activities correctly recognized (true positive) and percentage of quality correctly assessed [72]

considers adding raisin/sugar as optional steps, so when those steps are missing, it still marks the activity as complete. For some classes there was not any way we could differentiate the rules of different activity classes in terms of recognizable activity steps. For example, we could detect 'talking on the phone', but not what he/she is talking or whether the person received the call or dialed the number. Both for 'washing and drying' and in case of 'talking on the phone', we missed lots of instances, because unless there is sensor value triggered, our algorithm cannot tell whether somebody attempted to do something or not.

## Discussion

**Complexity of the Grammar** It is interesting to find out that grammar can be designed just from the description of the sensor setting. Moreover, the activities in this particular dataset could be described by a regular grammar. Therefore, converting the grammar rules to the extended regular expression (ERE) works and provides greater advantage in matching the detected sequence with the grammar.

**Importance of Sensors** Many of the missing activity steps could not be recognized due to the inability of recognizing those steps with the sensor, and therefore modifying the grammar rules without those steps. If the researcher has the liberty to design the whole system and choose sensors, then the performance of the system will be much higher.

**Importance of Choosing the right Activities** Activity 6,12, 16 provides us only with the information of ‘using the phone’. All other steps are not detectable by the sensors, and therefore is not very helpful in identifying the missing steps or other anomalies from the activity. Whereas activities such as 7 or 14 have more elaborate descriptions and have steps that are both detectable and significant in evaluating patients condition without a human observer.

### 7.6.2 Collected Data

There is a scarcity of public dataset with ADL/IADL instances which labels activity steps or has the notion of steps. Therefore, to evaluate our solution, we designed experiments to collect ADL/IADL data from home and investigate what activity steps occur. Based on data collection description from the literature and our own experience in collecting data for recognizing activity steps, we created a testbed by instrumenting sensors in a real home. We collected activities listed in table 7.5 from four users. All the users were healthy young adults from both gender groups (males and females). A user was wearing multiple waearable sensors in both hands , a sensor like a necklace, and a sensor on the bend. She/he was free to perform more than one activities in parallel. The experiment was not entirely controlled since we did not specify the exact steps to complete the activities or constrain their movement in any way. In fact, one reason for data collection was to observe the variety of activity steps that occur in home settings. Users were free to perform the activities in their own way and were not required to start/stop

Activity List: Collected Data	Example Activity Steps
1. Study	sitting on the study chair using keyboard making typing motion
2. Watching TV	sitting on living room sofa occasional hand gesture for using remote
3. Making Tea	using cabinets using refrigerator heating water gesture of stirring gesture of putting items
4. Eating	sitting on the dining chair hand gesture of eating
5. Washing Dishes	using tap scrubbing dishes rinsing dishes
6. Cooking	using cabinets using refrigerator using microwave using oven hand gesture for chopping ingredients hand gesture for stirring
7. Dressing	choosing outfit from closet motion for changing clothes

TABLE 7.5: ADL/IADL classes observed in the Collected Dataset

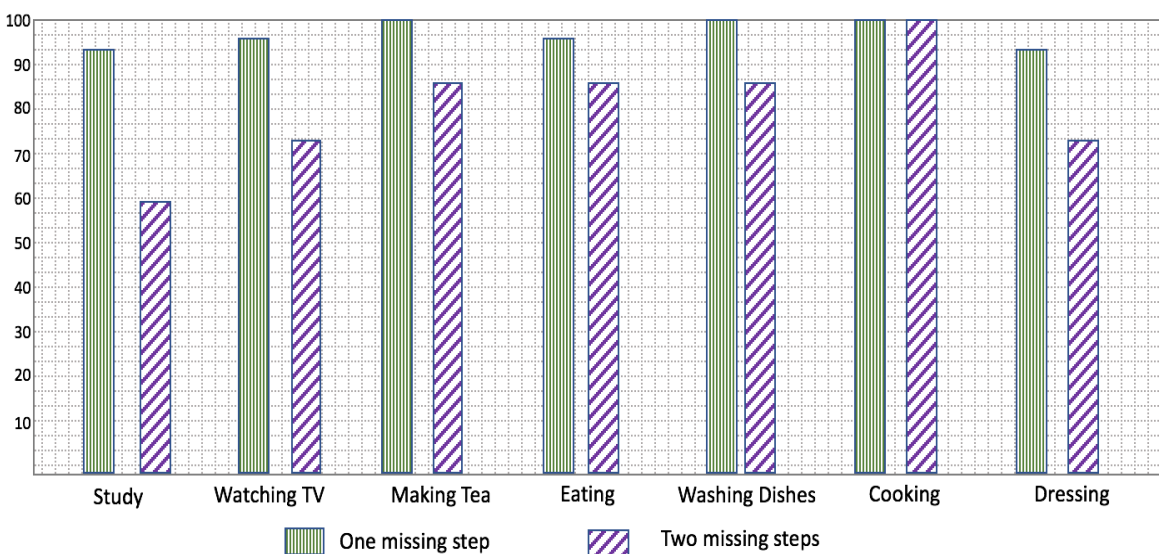


FIGURE 7.5: Percentage of incomplete activities correctly recognized (true positive) when steps are missing

each activity step from a resting position. Each data collection session was about one hour long. A user participated in multiple sessions.

To collect the ground truth data, an observer recorded video of each session. Later, all the activities and activity steps were annotated from the video using ‘Chronoviz’ software [74]. Since the purpose of this chapter to provide analysis on how to detect activities and find activity quality from activity steps, we use the labeled ground truth of activity steps as our input. However, we only use the steps that are recognizable by the underlying sensing system, although we assumed that all the steps would be correctly identified.

### Results: Recognizing Complete Activities

In chapter 6 section 6.5 we have already shown the results of activity recognition based on only the activity steps that were correctly identified by the lower components. The accuracies of detecting high-level activities are 91% to 98% using grammar rules.

Therefore, when we assumed all activity steps are correctly identified and used the ground truth label as sensor input, all the activity instances were detected. In fact, the complex ADL classes where low performance was observed before due to low accuracy in activity steps recognition were more easily detected. For example, ‘cooking’ and ‘making coffee’ triggers similar set of in-situ sensors, but cooking has a lot more activity steps within it which are hard to recognize. But could easily identified when details about the user activity steps are provided.

Here, we want to point out that although data was collected in a real home setting and without constraining the movements and steps, the result still does not reflect the actual world. Because here the activity classes were defined and within the data collection session the user was performing activities only from the given list. For example, sitting on dining chair during data collection was observed only during ‘eating’ activity. But in the real world often activity classes will be encountered that are not defined by the grammar. In those case, human-in-the-loop (the optional verification step described in section 3.2) would be helpful.

Moreover, although each users performed activities differently, the ERE rules are able to recognize all the different variations. However, the since the defined activity classes are quite different, therefore the grammar rule could easily differentiate the instances from different type of activity classes easily.

### **Results: Recognizing Activities with Errors**

Since we collected data from only healthy adults, none of the activity instances had an error. Therefore, to we created a simulated dataset by injecting errors in the stream of activity steps from the collected data. We randomly dropped first one and then a total two steps respectively from each activity instances. . The figure 3.2 shows how our algorithm recognized partially complete activities.

We found that missing activity steps from longer activities vs. shorter activities (in terms of the number of activity steps) affects the performance in activity recognition. Because activity instances that already have very few steps, missing a high number of steps indicates that the number of steps performed by the user was very few (or none). On the other hand, missing two steps did not affect the accuracy of cooking. Again, what activity steps got deleted matters. For instance, while washing dishes the step 'turn on tap' and 'turn off tap' are very important. Again, activities that have variations in definition is not fixed in particular location are also hard to detect. For example, we considered two definitions of 'Studying', one is working on a computer and the other is reading any book on sofa or bed. Since reading itself cannot be detected when the activity step detecting 'sitting on a sofa' is not found, there is no other activity step remaining detecting those variations of the instance. Similar problem is observed in specific variations of instances of 'Watching TV' and 'Dressing' class.



## Chapter 8

# Notification System

Medication adherence is extremely important for effective health outcomes. One of the main reasons behind poor medication adherence is forgetfulness, and reminder systems are usually used to address the problem. This chapter presents MedRem, a novel medication reminder and tracking system on wearable wrist devices. The system is handy and interactive, and it is enriched with several useful features. To address the limitations of the tiny display size of the wrist devices, MedRem incorporates speech recognition and text-to-speech features along with clever interface design. Users interact with the system using their voice commands as well as using the display available on the device. A dictionary based training approach is used on top of the state of the art speech recognition systems to reduce the errors in recognizing the commands from the users. The system is evaluated for both native and non-native English speakers. The error rates for recognizing voice commands are 6.43% and 20.9% for the native and the non-native speakers, respectively, when a off-the-shelf speech recognition system is used. MedRem reduces the error rates to nearly zero percent for the both types of user through a dictionary based training approach. On average, only 1.25 and 15 training commands are required to achieve this performance for the native and the non-native speakers, respectively.

The rest of the chapter is organized as follows. Section 6.1 introduces the problem and section 6.2 highlights the contributions. Section 6.3 describes the *QuActive* framework and our approach, and section 6.4 describes the *QuActive* system design and implementation. Finally, the evaluation and discussion are presented in section 6.5.

### 8.1 Motivation

Proper adherence to prescribed medications is extremely important for health outcomes. The possible consequences of poor medication adherence include reduced effectiveness of treatments, deterioration of health conditions, longer recovery time, increased cost, irrecoverable damages to health, hospitalization, and even death. Despite the severe consequences, the medication adherence rate among patients is significantly low [77] [78] [79]. World Health Organization (WHO) reports that the adherence to long-term therapy for chronic illnesses in developed countries averages 50%, and the rates are even lower for the developing countries [77]. Poor medication adherence is a public health problem, and WHO identifies it as a worldwide problem of striking magnitude[77]. About 33-69% percent of all the medication-related hospital admissions in the United States result from poor medication adherence [80].

One of the main reasons for poor medication adherence is forgetfulness. People often forget to take medication at an appropriate time, and even sometimes make mistakes to take proper

dosages. Though forgetfulness is more prevalent in people with a reduced cognitive ability such as the elderly, it is also very common to healthy and young people, as because many factors beyond cognitive ability like daily routine, habits, and changes in medication regimen result in the forgetfulness regarding medication. For example, though irregular use of contraceptive pill increases the risk of unintended pregnancy, 68.1% of the participants of a study reported missing one or more pills, and 48.9% reported missing two or more during a 3-month study period [81]. In the study, the average age of the participants is 20.9 years, and forgetfulness is listed as one of the main reasons for missing the pills. Several studies show that medication adherence is improved significantly with the use of automated reminder systems [82][83][84].

A number of smartphone applications for medication reminder and tracking are available in the app stores [41] [42]. Researchers have also designed, developed and evaluated reminder systems using smartphones [44][45]. However, smartphone apps are not convenient and effective enough for medication reminder and tracking. Though smartphones can be used by one hand when it is laid on or hanged against some surfaces, most of the times users need to hold the phone by one hand and use it by another. Even for only viewing some notifications, the phone usually needs to be grabbed. Occupation of the hands for using the phones, and the attention required to use the apps justify that smartphone-based systems are significantly intrusive, and are not convenient, particularly for long-term and complex medication regimen. Also, smartphones provide limited effectiveness in different contexts. A user is very likely to miss a reminder at home when the phone is far enough from his/her position at the time the reminder is given. For example, a user may miss a reminder when he/she is busy in the kitchen, but the phone is in the bedroom far away from the kitchen. For contexts like listening to songs, TVs or videos, the user may not perceive the reminder even if the phone is located near him/her. The limitations of the smartphone based reminder systems in such contexts are revealed by a feasibility study [46]. In many situations like in meetings and classrooms, smartphones often need to be kept silent, and users often forget to return the devices back to the non-silent mode when silence is not required anymore. It is very likely that a user misses some reminders in such scenarios.

This thesis chapter presents MedRem, a novel medication reminder and tracking system on wearable wrist devices. As the device is placed on the wrist, it is free from the above-mentioned limitations of the smartphones. However, one of the significant challenges in developing interactive systems for wrist devices is their form factor. The touch screens available on these devices are tiny, and they are much smaller compared to smartphones and tablet computers. *MedRem* enables interactions with the users by incorporating speech recognition and text-to-speech features along with intelligent interface design. It uses the microphone and the speaker along with the touch screen available on the wrist devices to take inputs from and give outputs to the users. The tiny display of the device is used for minimal inputs and outputs, while a user can retrieve and provide more information from/to the system through voice commands. Personalized models are built and updated over time to reduce errors in recognizing users' voice commands, and thus better user experience is provided.



## 8.2 Contributions

This chapter highlights the contributions of the notification subsystem:

- A novel medication reminder and tracking system on wearable wrist devices that is more handy and less intrusive compared to existing systems.
- User interactions are enabled by incorporating text-to-speech and speech recognition features along with clever interface design.
- It is a general purpose reminder and tracking system that can be customized according to the patients' needs.
- A novel dictionary based training approach is used on top of the state of the art speech recognition systems to reduce the errors in recognizing the voice commands from the users.
- The technical accuracy of the system is evaluated for both native and non-native English speakers in controlled experiments.
- The dictionary based training approach reduces error rates to nearly zero percent for both the native and the non-native speakers with only 1.25 and 15 training commands on average, respectively. In contrast, the error rates of using the off-the-shelf speech recognition systems with no training are 6.43% and 20.9%, respectively.

## 8.3 System Description

MedRem is a highly flexible, customizable, and automated system. Most of the functionalities of the system can be configured and customized easily using a cloud platform. Configurations and updates from the cloud are automatically downloaded to the wrist device, and data from the device is also automatically uploaded to the cloud platform. The uploaded data can be used by the caregivers, physicians and other legitimate stakeholders for different purposes including monitoring medication adherence of the user.

### 8.3.1 Operating Script

The core of MedRem is the Operating Script (*OS*) that contains the list of reminders for the user along with necessary settings and information. When required, the *OS* is updated in the cloud, and the system on the watch fetches the updates automatically. There are two components of the *OS*, general settings and reminder list. General settings include *OS* identification number, last date and time the *OS* is updated, user name, and user preferences that are applicable to all the reminders. Each of the entries of the reminder list contains details of a reminder with fields like id, type, time, message and so on. Reminder specific settings are also available in the corresponding entry. The number and values of the fields in an entry depend on the specific reminder. An example of an entry in the reminder list is shown Figure 8.1. The "type" and "time" fields indicate that the reminder is provided daily at 2:00 pm, and the "display symbol" field indicates the symbol that is displayed on the wrist device when the reminder is given. The "message" field contains the message that is provided for the reminder, and the "details" field contains more information about the medication. Some other fields from this example reminder entry are discussed later in this paper.

```
{
  "id": 1,
  "type": "daily",
  "time": "2:00 pm",
  "display symbol": "pill_X",
  "message": "Please take a pill of X ",
  "details": "Take the pill after eating. Report your
physician immediately if there is any side effect.",
  "repetition interval": "00:20:00",
  "repetition period": "05:00:00",
  "repeat message": "Have you taken a pill of X?"
}
```

FIGURE 8.1: Example of a reminder entry in the Operating Script (OS) in JSON format

### 8.3.2 Reminder Life Cycle

MedRem allows the users to reschedule or postpone a reminder. When a reminder is provided, if the user doesn't respond to it or doesn't confirm that the medication is taken, the reminder for that medication is rescheduled at a later time according to the settings of the reminder. In case the user explicitly asks the system to reschedule the reminder at a later time of his/her preference, it is rescheduled according to user's preference instead of the default settings.

A reminder is rescheduled again and again until any of the following occur:

- The user confirms that the medication is taken
- The user explicitly asks the system to stop rescheduling the reminder
- A predefined period of time is elapsed from the original schedule time of the reminder.

For example, the repetition interval and period of the reminder of Figure 8.1 are defined as 20 minutes and 5 hours, respectively. So, the reminder is given first at the specified time at 2:00 pm, and it is repeated periodically with a 20 minute interval for up to 5 hours. If the user confirms that the medication has been taken, or he/she asks the system to stop the repetitions, the reminder is not repeated any more. The default repetition interval for the reminder is changed according to user's preferences, if there is any. As an example, consider a scenario when the user is driving and a reminder is provided, but the user needs one more hour to take the medications. In this case, the user can ask the system to remind him/her after one hour instead of the default interval. This feature allows the user to avoid unnecessary reminders. The life cycle of a reminder is illustrated in Figure 8.2.

The repetitions of a reminder help the user not to forget the medication, as well as being useful for tracking medication intake through the confirmation from the user. Stopping the repetitions after a certain period, which is configurable, ensures that a reminder is not repeated irrelevantly or unnecessarily. For example, the medication that is supposed to be taken at 2:00 pm may be ineffective after 7:00 pm. So, providing reminder of the medication after 7:00 pm is irrelevant. The repetition period as well as the repetition interval can be different for different medication reminders.

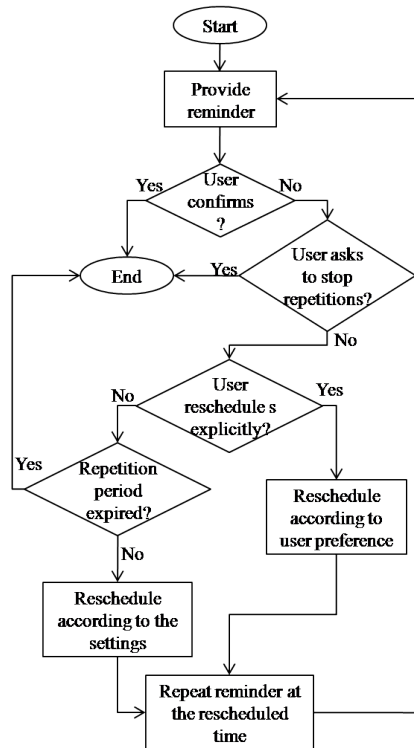


FIGURE 8.2: Life cycle of a reminder

### 8.3.3 Reminder Session

A reminder session starts when the system provides a reminder alert to the user, and it ends when the interaction between the user and the device for the reminder is finished. If a user doesn't respond within some duration after the reminder alert is given, the session terminates automatically. During a reminder session, a user can interact with the system using the touch screen and/or voice commands. When a user uses voice commands, the responses from the system are also given in voice through the speaker of the wrist device.

### 8.3.4 System Architecture

As mentioned earlier, MedRem uses a microphone and a speaker along with the touch screen for taking inputs from and providing outputs to the user. The microphone and the speaker work as input and output media respectively, whereas the touch screen works as both. MedRem is composed of several modules namely I/O Manager, Network Manager, Schedule Manager, Session Manager and Storage Manager. The architecture of the system is shown in Figure 8.3.

The I/O Manager takes inputs from the user, and provides the data to the Session Manager. It also updates the display and speaks through the speaker following the data from the Session Manager. The Schedule Manager is responsible for scheduling the reminders. It maintains a dynamic list of Repeated Reminders (RR) that contains information about the reminders that need to be repeated. Using the OS and the RR, the Schedule Manager schedules the next reminder session, and details of the next reminder session are sent to the Session Manager that starts the next session according to the schedule. The Session Manager manages the whole

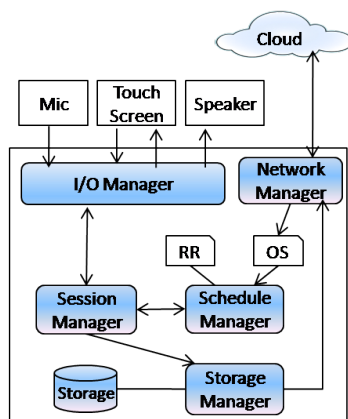


FIGURE 8.3: System Architecture

workflow of a reminder session, including the recognition of the voice commands. It stores necessary information like medication intake confirmations to the storage. Data is stored temporarily on the device, and the Network Manager uploads the data to the cloud periodically. It is also responsible to look for and download the updates available in the cloud. The Storage Manager helps to organize, store and retrieve data to/from the storage.

## 8.4 Solutions

### 8.4.1 Alerts

A major problem of the smart phone based systems is that a user may miss important reminders in different contexts as described earlier. To provide an alert, MedRem vibrates the device. Since the device is attached with the wrist, the user gets the alert if the device is worn. The vibration duration is configurable, and it is set to two seconds by default.

### 8.4.2 User Interaction

#### Interactions through Touch Screen

Due to the form-factor, the information displayed on the screen of the wrist device is kept limited to very few symbols and words, possibly within one or two so that they are bigger in size, and require very little attention or effort from the user to understand. Different symbols and texts are used for different types of reminders. For example, a symbol of a pill is shown when the user is reminded to take a pill, and a symbol of an inhaler is shown when the user needs to use that. Customized symbols for the reminders of different kinds of medications help the user to better comprehend about why the reminder is given, particularly when the user needs to take multiple types of medications with different dosages. Examples of reminder symbols are shown in Figure 8.4. If a reminder is provided for multiple medications, the total number of medications is also displayed on the screen, as shown in Figures 8.4(c) and 8.4(d). When a user needs to be provided with critical information like changes of medicine, dosage or schedule, the display is blinked so that the user can easily understand that some important information for him/her is available there.

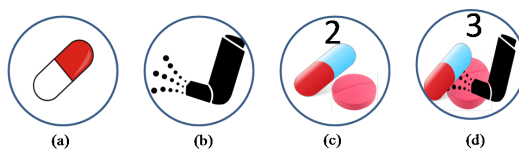


FIGURE 8.4: Some example symbols for a reminder (a) one pill (b) inhaler (c) two pills (d) two pills and the inhaler

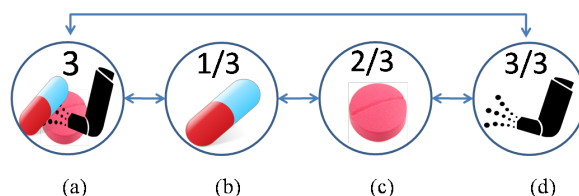


FIGURE 8.5: Example of multiple display pages for a reminder

MedRem not only provides reminders, but also tracks medication intakes. Whenever a reminder is provided, either a fresh reminder or a repeating reminder, if the user has already taken the medications, he/she confirms just by clicking the display. It ensures medication tracking with minimum intervention. If the user hasn't taken the medication, the reminder session can be closed by double clicking on the display, or just by leaving as it is. For the later case, the system automatically closes the reminder session as well as the display after some predefined time period. If medication intake is not confirmed, the reminder is rescheduled at a later, as described earlier.

In cases when a reminder is given for multiple medications, a display page for each of the medications is available in addition to the combined display, as shown in Figure 8.5. Users can navigate between different pages through sweeping the display to the left or to the right. If the user has taken all the medications when the reminder is provided, he/she confirms it just by clicking on the combined display (Figure 8.5(a)). However, if the user has taken some of the medications, that can be confirmed by navigating to and clicking on the corresponding pages. This navigation feature enables easy tracking of partial medication intakes.

Though possible interactions through touch screens are limited, the interactions supported in MedRem the screens are very useful when the users do not need or prefer voice interactions. Most of the times, the user understands the reminder by just getting the vibration and/or looking at the display.

### Interaction through Voice Commands

There are cases when the touch screen is not feasible for exchanging information between the user and the system. MedRem addresses the limitation of the touch screen through enabling voice interactions. Listed below are some of the scenarios when voice interactions are used.

- When the display flips, the user understands that there is some important information there. The user can command the system to provide the information.
- The user needs more information about a reminder beyond what is displayed on the screen.

- The user needs to reschedule the reminder at a later time of his/her preference.
- The user need to record some information related to the medication.
- The hands of the user are occupied, and the user wants to confirm medication intake.

To ensure that MedRem doesn't talk in the contexts where the user doesn't want it to, voice interaction is started during a reminder session only when the user provide some specific commands, known as 'session initiators'. For example, users can use any of the words of "System", "Reminder" or "Device" as session initiator. After providing a reminder, if the system recognizes any of these session initiators, it starts talking with the user.

For any voice commands, except the session initiators, MedRem doesn't require strict format, and so the users only have to memorize few keywords, called the 'command keywords' that are defined for different purposes. The user needs to include corresponding keywords in the voice command for the specific purpose. For example, if a user wants to reschedule a reminder after half an hour, possible expressions or commands include, but not limited to:

*"Remind me after half an hour"*

*"Remind after thirty minutes"*

*"Please ask me after half hour"*

However, these commands need to include any of the keywords "remind" and "ask", as well as it should include a specific interval. In cases when the user provides some information or instructions to the system like medication intake confirmation or to reschedule a reminder, the system repeats what it recognizes to make sure that the information is not recorded incorrectly. A user can repeat the command in case it is not recognized correctly by the system. A user can repeatedly ask the system for the same information even within the same reminder session.

It should be noted that 'Command' or 'Voice Command' in this paper denotes the word or the sequence of words that a user speaks to the system. The sentence "I've taken medicine" is also considered to be a command here. Examples of command keywords, their purposes, and some possible voice commands associated with the keywords are listed in Table 8.1. The command keywords along with their purposes are configurable in MedRem. Even the session initiator keywords can be changed. This feature allows to provide customized options for any user or user group.

Unlike other voice commands, a user can optionally add only any of the words "Hi" or "Hello" in front of the session initiators. This restriction reduces the possibility of starting voice interaction during natural conversations in the user's environment where the user does not intend to start voice interaction with the system, but he/she or someone else nearby utters the session initiator keywords.

### 8.4.3 Voice Command Recognition

For recognizing voice commands, MedRem uses off-the-shelf speech recognition tools that are available in the smart wrist devices like the Android Speech Recognizer [85] for the Android powered watches. However, the actual commands provided by the users, and the words generated by the speech recognition system often differ significantly, particularly for the non-native speakers. Though the problem is less severe for the native speakers, the errors reduce usability of the system, and thus they result poor user experiences. In this paper, the actual text representation of a voice command, and the text output of the speech recognition system for that command are denoted as Actual Command Text (ACT) and Recognized Command Text (RCT), respectively.

TABLE 8.1: Examples of Command keywords, their purposes, regular expressions and related commands

Command Keywords	Purpose	Regular Expression	Example Commands
System Device Reminder	Session Initiator	(Hi Hello)? (System   Reminder   Device)	System Hello Reminder
detail   more	To get more information about the medication	*(detail   more)*	Details please Give me more information Tell me details
repeat   what   say again	To ask the system to repeat what it just told	*(repeat   what   say * again)*	Say it again please Please repeat it What?
done   taken	Confirm that the medication is taken	*(done   taken)*	I've taken it I'm done
yes   no	Answer to a yes/no question from the system	(yes no)*	yes yes, I've taken no
okay   ok   thank	To confirm the system that required information has been received	*(okay   ok   thank)*	Okay Thank you Okay thanks
remind   ask + later   after specific time	To ask the system to remind later (after predefined time interval) or after the interval the user prefers.	*(remind ask)* (later   after TIME)* TIME → NUM hour (and)? NUM minute   NUM hour   NUM minute NUM→ number (and half)   half	Remind me after half hour Ask me after one hour and thirty minute Please remind me later
don't remind	To ask the system not to remind again	*(don't remind)*	Don't remind again Please don't remind

To support the users to provide voice commands in a more natural way, regular expressions based techniques are used in the system. A set of regular expressions are defined corresponding to the command keywords. Table 8.1 shows some command keywords along with the corresponding regular expressions. An *ACT* or a *RCT* is defined as valid if it matches with one and only one of the regular expressions in the system, otherwise it is invalid.

The voice command recognition process in MedRem is depicted in Figure 8.6. Whenever a user provides a voice command to the system, the speech recognition tool is used to retrieve the *RCT* for that command. The voice command is recognized if the *RCT* is valid that means if *RCT* matches with one and only one of the regular expressions, otherwise a dictionary search is carried out as described later. An *ACT* and its *RCT* often differs, and different *RCT*s may be generated for the same *ACT*. So, it is likely that an *RCT* is invalid though the corresponding *ACT* is valid that means the voice command from the user is valid.

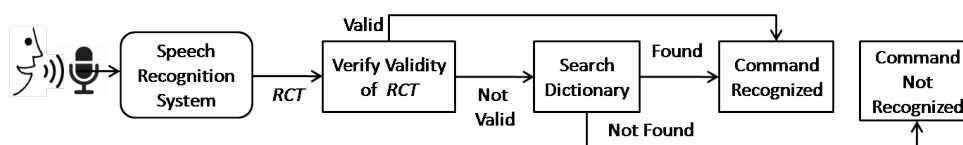


FIGURE 8.6: Voice command recognition process

MedRem uses a personalized dictionary on top of the speech recognition system to reduce the errors in recognizing the intended voice commands. The dictionary maps an invalid *RCT* to a corresponding valid *ACT*. When an invalid *RCT* is generated by the speech recognizer, the dictionary is searched for the *RCT*. If it is found, the command is recognized by the corresponding *ACT* in the dictionary. Otherwise, the command is not recognized. To train the system for a valid command that is not recognized by the system, the user needs to provide the system with both the voice command and the corresponding *ACT*. Text inputs are required to enter an *ACT* to the system, which is not feasible to be done in the wrist devices. So, the user provides the voice command to the wrist device, and inputs corresponding *ACT* using devices with a larger display like a smartphone or a desktop computer. The watch is connected with the larger device using Bluetooth and/or WiFi. The *ACT* provided by the user is tested for validity first, and if the *ACT* is valid, the *RCT* generated from the voice command as well as the *ACT* are entered into the dictionary. When the same invalid *RCT* is generated again from a voice command, the command is recognized using the valid *ACT* from the dictionary. Thus, the errors in recognizing voice commands are reduced through personalized training.

The training is a continuous process. However, the average number of training commands required to achieve nearly zero error rate is not very high even for the non-native speakers due to the fact that limited number of command keywords are defined in the system, and the varieties of commands used by the users are usually not very large.

#### 8.4.4 Cloud Connectivity

The wearable devices used in MedRem are equipped with Bluetooth and Wi-Fi capability. In cases where Wi-Fi access is available, MedRem uses it to connect to the cloud. Otherwise, the wrist device is connected to a smartphone using Bluetooth, and then the system connects to the cloud through the smartphone.



### 8.4.5 Energy Efficiency

Due to the form factor of the wrist devices, the capacity of the batteries available in the devices is usually very limited. So, any system for these devices needs to be energy efficient. MedRem uses the display, the microphone and the speaker only during the reminder sessions. A reminder session typically runs for very short time, usually less than a minute. As stated before, if a user doesn't respond to a reminder within a predefined time period after the reminder is given, the reminder session terminates automatically. The time period is configurable, and it is set to 10 seconds by default. When a reminder session ends, MedRem schedules the next reminder, notifies the underlying operating system of the device about the schedule, and moves to hibernate state. The underlying operating system wakes MedRem up according to the schedule. So, the energy consumption by MedRem depends upon how many reminders need to be provided within a time period, and how long the reminder sessions run. The total time MedRem runs per day is likely to be few minutes, and so it doesn't consume significant energy.

To save energy, automatic upload or download of data to/from the cloud is carried out very sparsely, only once a day by default. However, the frequency can be configured according to the need of the users. In case a user needs MedRem to upload/download data immediately, that can be done just by pressing a button available in the MedRem app in the wrist device. This approach of data exchange between the wrist device and the cloud platform ensures that very less energy is consumed while the needs of the users are not compromised.

## 8.5 Experiment

A prototype of MedRem has been developed using ASUS Zenwatch2, an Android powered smart watch that comes with a microphone, a speaker, and a 1.63 inch touch screen. The Text-to-speech and Speech Recognition features available in the Android platform have been used in the prototype.

### 8.5.1 Data Collection

Data has been collected from 4 native and 6 non-native English speakers. The participants in the study include undergraduate students, graduate students, faculty and a housewife. The command keywords listed in Table 8.1 have been used for the experiment. The participants are provided with mock reminders on the smart watch, and they interact with the system using voice commands. The experiments are carried out in a semi-controlled environment where the actual voice commands provided by the users to the system (the *ACTs*) and the command text generated by the Android Speech Recognizer (the *RCTs*) are recorded by a second person. There is no constraint on the number or order of commands to be used for each of the reminders. Total 292 reminders are provided to the participants, and a total of 1142 commands from the participants are recorded for the reminders. 182 of these commands can not be recognized by the prototype that uses the Android Speech Recognizer only with no training or dictionary. It should be noted that the participants use several commands in a reminder, on average, for the purpose of the experiment. Also, similar commands are repeated during many of reminder sessions. In real deployments, the average number of voice commands used in a reminder will likely be smaller.

## 8.5.2 Analysis

The performance of the system in recognizing the voice commands is evaluated for two types of training approaches: leave one person out (*LOPO*) training and personalized training. Here, accuracy is defined as

$$accuracy = \frac{\text{Number of commands recognized}}{\text{Total number of commands}}$$

As usual, error rate is defined as,  $error\ rate = 1 - accuracy$

In the *LOPO* training, commands of each of the subjects are tested using a dictionary that is trained by the unrecognized commands (the invalid *RCTs*) of the other subjects. Figure 8.7 shows the error rates both when only the speech recognizer is used and when the dictionary built from the *LOPO* training, called the *LOPO* dictionary, is used on top of the speech recognizer. As shown in the figure, the error rates are 6.43% and 20.9% for the native and non-native speakers, respectively, when no dictionary is used. Using the *LOPO* dictionary with the speech recognizer reduces the error rates to 2.96%, and 12.88% for the native and the non-native speakers, respectively. As several of the invalid *RCTs* of a subject do not matches with those from the other subjects, it is manifested that many of the errors are not generic, rather they are person specific.

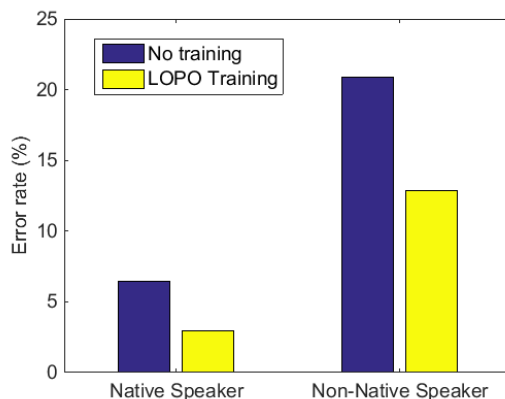


FIGURE 8.7: Error rates in recognizing intended voice commands with speech recognizer only (No training) and with using speech recognizer with leave-one-person-out (*LOPO*) training

For the personalized training, the dictionary for a user is built and updated with the unrecognized commands from him/her in the temporal order the commands are provided. The *RCTs* and the corresponding *ACTs* are used for the training. The updated dictionary is used to recognize the subsequent commands from the subject when the corresponding *RCT* is found to be invalid. Initially the dictionary is empty, and it is updated over time. Figure 8.8 illustrates how errors are reduced with the average number of commands that are used in training the system. The error rates are reduced to nearly zero after using 1.25 and 15 commands, on average, for the training of a native speaker and a non-native speaker, respectively. Once the system is trained for an unrecognized *RCT* command, the command is recognized later if the same *RCT* is generated even though the *RCT* is invalid. So, the error rates reduces if the system is trained

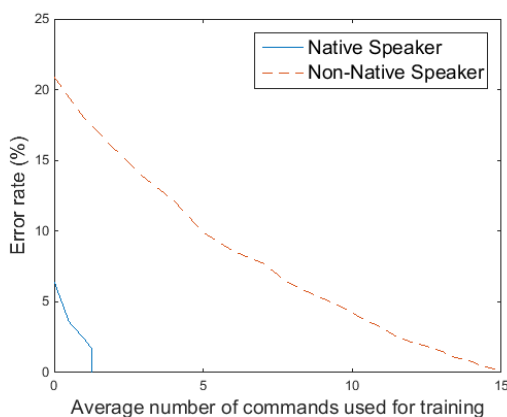


FIGURE 8.8: Reduction of error rates with personalized training

over time. As limited number of command keywords are defined in the system, and the varieties of the commands (using the same set of keywords) used by the users are usually not very large, the average number of training commands required to achieve nearly zero error rate is not high. The required training efforts for a user depends on the variety of commands used by the user as well as the factors like pronunciation and accent of the user that are associated with speech recognition accuracy.

## 8.6 Discussion

As expected, the error rate of speech recognition by the state of the art system is much higher for non-native speakers compared to native speakers. The performance of the speech recognizer may also differ for native speakers with different accents. The dictionary based training approach enables MedRem to be robust in recognizing command even from non-native speakers. Though the system is evaluated using English language only, the design of the system is language independent. Beyond the purpose of a medication reminder and tracker, MedRem can be used for other purposes like providing reminders for exercise and other daily activities, and for tracking well-being of the users. For example, MedRem can be configured to ask the users periodically about their physiological conditions, and to record the users' feedbacks.

One of the limitations of our experiments is the lack of real world deployment. Previously unseen problems along with new issues are often observed when a system is used in real world settings. The effectiveness of MedRem in improving medication adherence has not been explored in this study. However, studies show that automated reminder systems are effective in improving medication adherence, as mentioned before. Considering that MedRem comes with several useful features, and it overcomes many of the limitations of the existing automated systems, it would be more effective in improving medication adherence compared to the state of the arts. MedRem can be extremely useful for people with visual impairment, Essential Tremor, Parkinson Disease, and/or other disabilities. In the future, different aspects of MedRem like its effectiveness and usability will be studied through long term and real world deployments.

To reduce interventions in tracking medication intake, automatic medication tracking features can be integrated into the system. The sensors like the accelerometers and the gyroscopes embedded into the wrist devices can be used to monitor medication intake through tracking

the hand movement of the users. Automatic tracking of the medication using the wrist device will be explored in future endeavors.

Smart wrist devices can be used for multiple useful purposes beyond just as a timepiece, and so the use of these devices has proliferated in the recent years. MedRem can be installed as an additional application on these devices with no/very low additional cost.

## 8.7 Conclusion

This chapter presents *MedRem*, a novel medication reminder and tracking system on wrist devices. State of the art speech recognition tools are combined with novel approaches that makes the system very usable and robust. Experimental results show that with very little efforts from the users, only 1.25 and 15 training commands on average for native and non-native English speakers, respectively, *MedRem* achieves nearly zero percent error rate in recognizing users' voice commands. As literature shows that medication adherence is improved significantly with the use of automated reminder systems [82][83][84], and as *MedRem* overcomes many of the limitations of the state of the art, this system would be very effective toward improving medication adherence.

## Chapter 9

# Conclusion

In this thesis, we addressed the problem of monitoring quality of activities of daily livings (ADL) and instrumented activities of daily livings (IADL) in a home settings. We presented a solution for the problem that does not sacrifice the performance of the activity recognition process and is capable of handling various activity types and number of people. As part of the thesis, we developed four systems and performed experiments to evaluate the algorithms and approaches used in our systems. If we look from broader perspective, we can identify two key areas in Home Monitoring System research where the work in our dissertation have made the most impact. The first one is the research on activity recognition. A lot of interesting works were already available when we started investigating this area. We build our thesis on top of those methods and systems. We identified the problems and parameters that were not fully explored and observed some interesting results. In contrast, we did not find many work on the research of activity quality monitoring, specially targeted to in-home ADLs and IADLs. Therefore, we needed to define specific parameters and plan the experiments carefully by studying related research works. In this chapter, we will summarize our findings and the observations, as well as provide ideas for future exploration.

### 9.1 Summary

In this section, we summarize the results and observations from the studies of activity detection and recognition and activity quality monitoring.

#### 9.1.1 Research on Activity Recognition

##### Novel Segmentation Algorithm

As part of our system SARRIMA, we enhanced the location based apriori algorithm for activity detection by modifying the segmentation algorithm. The modified algorithm considers time difference of sensor triggering in addition to sensor location information. Thereby, making the system work in broader scenarios such as with multiple persons, or constraint apartments where many activity instances are performed within a single room. We see that SARRIMA achieved 97.34% and 98.15% accuracy on average, respectively, on the CASAS Spring and Summer dataset. The accuracy is higher than HMM and SHMM (92% on average) applied on the same dataset [7]. In the ARAS datasets, SARRIMA achieved average accuracy of 87% for House A and 95.3% for House B when considering differentiating activity sub-class, but achieved average accuracy of more than 98% on activity recognition in both houses when considering only high level activity class. For example, recognizing a high level class 'preparing meal' gives

higher performance instead of recognizing the sub-classes ‘preparing breakfast’, ‘preparing lunch’, and ‘preparing dinner’ of the high level class.

### **Techniques for Person Identification without Specialized Sensors**

In SARRIMA, we identified the person who performed the activity from in-situ binary sensors. We observed that in houses (CASAS Spring and Summer datasets) where person to person interaction is much less and each person has their own room and separate routine, the process of identification is simpler since different set of in-situ sensors are triggered by each person or their activity timing is very different. However, the problem is very tricky in houses (ARAS) where the residents lifestyle is shared. The percentage of activities where a person could be identified were very low and by considering binary sensors this percentage could be increased up to a certain extent. We showed how adding very few specialized sensors the percentage number of person to activity assignment almost doubles by using the correlation information among the occupancy episodes. Here, we would like to mention and clarify that SARRIMA refrained from assigning person to activity randomly. Therefore, if we consider the effectiveness of person identification in terms of only the activity instances where assignments were made instead of total activity instances, the accuracy is quite high. Therefore, the technique of occupancy episode correlation and using user behavioral data shows promise in person identification research.

### **A Novel Framework for Activity Recognition based on Grammar**

We designed and implemented a novel activity recognition framework, *QuActive*, which models in-home activities in terms of activity steps. The framework uses grammar rules for establishing relationships among the activity steps within an activity and thereby handles randomness and manages variations of the different activity instances belonging to the same activity class. The framework is reusable and extensible, since changes can be handled just by adding new rules. Our evaluation on three different data sets shows that the framework is capable of handling different activity types and outperforms the state-of-the-art techniques for all of these datasets. The system was also deployed in a real home and in a semi-controlled setting and it could detect 98.6% of the defined activity instances.

## **9.1.2 Research on Activity Quality Monitoring**

### **Defining Quality Parameters and Developing a System for Quality Monitoring**

The main purpose of monitoring in-home activities as part of home health-care systems is to provide information for early detection of diseases. The existing studies focus more on improving the technical aspects of the system but are ambiguous about how the collected information could help. Therefore, as part of this thesis, we attempted to understand the quality of daily in-home activities and defined clear and measurable parameters for accessing the quality of a performed activity. Since recognizing activities when the activity process is not complete or erroneous is challenging, we provided a novel solution that handled the problem. We have also implemented our solution and developed a system that recognizes both completed and partial activities and reports the errors. We tested our system by simulating error in our collected dataset and also with a public dataset having erroneous activity instances from dementia patients.

## **Comprehensive Studies on Understanding Activity Steps in Daily Life**

Recognizing activity steps is extremely important for finding the quality of an activity. However, as it turns out the solution to the problem is not very straightforward. In this thesis, we have documented all the challenges we faced while working on a solution. We have elaborated our attempts in using different sensing modules and the interesting facts we noticed as part of the data collection procedure. We also provided examples to help others understand our point. The ideas for tackling/minimizing some of the problems and our lessons learned have also been cataloged. We believe the provided guidelines will be beneficial to future researchers.

## **9.2 Future Work**

No systems are perfect. Yet, the realization of imperfection gives the opportunity to learn and improve. Our experience while working on the projects in this dissertation have led us to different interesting insights and possible research directions. There are some obvious choices, such as applying our method in other datasets or using different parameters for the algorithms. However, in this section we want to list some ideas we believe are worth exploring. We not only provide research directions for computer science majors, but also exciting future works from various perspectives.

### **9.2.1 Studies in Detection and Recognition of Activity Steps**

One of the core components of our activity quality monitoring system is detecting and identifying activity steps. The success of the upper layer modules depends on the correctness and confidence level of the activity step recognition process. Although, we used classical machine learning algorithms and feature sets, the performance results were below expectation on various scenarios. One of the reasons for this limitation might be the lack of data. Thus, we encourage further studies concentrating specifically in this area, which would likely provide favorable results. Applying more advanced machine learning algorithms and performing feature engineering might be another alternative path to overcome the data limitation. Again, since labeling data for all the different activity steps is really exhausting and prone to error, techniques that use weak labels in other areas of research can also be applied to this particular problem.

### **9.2.2 Real World Deployment with Alzheimer's' Patients**

One of the most important applications of monitoring activity quality is early detection of dementia to prevent the rapid decline of functional and cognitive ability. However, the actual effectiveness of the system cannot be fully realized unless tested in the field and deployed in the house of real Alzheimer's' Patients for monitoring the progression of dementia. This new direction requires collaboration and expertise from both clinicians' and engineers'. It would be helpful to know which activity steps monitoring would be sufficient for activity quality assessment of dementia patients. How should we design a system and choose sensors targeted for those steps thereby reducing cost and complexity? How much impact does the notification system have in the patients' life? Surveys and statistics will also be helpful for social and behavioral studies. In our work, we provided parameters for activity quality, but did not make any inference about the dementia stage. Once the system is in the field, the experience and observation will be valuable in determining automatic dementia screening to a certain extent.

### 9.2.3 Grammar Learning from Activity Data

Another limitation of our work is that we did not learn the grammar rules due to lack of data exemplifying the various use cases. We mitigated this limitation by creating rules after considering the description of sensor settings in the datasets provided by the researchers and the defined activity steps used in their experiments. However, in today's world technology is driven by data and once additional data is available, an exciting research direction would be learning the grammar rules/expression from a large set of real world examples. In our work, we used a variation of the *Apriori* rule mining technique for sensor association with the activity. It would be interesting to find out whether *Apriori* or other mining and association rule learning methods can be used to learn the grammar rules from the data.

### 9.2.4 Multiple People

In this dissertation, we presented an approach that works in a multi person home. However, our evaluation considers two person scenarios. Although, we are confident that the technique will be applicable in homes with more than two people, the performance of the system might drop. In scenarios with a large number of people additional considerations might have to be taken. For example, monitoring activity quality in nursing homes or elderly facilities would have a unique set of challenges and would require further investigation. As part of our studies in multiple homes, we showed effectiveness of using different sensors in person identification. However, we could only simulate the specialized sensors of those where the same binary sensors were used in the environment. Therefore, additional studies comparing and contrasting the effectiveness of all the common person identification sensors would help engineers make more informed choice about sensor settings depending on the application requirements.

### 9.2.5 System Robustness

Various aspects of the system can be modified to make the system more robust.

#### Missing Data vs Missing Activity Steps

In the real world, it is not possible to detect all the activity steps with 100% accuracy. Therefore, the system would report missing activity steps when in reality the activity step could be misclassified. Although we provided an analysis of how activity step recognition accuracy affects activity recognition accuracy in this thesis, we did not analyze its effect on activity quality monitoring. Therefore, studies evaluating which percentage of missing sensor data or error in lower level recognition significantly affects the performance of the activity quality monitoring system up to which extent would make the future systems more robust.

#### Real Time Constraint

In this thesis, we performed evaluation offline from previously collected data or on public datasets. We believe the performance of the system will be the same with real time constraints, but additional testing and validations are necessary. Again, although our system has timing threshold parameters for different types of notifications, it would be interesting to find out what values for those timing thresholds are most effective in the real world. Moreover, it should be studied whether dynamically changing the time thresholds helps in any way.



### **Handling Exceptional Cases**

Our system performance is evaluated based on specific assumptions and there can be real world exceptional examples which goes against those assumptions, such as incoherent behavior from a drunk person, or the activity routine of an extremely busy person. For example, a mother with several small children might perform ADLs in the middle of the night after all the kids have gone to bed. We ignored those extreme cases since they were out of the scope of our research. However, monitoring activity quality for alcoholism or for detecting post pregnancy depression, we will have to consider those cases and change the system settings accordingly.

### **9.2.6 Applications in Related Areas**

Finally, we want to list some example applications outside the home environment where there is a notion of activity steps and requirement of performing the steps in a specific order or having partial ordering among the subset of steps. Most of the listed examples are related to the healthcare domain, although similar applications are observed in other domains as well.

#### **Emergency Medical Responders (EMT)**

An emergency medical responder is a person with specialized training who is among the first to arrive and provide assistance at the scene of an emergency, such as an accident, natural disaster, or other situations calling for medical assistance. In other words, s/he is the first responder aiding injury or illness. Whenever an EMT arrives, s/he has a checklist with specific tasks which need to be performed. Although the checklist might vary slightly depending on the situation or in different organizations, tracking these steps is very helpful. A activity quality monitoring system tailored for this scenario would not only be able to assist the first responder by reminding the tasks one after another, it can store and show the information to the physician treating the patient later.

#### **Nursing Activity Monitoring**

In smart hospitals, nursing activities are monitored for tracking the quality of service and the well-being of patients. Nurses/physicians perform operations based on the patient chart of previous activity steps and sometimes within a specific time limit. Thus, the grammar can be modeled on different nursing steps to identify the steps already performed on a patient, then notify the nurses who are responsible for the next steps.

#### **Surgical Procedure Monitoring**

Surgical operations are often done by a group of people where each person has a specific role to play. Although the activity step performed by each person is usually predefined, performing it depends on the condition of the patient and the activity step performed by other physicians and nurses before him/her. During long surgical operations, doctors can change shifts or leave temporarily in the middle of the operation. Thus, a system monitoring all the steps and their quality during an operation is extremely helpful for the surgeon group as well as ensuring better safety of the patient.

**Worker Training**

Training programs are common in factories, culinary activities, nursing, laboratory, and many other settings. It is natural for a trainee to miss steps or perform steps in the wrong order. Hence, our system can help in such scenarios by monitoring the progress of the trainees and providing real-time notifications and immediate feedback that improves training.

In all the above examples, we can use the system developed in this dissertation with little or no modification by training on activity steps relevant to the particular application, and report whether any required activity steps were missing or done incorrectly. Therefore, we believe this dissertation serves as an important component and provides directions in many future research.

# Bibliography

- [1] *Health expenditure usa, 2018*, <https://www.crfb.org/papers/american-health-care-health-spending-and-federal-budget>.
- [2] "A profile of older american : 2013", U.S. Department of Health and Human Services, Tech. Rep., 2013.
- [3] K. Rockwood, "The measuring, meaning and importance of activities of daily living (adls) as an outcome", *International psychogeriatrics*, vol. 19, no. 03, pp. 467–482, 2007.
- [4] L. Mercier, T. Audet, R. Hébert, A. Rochette, and M.-F. Dubois, "Impact of motor, cognitive, and perceptual disorders on ability to perform activities of daily living after stroke", *Stroke*, vol. 32, no. 11, pp. 2602–2608, 2001.
- [5] R. F. Dickerson, E. I. Gorlin, and J. A. Stankovic, "Empath: A continuous remote emotional health monitoring system for depressive illness", in *Proceedings of the 2nd Conf. on Wireless Health*, ACM, 2011, p. 5.
- [6] J. M. Wiener, R. J. Hanley, R. Clark, and J. F. Van Nostrand, "Measuring the activities of daily living: Comparisons across national surveys", *Journal of Gerontology*, vol. 45, no. 6, S229–S237, 1990.
- [7] D. J. Patterson, D. Fox, H. Kautz, and M. Philipose, "Fine-grained activity recognition by aggregating abstract object usage", in *International Symposium on Wearable Computers*, IEEE, 2005, pp. 44–51.
- [8] D. Lymberopoulos, A. Bamis, and A. Savvides, "Extracting spatiotemporal human activity patterns in assisted living using a home sensor network", *Universal Access in the Information Society*, vol. 10, no. 2, pp. 125–138, 2011.
- [9] M. Van Kleek and H. E. Shrobe, "A practical activity capture framework for personal, lifetime user modeling", in *User Modeling*, Springer, 2007, pp. 298–302.
- [10] J. C. Niebles, C.-W. Chen, and L. Fei-Fei, "Modeling temporal structure of decomposable motion segments for activity classification", in *Proceedings of the European Conference of Computer Vision*, Springer, 2010, pp. 392–405.
- [11] T. S. Barger, D. E. Brown, and M. Alwan, "Health-status monitoring through analysis of behavioral patterns", *Systems, Man and Cybernetics, Part A: Systems and Humans*, *IEEE Transactions on*, vol. 35, no. 1, pp. 22–27, 2005.
- [12] P. Dawadi, D. Cook, C. Parsey, M. Schmitter-Edgecombe, and M. Schneider, "An approach to cognitive assessment in smart home", in *Proceedings of the workshop on Data mining for medicine and healthcare*, ACM, 2011, pp. 56–59.
- [13] D. J. Cook, M Schmitter, *et al.*, "Assessing the quality of activities in a smart environment", *Methods Inf Med*, vol. 48, no. 5, pp. 480–485, 2009.

- [14] M. A. S. Mondol, I. A. Emi, and J. A. Stankovic, "Medrem: An interactive medication reminder and tracking system on wrist devices", in *Proceedings of the Conference on Wireless Health*, ACM, 2016.
- [15] A. L. Lansky, "A representation of parallel activity based on events, structure, and causality", in *Reasoning About Actions & Plans—Proceedings*, 1987, pp. 123–160.
- [16] E. Hoque and J. Stankovic, "Aalo: Activity recognition in smart homes using active learning in the presence of overlapped activities", in *Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, IEEE, 2012, pp. 139–146.
- [17] M. Prosegger and A. Bouchachia, "Multi-resident activity recognition using incremental decision trees", in *Adaptive and Intelligent Systems*, Springer, 2014, pp. 182–191.
- [18] I. A. Emi and J. A. Stankovic, "Sarrima: Smart adl recognizer and resident identifier in multi-resident accommodations", in *Proceedings of the conference on Wireless Health*, ACM, 2015, p. 4.
- [19] B. G. Steele, B. Belza, K. Cain, C. Warms, J. Coppersmith, and J Howard, "Bodies in motion: Monitoring daily activity and exercise with motion sensors in people with chronic pulmonary disease", *Journal of rehabilitation research and development*, vol. 40, no. 5; SUPP/2, pp. 45–58, 2003.
- [20] N. E. Sherwood and R. W. Jeffery, "The behavioral determinants of exercise: Implications for physical activity interventions", *Annual review of nutrition*, vol. 20, no. 1, pp. 21–44, 2000.
- [21] S. Consolvo, D. W. McDonald, T. Toscos, M. Y. Chen, J. Froehlich, B. Harrison, P. Klasnja, A. LaMarca, L. LeGrand, R. Libby, *et al.*, "Activity sensing in the wild: A field trial of ubifit garden", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 2008, pp. 1797–1806.
- [22] H. Kalantarian, N. Alshurafa, and M. Sarrafzadeh, "A wearable nutrition monitoring system", in *11th International Conference of Wearable and Implantable Body Sensor Networks (BSN)*, IEEE, 2014, pp. 75–80.
- [23] J. E. Froehlich, E. Larson, T. Campbell, C. Haggerty, J. Fogarty, and S. N. Patel, "Hydrosense: Infrastructure-mediated single-point sensing of whole-home water activity", in *11th international conference on Ubiquitous computing*, ACM, 2009, pp. 235–244.
- [24] B. Ni, G. Wang, and P. Moulin, "Rgb-d-hudaact: A color-depth video database for human daily activity recognition", in *Consumer Depth Cameras for Computer Vision*, Springer, 2013, pp. 193–208.
- [25] E. M. Tapia, S. S. Intille, and K. Larson, *Activity recognition in the home using simple and ubiquitous sensors*. Springer, 2004.
- [26] D. H. Wilson and C. Atkeson, "Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors", in *Pervasive computing*, Springer, 2005, pp. 62–79.
- [27] H. Alerndar, H. Ertan, O. Incel, and C. Ersoy, "Aras human activity datasets in multiple homes with multiple residents", in *Pervasive Computing Technologies for Healthcare (PervasiveHealth)*, 2013 7th International Conference on, 2013, pp. 232–235.

- [28] A. Benmansour, A. Bouchachia, and M. Feham, "Multioccupant activity recognition in pervasive smart home environments", *ACM Computing Surveys (CSUR)*, vol. 48, no. 3, p. 34, 2016.
- [29] Q. Li and J. A. Stankovic, "Grammar-based, posture and context-cognitive detection for falls with different activity levels", in *Proceedings of Wireless Health Conference, ACM*, 2011, p. 6.
- [30] I. N. Junejo, E. Dexter, I. Laptev, and P. Perez, "View-independent action recognition from temporal self-similarities", *Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 172–185, 2011.
- [31] D. Lymberopoulos, T. Teixeira, and A. Savvides, "Detecting patterns for assisted living using sensor networks: A case study", in *Sensor Technologies and Applications, 2007. SensorComm 2007. International Conference on*, IEEE, 2007, pp. 590–596.
- [32] —, "Macroscopic human behavior interpretation using distributed imager and other sensors", *Proceedings of the IEEE*, vol. 96, no. 10, pp. 1657–1677, 2008.
- [33] S. C. Geyik and B. K. Szymanski, "Event recognition in sensor networks by means of grammatical inference", in *INFOCOM 2009, IEEE*, IEEE, 2009, pp. 900–908.
- [34] D. Moore and I. Essa, "Recognizing multitasked activities from video using stochastic context-free grammar", in *AAAI/IAAI*, 2002, pp. 770–776.
- [35] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele, "A database for fine grained activity detection of cooking activities", in *Computer Vision and Pattern Recognition, IEEE*, 2012, pp. 1194–1201.
- [36] J. Lei, X. Ren, and D. Fox, "Fine-grained kitchen activity recognition using rgb-d", in *Proceedings of the Conference on Ubiquitous Computing, ACM*, 2012, pp. 208–211.
- [37] R. Blasco, Á. Marco, R. Casas, D. Cirujano, and R. Picking, "A smart kitchen for ambient assisted living", *Sensors*, vol. 14, no. 1, pp. 1629–1653, 2014.
- [38] A. W. Armstrong, A. J. Watson, M. Makredes, J. E. Frangos, A. B. Kimball, and J. C. Kvedar, "Text-message reminders to improve sunscreen use: A randomized, controlled trial using electronic monitoring", *Archives of dermatology*, vol. 145, no. 11, pp. 1230–1236, 2009.
- [39] E. O. Kharbanda, M. S. Stockwell, H. W. Fox, and V. I. Rickert, "Text4health: A qualitative evaluation of parental readiness for text message immunization reminders", *American journal of public health*, vol. 99, no. 12, pp. 2176–2178, 2009.
- [40] M. Y. Hou, S. Hurwitz, E. Kavanagh, J. Fortin, and A. B. Goldberg, "Using daily text-message reminders to improve adherence with oral contraceptives: A randomized controlled trial", *Obstetrics & Gynecology*, vol. 116, no. 3, pp. 633–640, 2010.
- [41] *Android app store*, <https://play.google.com/store/apps?hl=en>.
- [42] *Apple app store*, <https://itunes.apple.com/us/genre/ios/id36?mt=8>.
- [43] K. Stawarz, A. L. Cox, and A. Blandford, "Don't forget your pill!: Designing effective medication reminder apps that support users' daily routines", in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, ACM*, 2014, pp. 2269–2278.
- [44] M.-Y. Wang, P. Tsai, J. W.-S. Liu, and J. K. Zao, "Wedjat: A mobile phone based medicine in-take reminder and monitor", in *Bioinformatics and BioEngineering, 2009. BIBE'09. Ninth IEEE International Conference on*, IEEE, 2009, pp. 423–430.

- [45] J. M. Silva, A. Mouttham, and A. El Saddik, "Ubimed: A mobile application to improve accessibility and support medication adherence", in *Proceedings of the 1st ACM SIGMM international workshop on Media studies and implementations that help improving access to disabled users*, ACM, 2009, pp. 71–78.
- [46] M Hayakawa, Y Uchimura, K Omae, K Waki, H Fujita, K Ohe, *et al.*, "A smartphone-based medication self-management system with realtime medication monitoring", *Appl Clin Inform*, vol. 4, no. 1, pp. 37–52, 2013.
- [47] M. A. S. Mondol and J. A. Stankovic, "Harmony: A hand wash monitoring and reminder system using smart watches", *EAI Endorsed Transactions on Ambient Systems*, vol. 15, no. 6, 2015. DOI: [10.4108/eai.22-7-2015.2260042](https://doi.org/10.4108/eai.22-7-2015.2260042).
- [48] E. Årsand, M. Muzny, M. Bradway, J. Muzik, and G. Hartvigsen, "Performance of the first combined smartwatch and smartphone diabetes diary application study", *Journal of diabetes science and technology*, vol. 9, no. 3, pp. 556–563, 2015.
- [49] V. Sharma, K. Mankodiya, F. De La Torre, A. Zhang, N. Ryan, T. G. Ton, R. Gandhi, and S. Jain, "Spark: Personalized parkinson disease interventions through synergy between a smartphone and a smartwatch", in *Design, User Experience, and Usability. User Experience Design for Everyday Life Applications and Services*, Springer, 2014, pp. 103–114.
- [50] F. Sailer, M. Pobiruchin, M. Wiesner, and G. Meixner, "An approach to improve medication adherence by smart watches", *Studies in health technology and informatics*, vol. 210, p. 956, 2015.
- [51] H Schipper, J. Clinch, A McMurray, and M Levitt, "Measuring the quality of life of cancer patients: The functional living index-cancer: Development and validation.", *Journal of clinical Oncology*, vol. 2, no. 5, pp. 472–483, 1984.
- [52] J. Gong, K. M. Rose, I. A. Emi, J. P. Specht, E. Hoque, D. Fan, S. R. Dandu, R. F. Dickerson, Y. Perkhounkova, J. Lach, *et al.*, "Home wireless sensing system for monitoring nighttime agitation and incontinence in patients with alzheimer's disease", in *Proceedings of the conference on Wireless Health*, ACM, 2015, p. 5.
- [53] S. Alqassim, M. Ganesh, S. Khoja, M. Zaidi, F. Aloul, and A. Sagahyoon, "Sleep apnea monitoring using mobile phones", in *2012 IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom)*, IEEE, 2012, pp. 443–446.
- [54] B. Jin, T. H. Thu, E. Baek, S. H. Sakong, J. Xiao, T. Mondal, and M. J. Deen, "Walking-age analyzer for healthcare applications", *IEEE journal of biomedical and health informatics*, vol. 18, no. 3, pp. 1034–1042, 2014.
- [55] M. W. Whittle, *Gait analysis: an introduction*. Butterworth-Heinemann, 2014.
- [56] L. P. Malasinghe, N. Ramzan, and K. Dahal, "Remote patient monitoring: A comprehensive study", *Journal of Ambient Intelligence and Humanized Computing*, vol. 10, no. 1, pp. 57–76, 2019.
- [57] D. T. Anderson, M. Ros, J. M. Keller, M. P. Cuéllar, M. Popescu, M. Delgado, and A. Vila, "Similarity measure for anomaly detection and comparing human behaviors", *International journal of intelligent systems*, vol. 27, no. 8, pp. 733–756, 2012.
- [58] V. Jakkula, D. J. Cook, and A. S. Crandall, "Temporal pattern discovery for anomaly detection in a smart home", 2007.

- [59] E. Hoque, R. F. Dickerson, S. M. Preum, M. Hanson, A. Barth, and J. A. Stankovic, "Holmes: A comprehensive anomaly detection system for daily in-home activities", in *2015 International Conference on Distributed Computing in Sensor Systems*, IEEE, 2015, pp. 40–51.
- [60] H. Gokalp and M. Clarke, "Monitoring activities of daily living of the elderly and the potential for its use in telecare and telehealth: A review", *TELEMEDICINE and e-HEALTH*, vol. 19, no. 12, pp. 910–923, 2013.
- [61] M. Buettner, R. Prasad, M. Philipose, and D. Wetherall, "Recognizing daily activities with rfid-based sensors", in *Proceedings of the 11th international conference on UbiCom*, ACM, 2009, pp. 51–60.
- [62] U. Maurer, A. Smailagic, D. P. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions", in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on*, IEEE, 2006, 4–pp.
- [63] R. Agrawal, R. Srikant, *et al.*, "Fast algorithms for mining association rules", in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499.
- [64] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise.", in *Kdd*, vol. 96, 1996, pp. 226–231.
- [65] R. Dickerson, E. Hoque, P. Asare, S. Nirjon, and J. Stankovic, "Resonate: Reverberation environment simulation for improved classification of speech models", in *Information Processing in Sensor Networks, IPSN-14 Proceedings of the 13th International Symposium on*, 2014, pp. 107–117. DOI: [10.1109/IPSN.2014.6846745](https://doi.org/10.1109/IPSN.2014.6846745).
- [66] H. Junker, O. Amft, P. Lukowicz, and G. Tröster, "Gesture spotting with body-worn inertial sensors to detect user activities", *Pattern Recognition*, vol. 41, no. 6, pp. 2010–2024, 2008.
- [67] Z. Lu, X. Chen, Q. Li, X. Zhang, and P. Zhou, "A hand gesture recognition framework and wearable gesture-based interaction prototype for mobile devices", *IEEE transactions on human-machine systems*, vol. 44, no. 2, pp. 293–299, 2014.
- [68] C. Zhu and W. Sheng, "Wearable sensor-based hand gesture and daily activity recognition for robot-assisted living", *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 569–573, 2011.
- [69] S. Geman and M. Johnson, "Probabilistic grammars and their applications", *International Encyclopedia of the Social & Behavioral Sciences*, pp. 12 075–12 082, 2002.
- [70] N. Saeedloei and G. Gupta, "Timed definite clause omega-grammars", in *LIPICs-Leibniz International Proceedings in Informatics*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, vol. 7, 2010.
- [71] D. Riboni, C. Bettini, G. Civitarese, Z. H. Janjua, and R. Helaoui, "Fine-grained recognition of abnormal behaviors for early detection of mild cognitive impairment", in *Proceedings of the International Conference on Pervasive Computing and Communications*, IEEE, 2015, pp. 149–154.
- [72] G. Singla, D. J. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies", *International journal of biosciences, psychiatry, and technology*, vol. 1, no. 1, p. 25, 2009.

- [73] ———, “Recognizing independent and joint activities among multiple residents in smart environments”, *Journal of ambient intelligence and humanized computing*, vol. 1, no. 1, pp. 57–63, 2010.
- [74] A. Fouse, N. Weibel, E. Hutchins, and J. D. Hollan, “Chronoviz: A system for supporting navigation of time-coded data”, in *CHI*, ACM, 2011, pp. 299–304.
- [75] A. Association *et al.*, “2018 alzheimer’s disease facts and figures”, *Alzheimer’s & Dementia*, vol. 14, no. 3, pp. 367–429, 2018.
- [76] K. Lyons, H. Brashear, T. Westeyn, J. S. Kim, and T. Starner, “Gart: The gesture and activity recognition toolkit”, in *Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments*, Springer, 2007, pp. 718–727.
- [77] E. Sabaté, *Adherence to long-term therapies: evidence for action*. World Health Organization, 2003.
- [78] A. Bozek and J. Jarzab, “Adherence to asthma therapy in elderly patients”, *Journal of Asthma*, vol. 47, no. 2, pp. 162–165, 2010.
- [79] P. Latry, M. Pinet, A. Labat, J.-P. Magand, C. Peter, P. Robinson, K. Martin-Latry, and M. Molimard, “Adherence to anti-inflammatory treatment for asthma in clinical practice in france”, *Clinical therapeutics*, vol. 30, pp. 1058–1068, 2008.
- [80] L. Osterberg and T. Blaschke, “Adherence to medication”, *New England Journal of Medicine*, vol. 353, no. 5, pp. 487–497, 2005.
- [81] J. D. Smith and D. Oakley, “Why do women miss oral contraceptive pills? an analysis of women’s self-described reasons for missed pills”, *Journal of Midwifery & Women’s Health*, vol. 50, no. 5, pp. 380–385, 2005.
- [82] M. Vervloet, A. J. Linn, J. C. van Weert, D. H. De Bakker, M. L. Bouvy, and L. Van Dijk, “The effectiveness of interventions using electronic reminders to improve adherence to chronic medication: A systematic review of the literature”, *Journal of the American Medical Informatics Association*, vol. 19, no. 5, pp. 696–704, 2012.
- [83] R. E. Sarabi, F. Sadoughi, R. J. Orak, K. Bahaadinbeigy, *et al.*, “The effectiveness of mobile phone text messaging in improving medication adherence for patients with chronic diseases: A systematic review”, *Iranian Red Crescent Medical Journal*, no. In Press, 2016.
- [84] S. L. Spoelstra, C. W. Given, A. Sikorskii, C. K. Coursaris, A. Majumder, T. DeKoekkoek, M. Schueller, and B. A. Given, “Feasibility of a text messaging intervention to promote self-management for patients prescribed oral anticancer agents.”, in *Oncology nursing forum*, vol. 42, 2015, pp. 647–657.
- [85] *Android speech recognizer*, <http://developer.android.com/reference/android/speech/SpeechRecognizer.html>.