UNIVERSITY OF CALIFORNIA,
IRVINE

A Life Event Detection System Using Real-Time Heterogeneous Context Monitoring and Formal Concept Analysis

THESIS

submitted in partial satisfaction of the requirements
for the degree of

MASTER OF SCIENCE

in Computer Science

by

Hyungik Oh

2015

DEDICATION

To my father, Jay, and my mother, Misuk
Who hooked me on study.

To my little sister Jiyoon and Jo's family, aunt Eunjoo,

Love you all

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

John 3:16 For God so loved the world that he gave his one and only Son, that whoever believes in him shall not perish but have eternal life. First and foremost, I would like to thank God, His Almighty, His forever love and His many blessings.

I would like to express the deepest and sincere gratitude to Bren Professor Ramesh Jain, my advisor who is a leader of machine vision, for giving me the chance to research with him and for providing me his invaluable guidance. I admire his keen insight, thought before the age, and most of all, his enthusiasm in regard to research and teaching. It is a great honor to become his PhD student now and to research together in succession. I really appreciate his help, advice and full support about my research.

I would like to thank my committee member, Professor Nalini Venkatasubra manian. Her valuable advice for mobile sensing data management and battery control inspired me to improve this life event detection research. In addition, I thank another committee member, Professor Marco Levorato who was my Computer & Communication Networks class professor. His teaching helped me a lot to construct this event detection system.

I am also very happy to study with my friends in Social Life Networks Lab, Laleh, Siripen, Mengfan, Eric and Pranav. I love to discuss with them in regard to our research line. In addition, special thanks to Jong Hak, Young Seok, Taewoo, Junghun, Myoung seo, Sun Jun, Sun Young, Jaewoo, Yunho, Youngho, and Minsik. Without their help getting master degree would not have been possible.

I am extremely grateful to my parents for their love, support, sacrifice, comforting and challenging. Whenever I fell into despair, my mother's comforting words "You are my lovely son Hyungik" always encouraged me to keep up my study. My father's coaching, advising, financial support and trust about what I am doing is great encouragement. I will make you happy. Love you all!

Finally, I would like to thank all the people who have helped me to finish this work and writing directly or indirectly.

ABSTRACT OF THE THESIS

A Life Event Detection System Using Real-Time Heterogeneous Context Monitoring and Formal Concept Analysis

By

Hyungik Oh

Master of Science in Computer Science

University of California, Irvine, 2015

Professor Ramesh Jain, Chair

The real-time detection of personal life events has great potential to provide human-friendly services. Using detected life events in the fields of health care, smart homes, elderly surveillance, and smart car, etc. will provide more relational information and/or services to a user. However, automatically detecting a life event from human behaviors and their surrounding contexts is a challenging problem. I believe that the combination of Formal Concept Analysis (FCA) and a context-aware mobile computing system can help a path toward automated life event detection. The advancement of the smartphone and its embedded sensors will enable this scenario. I propose a generic context-aware system based on Formal Concept Analysis, which covers both front-end and back-end processing, to detect human life events. The main contributions of this system are: 1) it provides a framework for real-time personal monitoring; 2) it integrates, processes and stores personalized latent features; and 3) it combines heterogeneous data streams into one life event. The experimental validation, which I implemented on an Android platform and server, demonstrates that a general life event model can be applied to each individual and that concept data analysis can be substituted for statistical data analyses in life event detection. I believe that my findings can lead to new event detection approach, which is not just confined to specific environments, such as Social Life Networking, video streams, or artificially organized sentient locations, but can be opened for all environments in the real world by using sensors and context factors.

CHAPTER 1. INTRODUCTION

Chronological analyses of personal life events can provide fundamental information about human beings. Understanding human life has always been the objective of various studies. For example, clinicians measured activities of daily living (ADL), which are the basic tasks for everyday life (i.e. eating, bathing or dressing), to consider the capability of performing basic daily activities [15,16]. S. Sandberg et al. have revealed that negative life events can severely affect the exacerbation of a new asthma attack [17]. McVeigh-Schultz et al. have studied context-specific interactions between drivers and their cars [18, 19], and Mennicken et al.'s smart home research insisted that understanding the life courses of users may support the iterative, incremental process of smart home technology adoption [20]. Thus, comprehending the relationships between humans and their surrounding conditions is a significant research area in the fields of health care, smart homes, elderly surveillance, and car-human-relationships, etc. with the aim of providing more user-friendly services. I contend that life events and event analysis can become a standard variable for understanding the present situations of people. Finally, the chronological analysis of life events will facilitate planning the future as well.

Technological advancement can now enable researches to automatically detect the events of daily life. With the emergence of the smart phone and its embedded sensors, such as accelerometers and GPS, it is possible to continuously track the physical activities and current positions of users. For example, the analysis of the values of an accelerometer sensor can correctly predict some basic physical activities, such as standing still, walking, running, bicycling and sitting in a vehicle, and the tracking of GPS points in a map application can show the places visited and how long people stayed there. One user tracking mobile application, called MOVES, organizes this information in chronological sequence and displays a daily tracking summary in the timeline. With smartphone's sensory data, now it is possible to automatically detect a life event, if additional meaningful information (i.e. venue, calendar events, photos, media or application usage etc.), which are extractable from a smartphone, are properly synthesized. I believe that this new method can be substituted for common event detection, which depends on cameras, audio, SNS or modulated sentient places. In order to achieve automated event detection, a prediction technique is necessary to organize the disordered raw data. Given the limitations of an environment with variable numbers of subjects, privacy issues, and the challenges of personal data, however, the difficulty lies in training a generalized statistical model and predicting the outcomes with machine learning techniques. Therefore, Formal Concept Analysis can be an alternative means of making a general model and identifying specific situations from the logs.

Formal Concept Analysis (FCA) provides a way to find information in data with concept-based data analysis [21]. Given the limitations of collecting life logs for a personalized statistical model, as noted above, the properties of FCA, such as not using mathematical operations, but emphasis on recognizing and structural similarities, can completely satisfy the conditions of life event detection. FCA defines a relationship which is a triplet $T = (O, A, R)$, where O is a set of objects, A is a set of attributes, and a relation or incidence R is $R \subseteq O \times A$, and requires concept pairs (X, Y), where $X \subseteq O, Y \subseteq A$. These concept pairs are organized as a conceptual representation, called a 'concept lattice,' which consists of partially ordered sets, which then helps to extract information using the relationships between objects and attributes [21]. To apply FCA for life event detection, FCA will use concept pairs $(Life\ event, Life\ logs)$, which are extractable from knowledge-based definitions of these relationships. A mobile cooperative system will enable the automatic construction of the concept pairs, navigate a concept lattice, and finally detect a life event in real-time.

The main goals of the proposed life event detection system are:

1) To provide a framework for real-time personal monitoring.

2) To integrate, process and store personalized latent features.

3) To combine heterogeneous data streams into one life event.

The constructed system will determine whether Formal Concept Analysis is applicable in detecting life events. The evaluation will demonstrate whether a general life event model can be applied to each individual in a sample and whether concept data analysis can be substituted for statistical data analyses in life event detection. This research line is meaningful because understanding human life events in real-time will help to provide more user-friendly services in the fields of health care, smart homes, elderly surveillance, car-human-relationships, or any user-related recommender systems. I believe that the combination of context-aware mobile computing and Formal Concept Analysis will lead to a powerful real-time event detection engine.

CHAPTER 2. RELATED RESEARCH AND SYSTEMS

Research into event detection, or into situation recognition that tracks an object and extracts some meaningful semantics from this tracking has become popular. However, to the best of my knowledge, there is no smartphone-specialized system to detect labeled life events by tracking a human, collecting life-logs, and analyzing the logs in real-time through a total directional approach, which spans from chronological observation and analysis of user contexts to a prediction model [4]. One study, which uses sensory data and concept analysis, tries to recognize situations in their modulated environments. This study's hypothetical example of an elderly person living alone in a sentient house illustrates the usefulness of a versatile lattice-based model [8]. They re-demonstrated the model using real-time human activity recognition in an indoor home environment by ambient sensors [7]. Chien Chin Chen et al. have presented a generic framework LIPED and tried to model the activeness trends of events using HMM-based life profiles. To model event activeness, they aggregated a set of chronologically ordered document streams from different kinds of information sources [22]. Jianshu Weng et al. demonstrated their life events detection strategy by analyzing text stream in *Twitter*. They tried to filter a huge amount of meaningless "babbles" in the text streams, and then analyzed frequency-based raw signals of the words [23]. Changsheng Xu et al. analyzed broadcast sports video streams through the web casting, and tried to detect personalized sports video events [24]. Yan Ke et al. suggested a framework that contains a real-time event detector, which learns a cascade of filters based on volumetric features. The event detector scans video sequences in space and time, and tries to catch actions on real-world video sequences [25]. In contrast to the approaches above, I focus more on a smartphone-centric, sensor-based system with that uses practical, real-time and automated processes, which do not require any user intervention. Given the limitations of collecting a sufficient quantity of life logs for a personalized statistical model, I apply concept data analysis rather than using statistical data analysis for an event prediction model.

CHAPTER 3. CONTEXT AND LIFE EVENTS

Since Schilit and Theimer [1] defined context, many researchers have tried developing their own meanings for this term and have built context-aware applications according to their definitions. Context in the work of Schilit et al. [1] is the location and the identities of people within the surrounding environment. Similarly, Ryan et al. [2] refer to context as location, environment, identity and time. The definition of context by A.K Dey et al. [3] is information that characterizes the situation of an entity. Some other researchers quote a lexical semantic definition of context such as “the circumstances that shape an event.” [5] Therefore, to construct a context-aware mobile computing system, the distinct meaning of my context needs to be first defined.

3.1 Definition of Context

Context is any components of the surrounding conditions of an entity or information that describes the entity itself. Contexts represent relationships between an entity and the circumstances that shape an event, as well as the status of the entity itself.

I define *context* based on A.K Dey et al. [3] and Gerald et al. [5], and confine *entity* to be persons. Particularly, the context in the life event detection signifies one piece of multiple life-log streams. For example, life-log streams of “sleeping at home” will include spatial data, temporal data, sensory data, and entertainment data, and contexts can be latitude, longitude, venue, time window, time band, Unix long time, physical activity, activity level, application and media usage etc. Therefore, the data that describes the

Algorithm 1 Context Change Detection Algorithm

```
Input: $T_i = (F_1, F_2, F_3, ..., F_n)$;
        // $1 \le i \le 288$, n = the number of context factors
Output: TRUE, FALSE
 1: if C-C-D is tracking context change
 2:     Calculate distance between $T_k$ and $T_i$;
        // $T_k$ = Target time-window,
        // $T_i$ = Incoming time-window
 3:     if distance > threshold
 4:         return TRUE;
            // Detect context change, and stop tracking
 5:     else if maximum tracking time up
 6:         return TRUE;
            // Stop tracking
 7:     else i++;
 8:         return FALSE;
            // Wait next time-window, keep tracking
 9:     endif
10: else if $T_i$ == tracking event factor
11:     Set $T_i$ to target time-window $T_k$, start tracking;
12:     i++;
13:     return FALSE;
14: endif
```

surrounding conditions of a person's life is called context, and the contexts finally organizes a given life event.

3.1.1 Beyond Context

Given the definition of context above, estimating a method for how to detect a life event from multiple data streams is possible. A fundamental assumption of the life event detection is that a day is composed of different life events, and the life events are characterized by different sorts of contexts. Hence, analyzing a day with transitions between sets of contexts can give a fresh insight into an approach for events detection. Algorithm 1 suggests one way to understand the transitions between contexts.

Table 3.1 Examples of Incoming *timewindow*

Time Window	Activity Level	Step Count	Media	Setting	App Count	Time Band	Week
20141004_173	0	0.049893	0	0	0.007912	2	1
20141004_174	0	0.049893	0	0	0.049299	2	1
20141004_175	1.375	0.423665	0	0	0.000693	2	1
20141004_176	1.307692289	0.049893	0	0	0.049299	2	1
20141004_177	1.333333373	0.049893	0	0	0.000693	2	1

Table 3.2 Example of Target *timewindow*

Time Window	Activity Level	Step Count	Media	Setting	App Count	Time Band	Week
20141004_1	0	0.049893	0	0	0.000693	0	1

Tables 3.1 and 3.2 are some life-log samples. To detect context changes, I suggested one method in *smartNoti* research [4] that calculating a L2-norm Euclidean distance between two *timewindow*s and then checking whether this was over a dynamic threshold of $\sqrt{2}$. The distance $\sqrt{2}$ means that at least two discrete factors or combination of discrete and continuous factors had changed (i.e. $\sqrt{(1-0)^2+(1-0)^2}$). Specifically, $\sqrt{2}$ can signify a variation from standing still to walking, standing still to 145 steps, or combination of media usage and sound setting change etc. The distance $\sqrt{2}$, however, does not mean an absolute standard, but It is first assumed as an initiation value and then dynamically changed according to user's situations.

To gain insights about how to detect a human life event, I randomly chose one day from all my experiment [4] and tried to express the context transitions by drawing distances. Figures 3.1, 3.2, 3.3 and 3.4 show the result of comparing the context distance to a datum point (*timewindow)* at 12:00 am. Each point in the x-coordinate indicates the frequency of collecting a life log matrix, a period of fives minutes, and the y-coordinate shows the distance from the datum point.

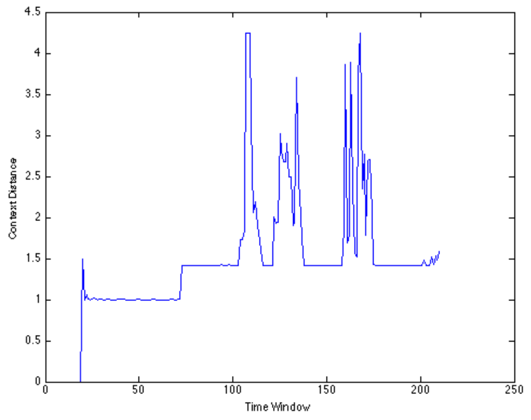


Figure 3.1 Context Transitions of Subject1

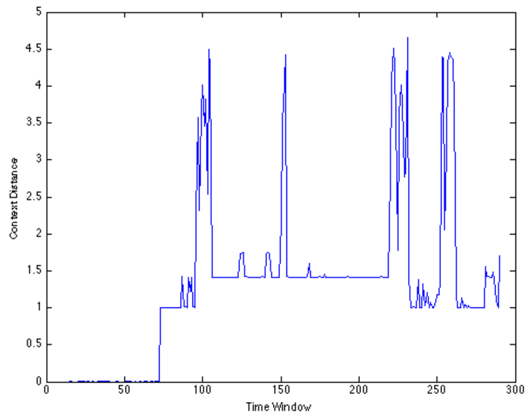


Figure 3.2 Context Transitions of Subject2

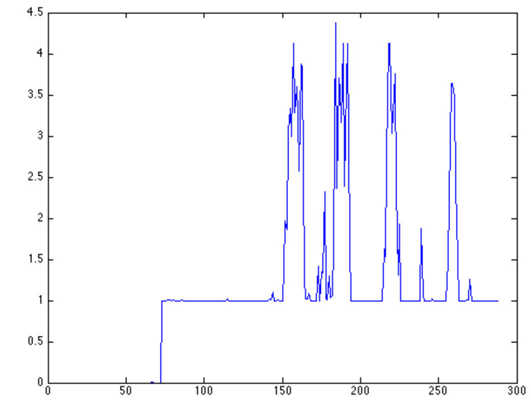


Figure 3.3 Context Transitions of Subject3

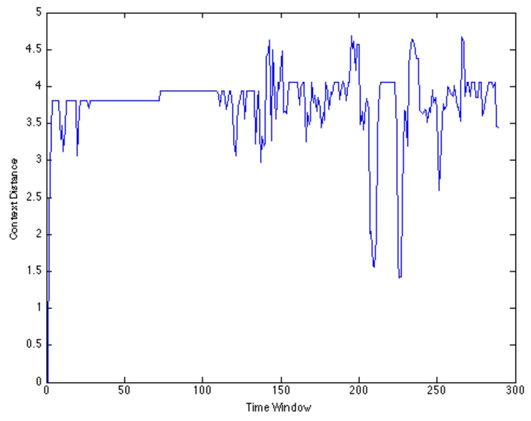


Figure 3.4 Context Transitions of Subject4

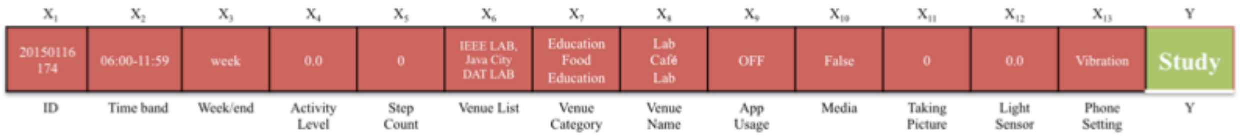
The fluctuations in each plot indicate that life events occurred (i.e. studying, going to class, having lunch, walking, sleeping etc.), maintained and then changed. Though it is difficult to come up with specific names of the life events, I surmise from these figures that the same y-values from the target *timewindow* indicate the same life event, and different y-values signify the different kinds of events. Therefore, the five minutes-based context analysis will give the solution for extracting one life event from multiple data streams.

3.2 Definition of Life Event

Life events are made up of a collection of low-level heterogeneous data streams called contexts. Combining low-level contexts from multiple sensors and smart phones makes it possible to extract life events occurring over certain period of time ranging from several minutes to hours.

To the best of my knowledge, there has been no general definition of life events in the field of multimedia computing. Therefore, the research [6] in which I was involved made an attempt to define a general meaning of life events. Compare to previous work [6], this research is focused on combining a larger number of heterogeneous data streams of short-term accumulated contexts, such as spatial data, temporal data, sensory data, and entertainment data than those of [6]. Such life events could include “sleeping at home”, “staying at home”, “playing with a phone”, “web surfing”, “watching a movie”, “going to class”, “dining”, and “having lunch” etc.

Given the experiment in Chapter 3.1.1, detecting an event can be achieved by an accumulated contexts matrix of five minutes. Each context $X_1, X_2, \dots, X_{13}$ in Figure 3.5 indicates the surrounding conditions within the minutes, and combines these accumulated contexts to describe the high-level event, named “study”. I apply Formal Concept Analysis



X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	Y
20150116 174	06:00-11:59	week	0.0	0	IEEE LAB, Java City DAT LAB	Education Food Education	Lab Café Lab	OFF	False	0	0.0	Vibration	Study
ID	Time band	Week/end	Activity Level	Step Count	Venue List	Venue Category	Venue Name	App Usage	Media	Taking Picture	Light Sensor	Phone Setting	Y

Figure 3.5 Life Event of a *timewindow*

in detecting processes and try to define relationships between life events and attributes $X_1, X_2, \dots, X_{13}$.

CHAPTER 4. CONTEXT-AWARE MOBILE COMPUTING

A.K Dey et al. [3] defines a context-aware computing system as one that "uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." This definition is most applicable to mobile systems. I suggest that context-aware mobile computing is more reactive to a user's contexts and provides more relational information and/or services to a user on their mobile device. To satisfy these requirements, life events detection in context-aware mobile computing must seek to automate understanding of the semantics of human behaviors and contexts with real-time based processing and minimization of user interventions. In this chapter, the major challenges in automated comprehension of human events are discussed. The challenges are:

1) What kinds of contexts have a decisive influence on events.
2) How we can obtain data on these contexts.
3) How we can process it into manageable data.
4) How we can extract a specific situation from the data.

4.1 Influential Life Logs

To identify what kinds of contexts have a decisive influence on events, collectable data on mobile devices must be first identified. These data can be called a life log. Smart phones can provide a low-level life log, or they can raise the low-level format to a high-level life log by processing it with APIs (i.e. google-play-service, foursquare) and then providing these data. For example, smartphones makes it possible to collect low-level data such as x,y,z values of an accelerometer sensor, the brightness values of a light sensor, latitude and longitude values, temporal values, and some setting options (i.e. wireless connections, network providers, sound setting). These low-level data provide a chance to obtain high-level values such as physical activities (i.e. still, tilting, walking, running, bicycling, in vehicle), activity levels [4], or venue data.

4.1.1 Life Logs Selection

Formal Concept Analysis is an instance-independent algorithm that predicts events without model training [7]. It requires the pre-defined binary relationships between concepts and attributes from a mobile device to build and navigate a concept lattice. Therefore, ambiguous definitions of relationships can give rise to uncertain life events detections.

Due to the limitations of collectable data in smart phones, all features that can be gathered from the device must be first considered for the concept lattice. In order to obtain

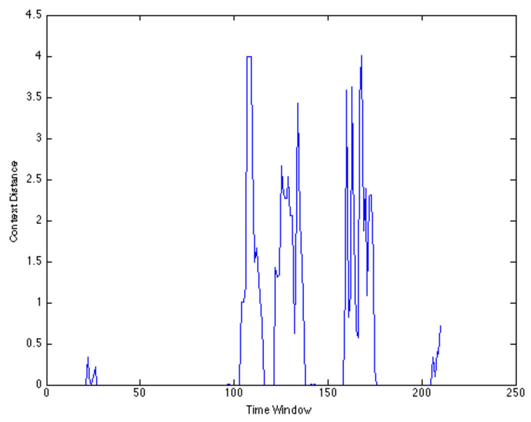


Figure 4.1 Activity Transitions of subject1

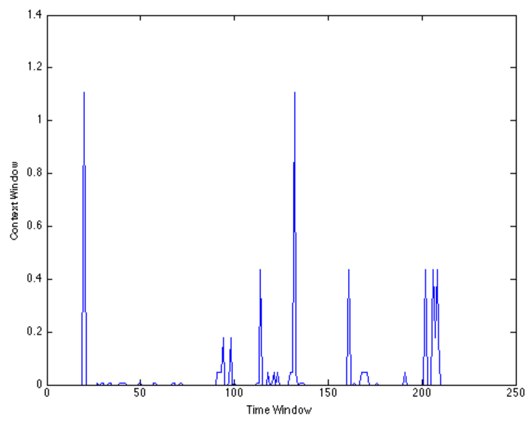


Figure 4.2 App Transitions of subject1

more accurate results, however, processes that evaluate the validity of features and then exclude inappropriate features are required. An experimental validation and heuristic knowledge are together able to help finalize these features. While the experiment of Figures 3.1, 3.2, 3.3, and 3.4 used all collectable features from a smart phone, the results in Figures 4.1 and 4.2 were also gained by only using activity and application usage based on Algorithm 1. The results signify that activity, application usage, and temporal contexts are substantially influential features for the life events of users. From a heuristic perspective, location related features are very significant as well. As demonstrated in the work of Mittal et al. [8] and Seth [9], location features play a key role in deciding events. Hence, to the exclusion of some well-used location categories, I additionally consider the "home" category as a location feature in this work by automatically catching a user's home location and trying to detect home-related life events. Like the home category, light sensor values, media usage, the number of pictures taken, and the sound setting from the smart phone will have an impact on our understanding of a user's real-time situation. However, data from calendar applications are excluded here. The work in [9] substantially depended on the calendar data to detect life events such as "meeting", or "attending class/seminar". I have strong doubts about the credibility of the calendar data due to people's different use habits of the calendar. Table 4.1 shows life logs that are used to detect life events in this paper.

4.2 Life Log Collection

Automated data collection without user intervention is required for life events detection using real-time based, context-aware mobile computing. Extending the *smartNoti* framework of the previous work [4] will enable this automated data collection. The logging layer of *smartNoti* is separated from the architecture of the notification system [4] entirely, and *smartNoti* is now specialized to collect the context factors listed in Table 4.1. Venue list, venue category, venue name, brightness, and number of photos taken are added to the new version of the data collector.

Table 4.1 Selected Life Logs

Sensor	**Device**	**Attribute**	**Description**
Accelerometer	Mobile Phone	Physical Activity	Standing still, tilting, walking, running, bicycling, in vehicle
		Activity Level	The average score of physical activities
		Step Count	The number of steps
GPS	Mobile Phone	Latitude	e.g. 33.64364
		Longitude	e.g. -117.841286
		Venue List	Java City Kiosk
		Venue Category	Food
		Venue Name	Café
Light	Mobile Phone	Brightness	0 – 1000
Mobile Data	Mobile Phone	Time Band	Dawn, morning, afternoon, evening
		Week/end	Week or weekend
		Application Usage	Application usage list
		Media Usage	True or false
		Taken Photos	The number of taken photos
		Sound Setting	Silence, vibration, bell
		Unix Long Time	1421275220009

This newly developed data collector is continuously run in the background of an android platform, collects life logs, processes the low level data, and sends them to a back-end server every 5 minutes frequency. For data regarding the battery consumption of a smart phone and the lack of high-level information in the collected data, the data collector provides an environment to join different types of APIs (i.e. google-play-service, foursquare), which are optimized for these issues, control the battery usage, and create high-level features.

4.2.1 Automatic Home Detection

As one feature of the location category, “home” plays an important role in detecting life events such as “sleeping at home” or “staying at home”. To pursue an automated awareness system, the data collector does not request a user’s manual intervention for home detection, but it does utilize at least five days’ worth of user logs. Algorithm 2 suggests one way to detect a user’s home location. It does not counteract all variables that influence the user’s home location such as traveling and working at night, but rather focuses on general cases and then tries to identify whether the system can automatically detect the user’s home.

Algorithm 2 Home Detection Algorithm

```
Input: $x_i, y_i$
          // $x_i$ is $latitude_i$ and $y_i$ is $longitude_i$
          // $1 \le i \le 5 \times 288$ , 5 days data
Output: $x, y$   // home GPS
 1: if the number of GPS data > $5 \times 288$ // 5 days data
 2:     Load 5 days' GPS data;
 3:     Extract data between $13 < i < 84$;
        // Data between 1:00 am and 7:00 am
 4:     Cluster data at most 5 groups;
 5:     Vote the largest cluster;
 6:     Randomly pick one set of $(x, y)$;
 7:     return $(x, y)$;
 8: else
 9:     return null;
10: endif
```

5×288 of Algorithm 2 represents that we collected five days worth of data and that data on events are collected in five minutes intervals. I heuristically choose a five day period and the time range 1:00 am to 7:00 am. The system selects, at most, five different places between these times and chooses the largest cluster among them, which identifies the most frequently visited places. The method used to cluster five different groups is similar to the hierarchical clustering algorithm (i.e. Agglomerative). First, the distances between all GPS points are calculated. Given that the hand-held GPS device is accurate, I set the clustering point 50 meters apart from each cluster. The distance is calculated based on a basic formula in [10], which uses the haversine formula to calculate the grate-circle distance between two points, where R is earth's radius, ϕ is latitude, $\leftthreetimes$is longitude, and a is the haversine formula.

$$R = 6371000, \phi_1 = latitude_{1,} \ \phi_2 = latitude_2 \tag{1}$$

$$\triangle \phi = latitude_1 - latitude_2, \triangle \leftthreetimes = longitude_1 - longitude_2 \tag{2}$$

$$a = sin^2(\triangle \phi \div 2) + cos\phi_1 \times cos\phi_2 \times sin^2(\triangle \leftthreetimes \div 2) \tag{3}$$

$$c = 2 \times \arctan\left(\sqrt{a}\sqrt{1-a}\right), d = R \times C \tag{4}$$

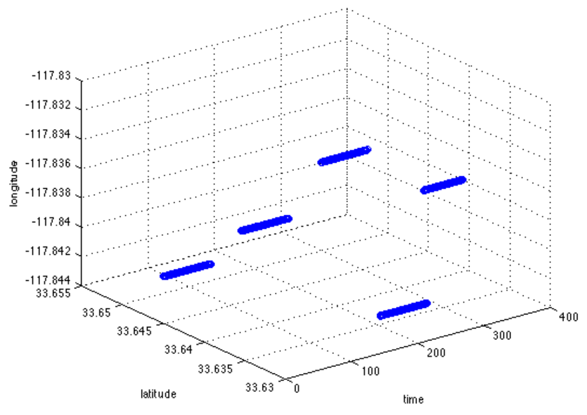


Figure 4.3 GPS Distribution of Subject 1 at between 1:00 am and 7:00 am

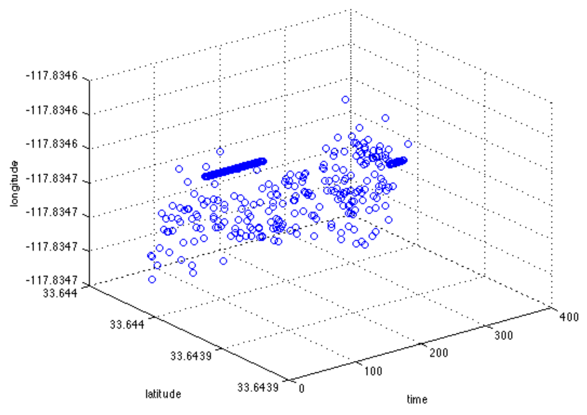


Figure 4.4 GPS Distribution of Subject 2 at between 1:00 am and 7:00 am

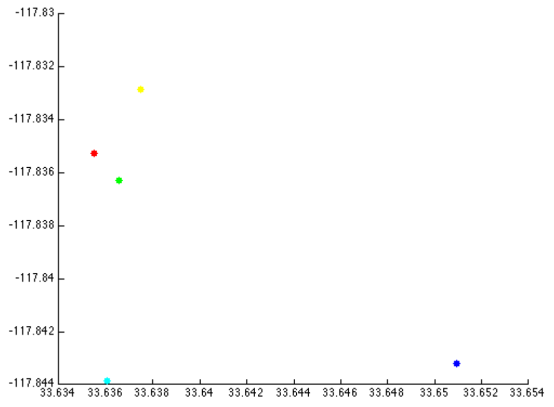


Figure 4.5 Clustered GPS of Subject 1 at between 1:00 am and 7:00 am

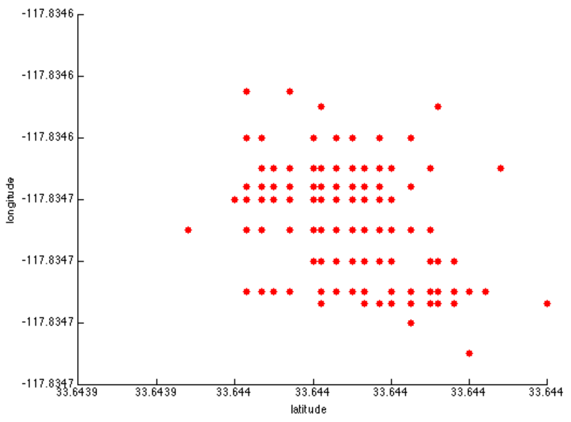


Figure 4.6 Clustered GPS of Subject 2 at between 1:00 am and 7:00 am

Figures 4.3 and 4.4 show the distribution of GPS points between 1:00 am and 7:00 am over five days. I assume that one group of points in each figure might be a users' home GPS. Though the above figures show widely spread GPS points, these points might actually be included into the same cluster. Therefore, given variation in GPS accuracy or user's life patterns, Algorithm 2 can help to detect one specific home GPS.

Figures 4.5 and 4.6 are the clustering results for Algorithm 2. Each color in Figures 4.5 and 4.6 indicates the groups. While subject2 shows only one cluster of GPS points, subject1 shows five GPS point clusters. The former result is because of the variation in GPS accuracy and the latter result is from the user's life patterns at these times. However, with voting being largest cluster, the system can get one correct home GPS from Figure 4.5. In the case of Figure 4.6, the system randomly selects any one of them.

In this paper, the purpose of home detection is to get one important feature of location categories and then to check how well the system can detect home-related life events. Therefore, considering other variables that help identify the user's home location are excluded, but will be handled in future work.

4.3 Context Matrix, "*timewindow*"

The processing of collected life logs into a manageable data format can aid in life events detection. The context matrix, called "*timewindow*", enables unstructured heterogeneous data to have a structured format and then enables of them to be analyzed in a chronological way. This is possible by segmenting each collected data streams into five minutes intervals and then building *timewindow*. Figure 3.5 shows one labeled *timewindow* that is arranged by chronological order. Each piece of the datastreams are made up of one *timewindow* per five minutes interval, and these context matrices facilitate the handling, comparing, and analyzing of user's situations with each other. In general*, timewindow* is a 1 X N matrix, which divides a day into 288 windows. Processed context data assembles as one matrix every five minutes and then it is analyzed (i.e. labeling).

4.4 Methods for context awareness

Given the limitations of an environment with variable numbers of subjects, privacy issues and the challenges of personal data, the difficulty lies in training a generalized statistical model and predicting outcomes with machine learning techniques. Therefore, Formal Concept Analysis can be an alternative means of making a general model and identifying specific situations from life logs.

4.4.1 Formal concept analysis

Formal Concept Analysis was introduced by Wille et al. [11] in 1982. They insisted that FCA organizes structural similarities from relationships between an object and its attributes as partially ordered sets, and builds a graphical representation of knowledge, called a Concept Lattice, to visualize the structure and then navigates it to retrieve a result. Therefore, FCA does not need to use any statistical calculations, but rather uses the structural similarities to produce results, even in uncertainty. Given the limitations of collecting life logs for a personalized statistical model, as noted above, it is still possible to efficiently predict some life events. In this chapter, the basic definitions of Formal Concept Analysis and its application to life events detection are discussed.

A context is represented by a triplet $T = (O, A, R)$, where O is a set of objects, A is a set of attributes, and a relation or incidence R is $R \subseteq O \times A$. Each node in the Concept Lattice is a pair (X, Y), where $X \subseteq O$ is called extension and $Y \subseteq A$ is called intension. Each pair satisfies a relation. i.e. :

$$X = \{x \in O \mid \forall y \in Y, yRx\} \tag{5}$$

$$Y = \{y \in A \mid \forall x \in X, yRx\} \tag{6}$$

A set of these pairs is called the concept. Between concepts, partial order is defined as in (7). The Concept Lattice is now drawn as a Hasse Diagram, which has an order-relation as in (7).

$$(X_1, Y_1) \le (X_2, Y_2), if\ X_1 \subseteq X_2 \iff Y_1 \supseteq Y_2 \tag{7}$$

Table 4.2 Sample Cross Table of Life Events and Attributes

	Location				Physical Activity				Time			
Attributes / Life Events	Home	School	Park	Restaurant	Still	Walking	Running	Bicycling	Dawn 00:00-06:00	Morning 06:00-12:00	Afternoon 12:00-18:00	Night 18:00-24:00
Sleeping	X				X				X	X		
Staying at home	X				X						X	X
Studying		X			X					X	X	X
Exercising			X			X	X	X	X	X	X	X
Dining				X	X							X

To apply Formal Concept Analysis to life events detection, binary relationships between objects and attributes must be first identified. Specifically, I set objects and attributes to life events and life-logs, respectively, in the FCA. For example, Table 4.2 describes the simplified relationships of life events and their attributes. An empty cell shows that the corresponding life event does not have that attribute and a cross indicates that the life event has the attribute.

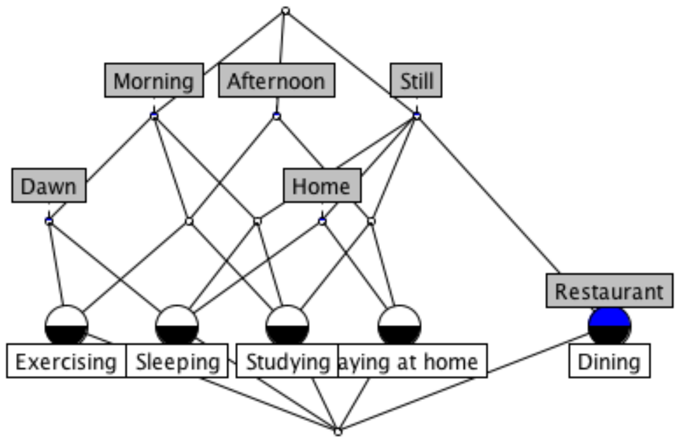


Figure 4.7 Concept Lattice Obtained for Table 4.2

With these relationships and the partial order relation (7), the Concept Lattice can be constructed as in Figure 4.7. The Concept Explorer software [12], which is a tool for data analysis based on Formal Concept Analysis, provides a function to draw this Concept Lattice. This figure shows not only the partially ordered concept pairs, but also the generic and specific concept relations in the lattice as well. According to the relation (7), a concept closer to the top indicates that it is more specialized, (i.e. closer to a specific object), and the opposite direction shows that it is more generalized (i.e. has more attributes in a concept pair (X, Y)). From these relations in the lattice, a method for detecting a life event can be inferred.

Algorithm 3 Algorithm to navigate a Concept Lattice

```
Input: incoming attributes $Y_n$, Concept Lattice $CL$
        // $n$ is the number of attributes from a smart phone
Output: Possible Life Events or Unknown Event
 1: $Count$ = all possible concepts of CL;
 2: $detectedEvent$ = null;
 3: for each concept pairs $(X, Y) < Count$
 4:     if $Y \subseteq Y_n$
 5:         $detectedEvent = X$;
 6:         return $detectedEvent$;
 7:     endif
 8: endfor
 9: if $detectedEvent ==$ null
10:     return "Unknown Event";
11: endif
```

An algorithm in [7], which is used to navigate the lattice for activity query, can help to detect a life event in the Concept Lattice. I applied the algorithm to my data to navigate events and life log pairs. With the constructed Concept Lattice and incoming attributes from a smart phone, the system navigates across all nodes and starts searching for a life event. The backtracked depth first search is used to traverse the lattice. Figure 4.8 shows the root in detecting “Sleeping” when attributes “Home”, “Still” and “Dawn” are given. Algorithm 3, however, only can find a completely matched life event. Therefore, it does not infer any events when certain attributes, such as “Home”, “Still”, and “Afternoon” are given. In this case, the algorithm would returns “Unknown Event”. For improving the detection accuracy and diversifying the life events detection of Formal Concept Analysis, more concrete attributes must be defined and a more detailed discretization of these attributes is necessary.

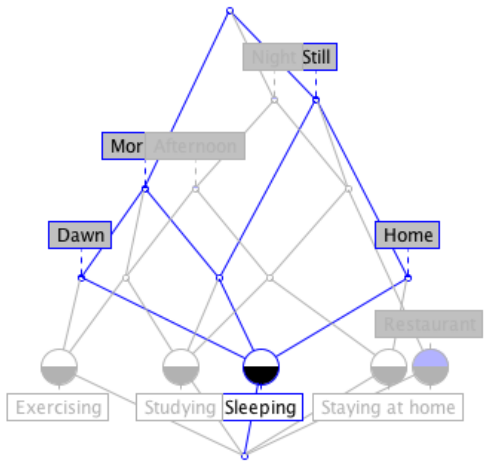


Figure 4.8 Lattice Navigation for Life Event Detection

4.4.2 Discretization

As discussed in Chapter 4.4.1, Formal Concept Analysis explicitly assumes that relationships between objects and attributes are binary, whereas some attributes among collected life logs, such as activity level, brightness, application usage, and taken photos are continuous values. Therefore, I use a symbolic representation of time series, SAX, which allows for distance measures to be defined by a symbolic approach that lowers the bound of the corresponding distance measures defined in the original series [13] and tries to convert these continuous values to discrete values.

Once continuous data are transformed to Piecewise Aggregate Approximation (PAA), SAX can produce more discrete presentations. The first condition for easily achieving the discretization technique, PAA, assumes that normalized continuous values will have a Gaussian distribution [14].

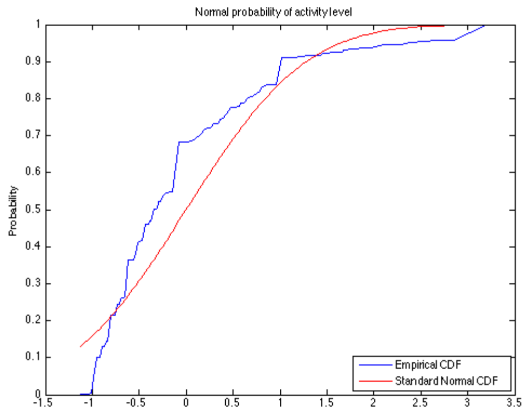


Figure 4.9 Normal Probability Plot of Cumulative Distribution of Activity Level

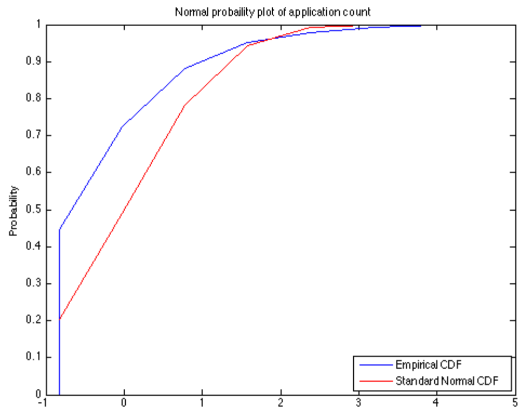


Figure 4.10 Normal Probability Plot of Cumulative Distribution of App Count

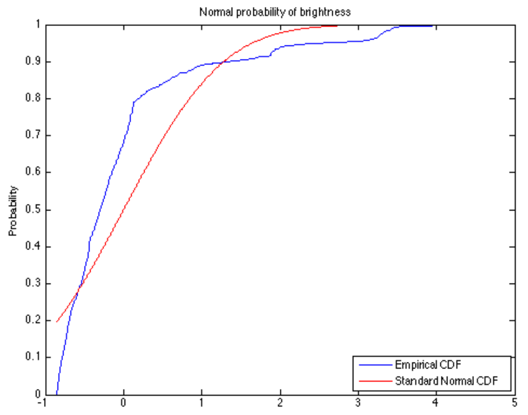


Figure 4.11 Normal Probability Plot of Cumulative Distribution of Brightness

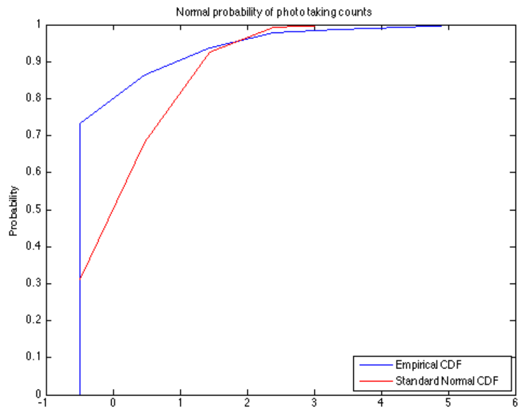


Figure 4.12 Normal Probability Plot of Cumulative Distribution of Taken Photo

Figures 4.9, 4.10, 4.11 and 4.12 show the normal probability plots of the cumulative distribution of each value. The highly linear nature of the plot indicates that the data follows a Gaussian distribution. Though the figures above do not exactly show their linearity due to their small amount of data, one month worth of data, I assume that these continuous features will finally follow a Gaussian distribution [26]. Lin et al. [13] define a "breakpoint as a sorted list of numbers $B = \beta_{1,\ldots,}\beta_{a-1}$ such that the area under a $N(0,1)$ Gaussian curve from β_i to $\beta_{i+1} = 1/a$ where a is equal sized areas under Gaussian curve, and β_0 and β_a are defined as $-\infty$ and ∞." Table 4.3 is a lookup table that showing the division a Gaussian distribution by breakpoints. It gives breakpoints that can divide a Gaussian distribution for values of a between 3 to 10 [13].

β_i \ a	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
β_1	-0.43	-0.67	-0.84	-0.97	-1.07	-1.15	-1.22	-1.28
β_2	0.43	0	-0.25	-0.43	-0.57	-0.67	-0.76	-0.84
β_3		0.67	0.25	0	-0.18	-0.32	-0.43	-0.52
β_4			0.84	0.43	0.18	0	-0.14	-0.25
β_5				0.97	0.57	0.32	0.14	0
β_6					1.07	0.67	0.43	0.25
β_7						1.15	0.76	0.52
β_8							1.22	0.84
β_9								1.28

Table 4.3 A Lookup Table that Contains the Breakpoints to Divide a Gaussian Distribution for Value of a from 3 to 10

$$level = the\ normalized\ result\ of\ a\ continuous\ value \tag{8}$$

$$mean = average\ of\ a\ continuous\ value \tag{9}$$

$$std = standard\ deviation\ of\ a\ continuous\ value \tag{10}$$

$$PAA = \frac{(level - mean)}{std} \tag{11}$$

Once the breakpoints are obtained as in Table 4.3, the PAA of each continuous value can be calculated by (11). These PAAs match up with the range of Table 4.3. For example, assuming that the chosen number of discretized ranges is three, a PAA coefficient will find where it can be included between $PAA < -0.43, -0.43 < PAA < 0.43$, and $0.43 < PAA$.

Activity Level a = 5	**Application Count** a = 4	**Brightness** a = 3	**Taken Photo** a = 4
Motionless	No Use	Low	No Use
Low	Low	Medium	Low
Medium	Medium	High	Medium
High	High		High
Very High			

Table 4.4 Discretized Ranges of Continuous Values

Considering the influence of activity level and the types of physical activity, such as standing still, walking, running, bicycling, or a vehicle, in context transitions, I give a weight more on activity level by subdividing it into five parts. In the case of brightness, the range "No Use" is excluded as the light sensor is continually turning on. To give less weight than the activity level, I divided application count and taken photo into four parts. Based on the breakpoints in Table 4.3 and the discretized ranges in Table 4.4, Figures 4.13, 4.14, 4.15 and 4.16 show the results of first the five samples of continuous values by the SAX discretization.

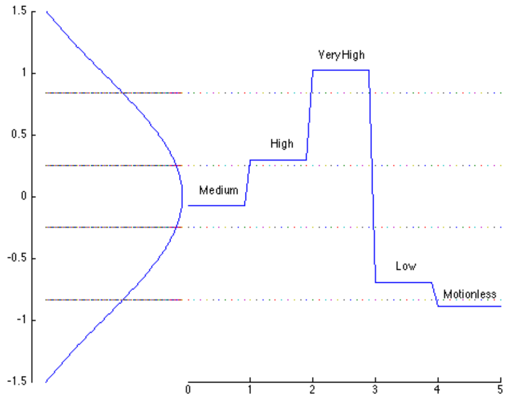


Figure 4.13 Discretized Activity Level

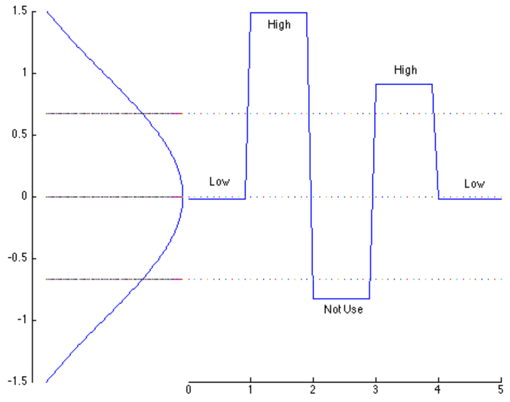


Figure 4.14 Discretized App Count

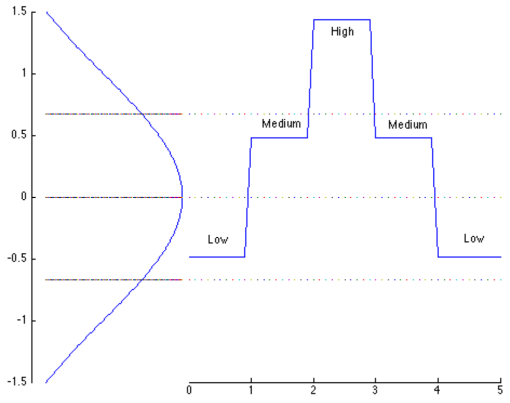


Figure 4.15 Discretized Taken Photo

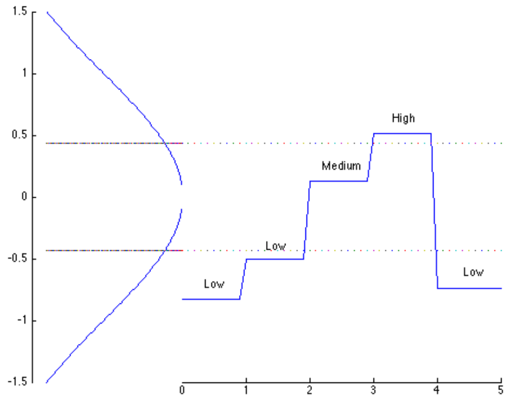


Figure 4.16 Discretized Brightness

CHAPTER 5. LIFE EVENT DETECTION

In this chapter, I introduce a systematic approach to detect life events. For real-time based processing and minimization of user interventions, this system covers both front-end and back-end of the life event detection, and interacts with each in real time. Therefore, the system can be more closely reactive to a user's contexts and identifies more relational information to detect life events on their mobile device. To follow the requirements of context-aware mobile computing, discussed in chapter 4, selected life logs are collected, and the "home" category of location features are automatically detected and then added to them. The system generates a *timewindow* every five minutes based on collected life logs, and identifies a life event in the *timewindow*, by applying SAX discretization and Formal Concept Analysis.

Life events can generally be divided into two types; past independent events and past dependent events. Past independent events can be directly detected by using only one *timewindow*, but past dependent events must consider past life logs and find some links between the past and present. Table 5.1 shows some example events.

Past independent events		**Past dependent events**	
Standing Still	Walking	Watching a Movie	Exercising
Running	Bicycling	Lunch	Dining
In vehicle	Studying	Shopping	Toileting
Playing with a Phone	Sleeping	Traveling	
Staying at home	Web Surfing		
Chatting	Meeting		

Table 5.1 Past Independent Events and Past Dependent Events

ID	Time band	Week/end	Activity Level	Activity Type	Step Count	Venue Category	App Count	Media Usage	Brightness	Taken Photo	Sound Setting
1	afternoon	week	0.0	still	0	education	0	false	32.71	0	silence
2	night	week	0.8	still	78	building	1	false	239	0	bell
3	afternoon	week	0.0	still	0	education	0	false	4.221	0	silence
4	night	week	1.66	run	532	building	1	true	6.03	0	bell

Table 5.2 Life Log Samples

The first row in Table 5.2 shows that a user is staying at school without movement and not using a smart phone at that moment. This may signify that the user is now studying or doing something study related. However, the second row data do not intuitively indicate a life event. Though it shows some step counts, activity type is standing still, which means the user had worked within the last five minutes but their current status is standing still. It also shows that now the user is in a building and the mobile phone's sound setting is bell. Therefore, the user might be in the toilet because of the activity patterns, but it is also possible that the user is in an elevator or has just arrived at an office. While the second sample needs some past data to find certain clues for a current life event, the first sample can predict that the user is doing some study related works. In this thesis, I try to focus on detecting past independent life events, such as the first row in Table 5.2, and to identify whether a general life events model can apply to each individual. To help this process, a rule based filtering mechanism is also necessary.

Standing Still	Walking
Running	Bicycling
In vehicle	Studying
Playing with a Phone	Web Surfing
Sleeping at Home	Sleeping at Other Place
Staying at Home	

Table 5.3 Candidates of Detectable Life Events

Attrubutes	Time						Activity										Location											Sensor			Entertainment										Setting	
	Time Band				WEEK		Physical Activity					Activity Level					Category											Light			Media		App Usage				Photo				Sound	
Life Events	Dawn(0)	Morning(1)	Afternoon(2)	Evening(3)	Week(0)	Weekend(1)	Standing Still(0)	Tilting/Unknown(1)	Walk(2)	Bicycle(3)	In vehicle(4)	Motionless(0)	Low(1)	Medium(2)	High(3)	Very high(4)	Arts&Entertainment(0)	Building(1)	Shops(2)	Event(3)	Food(4)	Parks&Ourdoors(5)	Travel(6)	Education(7)	NightLife(8)	Home(9)	Others(10)	Low(0)	Medium(1)	High(2)	No Use(0)	Use(1)	No Use(0)	Low(1)	Medium(2)	High(3)	No Use(0)	Low(1)	Medium(2)	High(3)	Silence/Vibration(0)	Bell(1)
Studying		x	x	x	x		x					x						x						x				x	x	x	x		x				x				x	x
Playing with Phone	x	x	x	x	x	x	x	x				x	x	x			x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			x	x	x	x	x	x	x	x	x
Sleeping at Home	x				x	x	x					x														x		x	x		x		x				x				x	x
Sleeping at Others	x				x	x	x					x					x	x	x	x	x	x	x	x	x		x	x	x		x		x				x				x	x
Staying at Home		x	x	x	x	x	x	x				x	x	x												x		x	x	x	x	x	x				x				x	x

Table 5.4 Cross Table of Detectable Life Events and Attributes

5.1 General Life Event Model

Table 5.3 shows the candidates for detectable life events in this system. I assume that a general life events model can be used to detect each individual's life events as in Table 5.3. Therefore, a cross table, that defines relationships between objects and attributes, Table 5.4, is first defined as the general model. This table was built by three subjects' two months worth of data.

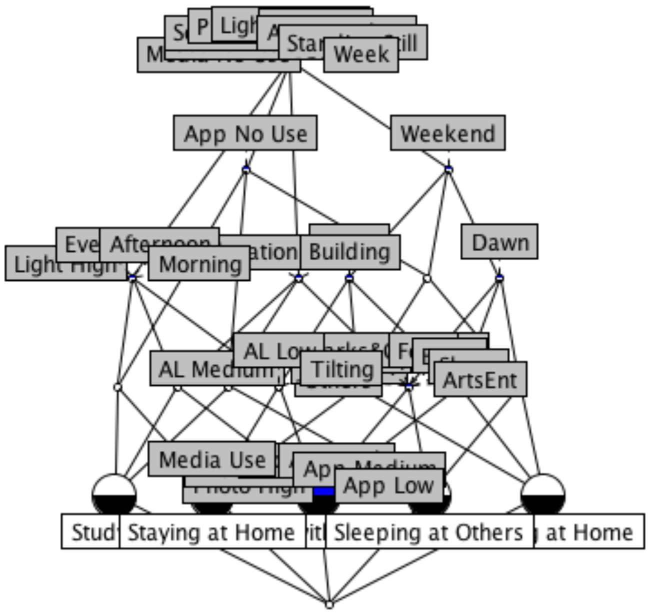


Figure 5.1 Concept Lattice Obtained for Table 5.4

Finally, a concept lattice can be constructed by the cross table as in Figure 5.1. This lattice contributes detecting the five kinds of past independent life events, such as “studying”, “playing with a phone”, “sleeping at home”, “sleeping at other place”, and “staying at home”. Once the concept lattice is constructed as the general model in the back-end server, the system starts navigating the lattice with incoming attributes and returns the result to the front-end smart phone.

5.2 Rule-based filtering

Rule-based filtering is a method that can help to filter life events based on pre-defined rules. Specifically, it is useful in detecting straightforward events and hierarchical events before or after navigating a concept lattice. Straightforward events are signified by “standing still”, “walking”, “running”, “bicycling”, and “in vehicle” in Table 5.3. Because life event detection identifies one event for every five minutes, these events are hard to detect if the system only uses a concept lattice for the detection process. For example, if a user walked, ran, and rode a bicycle within that time, the concept lattice cannot identify what event is the major one. “Web surfing” is an example of a hierarchical event. To detect this sort of a specific event, the cross table must have not only the application usage count, but also the application categories. However, it is difficult to categorize used applications into specific named groups because of the lack of descriptive information in the android applications. If such information were available, it would be possible to distinguish between simultaneous hierarchical events, such as between “playing with a phone” and “surfing the web”. Rule base filtering can help to distinguish between these events and to avoid unnecessarily time-consuming lattice navigation.

Algorithm 4 Algorithm for Rule-based Filtering

```
Input: incoming attributes $Y_n$
        // $n$ is the number of attributes from a smart phone
Output: Possible Life Events or Unknown Event
 1: $Array = getActivityType(Y_n)$;
 2: $Array$ = desc order by activity level;
 3: if $Array[0]$ == still or tilting or unknown // rule 1.
 4:     // navigate a concept lattice
 5:     $detectedEvent = Algorithm$ 3;
 6:     if $detectedEvent$ == "playing with a phone" // rule 2.
 7:         Check the package name of foreground applications;
 8:         if package name == internet related applications
 9:             return "web surfing";
10:         else
11:             return "playing with a phone";
12:         endif
13:     else $detectedEvent$ ;
14:     endif
15: else  // rule 3.
16:     return $Array[0]$;
17: endif
```

Algorithm 4 shows the rule-based filtering of life events detection. The navigation algorithm of a concept lattice is also contained in this mechanism. The system first retrieves incoming attributes, sorts activity types out, and then orders these types by their activity level. Then, the system checks the first rule "navigate a given concept lattice when the major activity type is "standing still", "tilting", or "unknown"". After that, the system checks the second rule "identify whether the foreground application is an Internet related one (i.e. google, browser, safari, chorm, firefox) when the detected event is "playing with a phone"". If the second rule does not satisfy the conditions, the system immediately returns a first ordered activity, such as "walking", "running", "bicycling", or "in vehicle" according to the third rule.

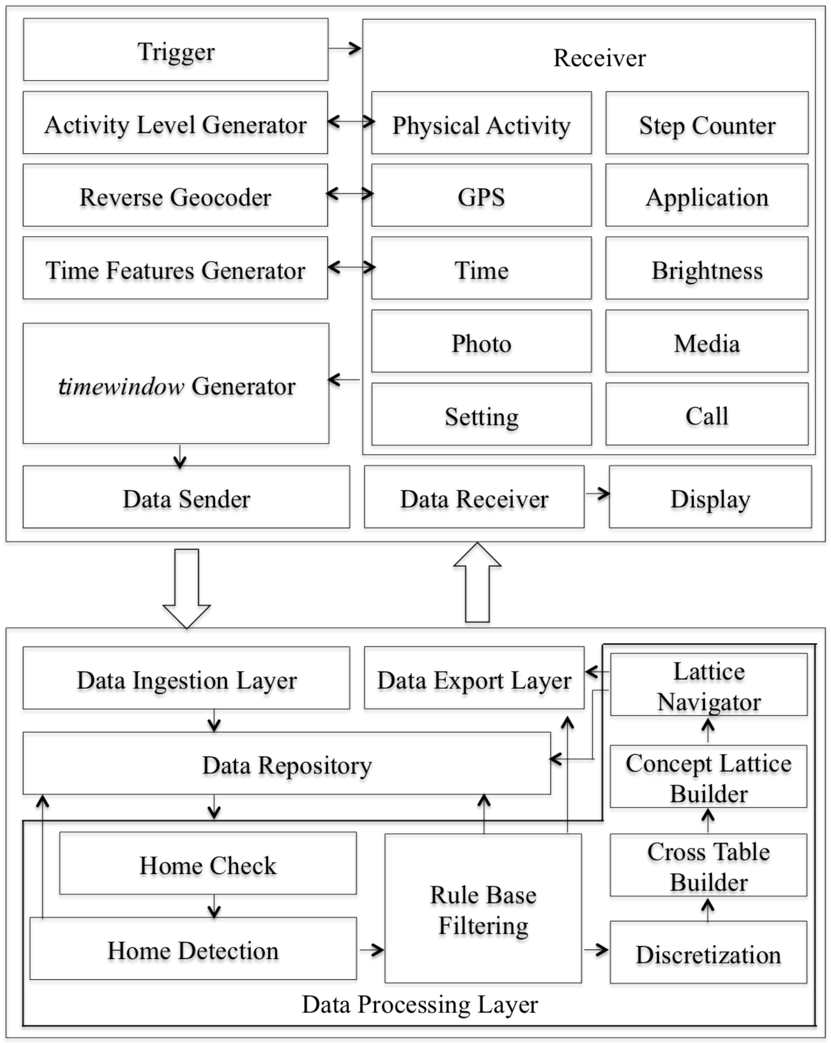


Figure 5.2 System Architecture; upper architecture is front-end, bottom architecture is back-end system

5.3 Architecture

Figure 5.2 shows the architecture of the life event detection system. The front-end architecture is entirely separated from the architecture of *smartNoti* [4] and manages the data collection, interaction with back-end, and event display. The system back-end processes the collected data, performs the event detection process, and returns a final result to the front. These two parts communicate with each other every five minutes. Figure 5.3 shows the entire system flow.

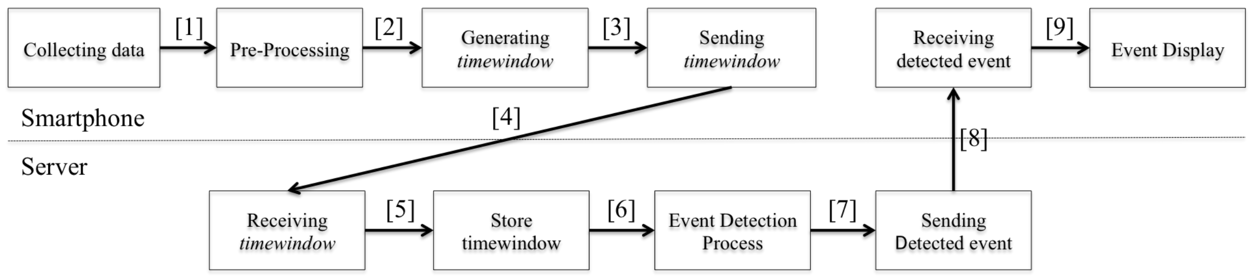


Figure 5.3 System Flow

5.3.1 Front-end System

The front-end contributes to two main tasks: data logging and event display. The logging part integrates six different categories: 1) activity (physical activity and number of steps), 2) location (latitude, longitude, venue list, venue category and venue name), 3) time (week, weekend, or dawn, morning, afternoon, evening), 4) logical (calendar), 5) entertainment (audio, video, application, photo), and 6) system (silence, vibration, bell). Each attribute has own receivers and these receivers run in the background in real-time.

To obtain more high-level data in the front-end, physical activity, GPS, and time receivers go through one more step. The physical activity receiver connects the ActivityRecognition API of the google-play-service whenever accelerometer sensor values indicate different patterns of x,y,z, sends the accelerometer data, and then receives the physical activity types from the API. These received activity types are converted to numeric values according to their intensity, and then calculated as an average of the accumulated physical activities of all five minutes range in the activity level generator. The GPS receiver connects the foursquare API with current latitudinal and longitudinal data every five minutes. It then immediately receives venue list, venue category, and venue name from the

API. The time receiver generates time-related features, such as week/weekend, time band, and unix long time with the time features generator.

The collected data for each receiver are transmitted to the data ingestion layer of the back-end side through the data sender. The data sender sends the data as a formulaic type, *timewindow*, from the *timewindow* generator. The *timewindow* generator reorganizes the collected data every five minutes and makes the data easy to process in the back-end. The data receiver receives a detected life event from the data export layer of the server, and then this event is displayed on its user's smartphone.

5.3.2 Back-end System

The data ingestion layer receives a *timewindow* every five minutes from the front-end. Because I designed the *timewindow* as a form of the JSON object, which is used on the web, I set up the data repository by using an open-source document database MongoDB. This database is at the web-scale and is not a fixed-scale schema, and is therefore suitable for reusing the JSON. As a result, the data ingestion layer can wholly insert a *timewindow* into the repository without decomposing the *timewindow*. Figure 5.4 shows a database sample.

The data processing layer tries to detect a life event in a *timewinow*. First, it checks whether the database has the home GPS. After checking the home data, it now checks the pre-defined rules. According to these rules, as in chapter 5.2, the layer can immediately

```
{
        timewindow: {
                personicle_id: “MAC ADDRESS”,
                timewindow_id: “20150429_201”,
                time_band: 3,
                long_time: 1429314020,
                week: 0,
                type: “unknown”,
                activity_level: 0.0,
                activity_type: ”{standing still=6}”,
                step_count: 0,
                latitude: 33.6436417,
                longitude: -117.8413306,
                venue_list: “[UCI IEEE Lab, DAT Lab, Java City Kiosk]”,
                venue_category: “[education, education, food]”,
                venue_name: “[Lab, Lab, Café]”,
                app_count: “[com.android.launcher]”,
                app_duration: “[8]”,
                media: 0,
                light: 142.92592592592592,
                photo: 0,
                setting: 0,
                }
}
```
Figure 5.4 Database Sample; time_band (0: dawn, 1: morning, 2: afternoon, 3: evening), week (0: week, 1: weekend), media (0: false, 1: true), setting (0: bell, 1: silence, 2: vibration)

catch the physical activity events in the *timewindow* or continue the process of the formal concept analysis. The data export layer helps to send these results.

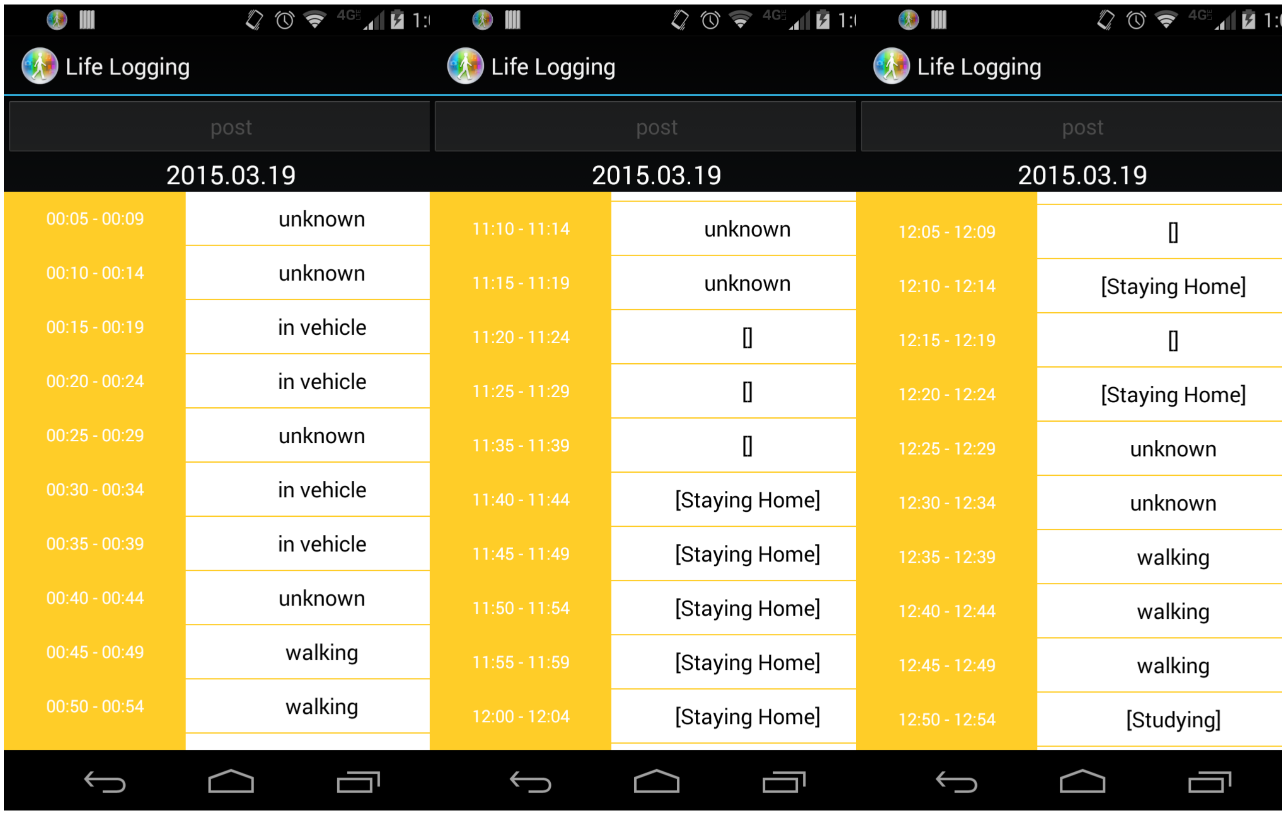
CHAPTER 6. EXPERIMENTAL VALIDATION

To demonstrate that Formal Concept Analysis is useful for automatically detecting past independent life events and to show whether a general life events model can apply to each individual in a sample, I implemented the system suggested in chapter 5 on both an Android smartphone and a server. As in longitudinal study, I carried out user tests for about two months with three people who installed the application.

6.1 Implementation

The life event detection system is optimized for Android 4.4. The Android prototype has been tested on a Samsung Galaxy S4, Motorola Moto x (Gen 1), and LG G pro2. This front-end system runs in the background in real-time, collects life-logs, interacts with the back-end server, and shows the timeline of detected life events. Due to the API's connection for high-level data, test users must always be connected to the network (i.e. 4G, LTE, wireless) and must have the "location" turned on in the phone's setting. The back-end server has been constructed on Ubuntu 12.04.3 and reacts to front-end requests in real-time.

All detection processes start running automatically with the installation of the android application. Detected events are shown in the form of a timeline in listview within an Android application. Figure 6.1 shows the implementation of the life event detections in an android smartphone. If the cross table does not contain a relationship between incoming attributes and pre-defined events at all, the system returns an empty event, "[]", and if the structural patterns of incoming attributes are similar to the pre-defined relationships, but the concept lattice still cannot find one converged event, the system displays “unknown”.


Life Logging

post

2015.03.19

00:05 - 00:09	unknown
00:10 - 00:14	unknown
00:15 - 00:19	in vehicle
00:20 - 00:24	in vehicle
00:25 - 00:29	unknown
00:30 - 00:34	in vehicle
00:35 - 00:39	in vehicle
00:40 - 00:44	unknown
00:45 - 00:49	walking
00:50 - 00:54	walking

Life Logging

post

2015.03.19

11:10 - 11:14	unknown
11:15 - 11:19	unknown
11:20 - 11:24	[]
11:25 - 11:29	[]
11:35 - 11:39	[]
11:40 - 11:44	[Staying Home]
11:45 - 11:49	[Staying Home]
11:50 - 11:54	[Staying Home]
11:55 - 11:59	[Staying Home]
12:00 - 12:04	[Staying Home]

Life Logging

post

2015.03.19

12:05 - 12:09	[]
12:10 - 12:14	[Staying Home]
12:15 - 12:19	[]
12:20 - 12:24	[Staying Home]
12:25 - 12:29	unknown
12:30 - 12:34	unknown
12:35 - 12:39	walking
12:40 - 12:44	walking
12:45 - 12:49	walking
12:50 - 12:54	[Studying]

Figure 6.1 Examples of the Life Event Detection in the Android Application

For future works, I added a self-reporting function that can be used to provide a corrected life event if the event is wrong. An analyst can click each row of the listview and select a correct event as shown in Figure 6.2, and then the Android application sends the result to the server, and finally the server adds the result to the database. This information will be used to analyze past-dependent events, such as “shopping”, “Dining”, “Watching a movie” etc.

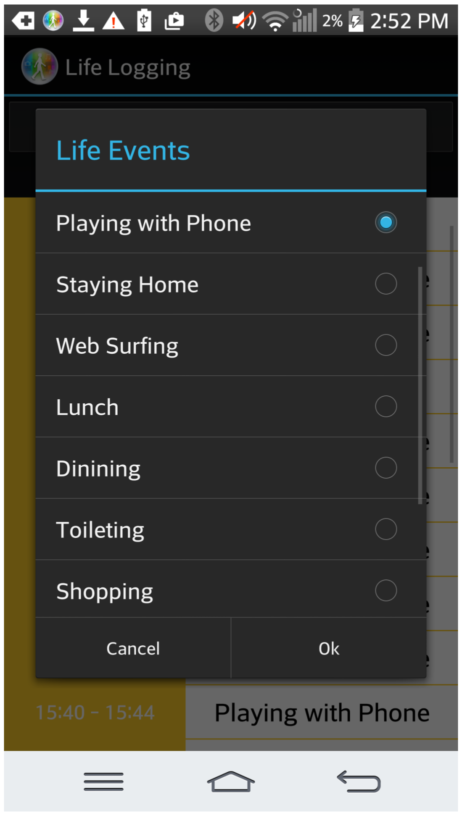


Figure 6.2 Self-correction of the Wrong Detected Life Event

6.2 Evaluation method and experimental results

To demonstrate the performance of the automated life events detection system, I first evaluate the applicability of FCA in terms of the accuracy of life event prediction. Also, I evaluate a generalized life events model of FCA with one-week worth of randomly selected data, and then I determined whether the model could be applied to each individual. I exclude the straightforward events and hierarchical events detection in these experiments, since the prediction algorithms do not affect them. The straightforward events can only be retrieved by google-play-service (ActivityRecognition) and the hierarchical events need preceded detection process, such as “playing with a phone.”

		Total	Prediction	Accuracy
FCA	Playing with a Phone	1030	1030	1
	Sleeping at Home	1453	1453	1
	Sleeping at Other Places	277	277	1
	Staying at Home	2579	2579	1
	Studying	3631	3631	1
Linear Classifier	Playing with a Phone	1030	700	0.6796
	Sleeping at Home	1453	1453	1
	Sleeping at Other Places	277	193	0.6968
	Staying at Home	2579	2579	1
	Studying	3631	2967	0.8171
KNN Classifier	Playing with a Phone	1030	804	0.7806
	Sleeping at Home	1453	1447	0.9959
	Sleeping at Other Places	277	275	0.9928
	Staying at Home	2579	2568	0.9957
	Studying	3631	3595	0.9901
Decision Tree	Playing with a Phone	1030	1030	1
	Sleeping at Home	1453	1453	1
	Sleeping at Other Places	277	277	1
	Staying at Home	2579	2579	1
	Studying	3631	3631	1
Random Forest	Playing with a Phone	1030	1030	1
	Sleeping at Home	1453	1453	1
	Sleeping at Other Places	277	277	1
	Staying at Home	2579	2579	1
	Studying	3631	3631	1

Table 6.1 Prediction Accuracy of Each Life Event by FCA, Linear Classifier, K-Nearest-Neighbor Classifier (K=1), Decision Tree, and Random Forest (Bags = 50)

Accuracy test. For showing that Formal Concept Analysis can substitute machine learning techniques in detecting life events, I compare FCA to other benchmark supervised learning algorithms, such as linear classification, knn classification (k=1), decision tree and bagged decision tree (bags = 50). The main purpose of this experiment is to show that FCA is applicable for life event detection. Therefore, the optimizations of machine learning algorithms are not an issue here. Using two months worth of data from three people, I first filter only obvious samples of each life event through knowledge-based scanning. Ambiguous samples are completely excluded from this experiment. Then, all the data is shuffled and divided into training and test sets by a ratio of seventy percent to thirty percent. I extract general relationships between attributes and detectable life events from the training data, make a general cross table for FCA as in Table 5.4, and also train the other four machine learning models with the training data. Finally, trained models and FCA with a cross table are tested by the test data. Table 6.1 shows results of the prediction accuracy of each life event by FCA, linear classifier, k-nearest-neighbor classifier (k=1), decision tree, and random forest (bags = 50).

	Accuracy	**Precision**	**Recall**	**F-Measure**
Person 1	0.7234	0.9614	0.7371	0.8344
Person 2	0.9569	0.9921	0.9482	0.9697
Person 3	0.9251	0.9976	0.9267	0.9608

Table 6.2 Performance of All Three People on Using a Generalized Life Event Model and the Rule-based Filtering

		Total	**Prediction**	**Accuracy**
Person 1	Playing with a Phone	224	209	0.9330
	Sleeping at Home	416	230	0.5529
	Sleeping at Other Places	0	0	1
	Staying at Home	211	197	0.9336
	Studying	332	245	0.7379
	Unknown	83	47	0.5662
Person 2	Playing with a Phone	41	41	1
	Sleeping at Home	335	334	0.9970
	Sleeping at Other Places	0	0	1

	Staying at Home	631	601	0.9526
	Studying	192	175	0.9115
	Unknown	103	95	0.9223
Person 3	Playing with a Phone	30	30	1
	Sleeping at Home	436	346	0.7936
	Sleeping at Other Places	0	0	1
	Staying at Home	810	810	1
	Studying	75	66	0.88
	Unknown	11	8	0.7272

Table 6.3 Accuracy of Detected Life Events of Each Person

Applicability test. After the proof of availability of FCA in life event detection, now, I perform an applicability test with one-week worth of randomly selected data and identify whether the generalized model can actually be applied to each individual. This test also shows how the model can correctly detect daily life events. In the case of undetectable life events, which are not defined in the FCA cross table, I classify them as "unknown" events and then check how many "unknown" events are detected. Table 6.2 shows the results of the personal applicability test. Table 6.3 will help to understand why some results in Table 6.2 are relatively low. Precision, recall, F-measure, and accuracy are calculated by following:

	Positive (Predicted)	**Negative (Predicted)**
Positive (Actual)	True Positive (TP)	False Negative (FN)
Negative (Actual)	False Positive (FP)	True Negative (TN)

Table 6.4 Confusion Matrix

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

$$F - Measure = \frac{2 \times Precision \times Recall}{Precision + Recall} \quad (3)$$

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \tag{4}$$

6.3 Evaluation and discussion

As shown in accuracy test, Table 6.1, Formal Concept Analysis demonstrates that it can substitute machine learning algorithms. The lattice-based model was built by manually scanning the training data set, and the relationships between collected attributes and detectable events were extracted based on knowledge about the events. No mathematical calculations were used to predict the events, but only a knowledge-based relationship table and concept lattice based on this were used. As a result, FCA, in common with the results of decision tree and random forest in Table 6.1, shows 100% accuracy under a given condition. It even has higher accuracy than linear classification and knn classification. These results demonstrate that FCA can correctly detect life events once it has detailed definitions of the relationships between events and their attributes.

Due to the proof of availability of FCA in Table 6.1, the applicability test with one-week worth of randomly selected data was now performed based on Formal Concept Analysis. To identify whether the generalized model can be applied to each individual, the performance obtained from the one-week worth of data of all three people is shown in Table 6.2. The extremely high performance, especially person 2 and 3 in Table 6.2, represents that either the system has good performance for all people or the daily data of them mainly has simple life patterns and most of their life events are one of the detectable events. For analysis of the real performance, I calculated the accuracy of each detected life events in Table 6.3. With regard to unknown event in this table, because the cross table only has “playing with a phone”, “sleeping at home”, “sleeping at other places”, “staying at home” and “studying”, and the rule-based filtering additionally detects “standing still”, “walking”, “running”, “bicycling”, “in vehicle” and “surfing the web”, other latent detectable events, such as “watching a movie”, “shopping” etc., which are not defined in the system, are regarded as “unknown” event. Table 6.3 shows that five detectable events occupy almost all the daily life events of the three people. Since FCA can detect life events pretty well if a structural similarity between defined events and incoming events is similar, these events can be detected with the highest prediction accuracy. Accuracy about “sleeping at home” and “studying” of person 1 and 3, however, show relatively low prediction accuracy, even if these are detectable events. These exceptional cases would happen due to some irregular life patterns. For example, if people fall asleep in daytime, wake up late, study at midnight, or study on weekend, FCA cannot detect these cases, which are beyond the cross table relationships and the person 1 and 3 often show these irregular life patterns. Therefore, except some irregular life patters, the generalized model can be applied to each individual and the model can correctly detect daily life events. Especially, this model shows the highest performance when the life patterns of people are simple and regular.

To deal with irregular cases and to avoid unexpected results, more detailed discretized ranges need to be set up. Current four kinds of time-band, which are dawn, morning, afternoon, and evening, show a limitation to detect “sleeping at home” when a person wakes up late. It needs to be more subdivided, such as two hours or one hour interval and check the life patterns more detail. The reverse geocoding information also needs to be supplemented. Sometimes the closest venue does not match to current location if a person is looking around a shopping mall or a great mass of buildings is nearby him. The location category also needs to be diversified for understanding more various life events, and more concrete attributes, which can be extracted from smartphone, need to be studied as well. Furthermore, a method how to harmonize Formal Concept Analysis with other possible algorithms to diversify life events detection and increase the detection accuracy will be discussed in future works.

CHAPTER 7. CONCLUSION AND FUTURE WORKS

Automatically understanding a situation between human activities and their surrounding contexts is a challenging problem. To extract a personalized life event in real-time from the uncertain semantics of heterogeneous raw data, a system must use an appropriate actuator, automatically collect influential life-logs, suitably process the logs, and combine the processed data into one targeted event. Therefore, to be a generic life event detection system, the system needs not only to cover all the conditions above, but also to be opened for all environments in the real world. However, most research on event detection has been confined a specific environment, such as Social Life Networking, video streams, or artificially organized sentient locations. To overcome the limitation, I have proposed a generic smartphone-specialized system, which can embrace all necessary features (i.e. ubiquity, open environment, practicality, real-time, and automated processes). Given the limitations of collecting life logs for a personalized statistical model, I applied concept-based data analysis to life event detection. As a means of evaluating whether a general life event model can be applied to each individual, and whether concept data analysis can be substituted for statistical data analyses in life event detection, I implemented the proposed system and tested it on three subjects over two months. The results revealed that using FCA, in conjunction with other tested machine learning algorithms, can be very effective, once cross table has detailed definitions of the relationships between life events and their attributes. Furthermore, the experiment showed that general life events can be applicable to all people if they have simple life patterns. However, the results also indicated that detection performance is dramatically reduced (i.e. has 20% lower accuracy) when life patterns are irregular cases with complex patterns. To handle the unexpected cases, 1) more subdivided discretization must be performed; 2) the accuracy of reverse geocoding needs to be increased; 3) finding more concrete attributes, which might diversify the number of detectable events, is necessary; and 4) combining other prediction algorithms, suitable devices (i.e. wearable sensors) and useful APIs (i.e. photo tagging) must be considered.

CHAPTER 8. REFERENCES

[1] Schilit, Bill N., and Marvin M. Theimer. "Disseminating active map information to mobile hosts." *Network, IEEE* 8.5 (1994): 22-32.

[2] Ryan, Nick S., Jason Pascoe, and David R. Morse. "Enhanced reality fieldwork: the context-aware archaeological assistant." *Computer applications in archaeology*. Tempus Reparatum, 1998.

[3] Abowd, Gregory D., et al. "Towards a better understanding of context and context-awareness." *Handheld and ubiquitous computing*. Springer Berlin Heidelberg, 1999.

[4] Hyungik Oh, Laleh Jaleh, and Ramesh Jain. "An intelligent notification system using context from real-time personal activity monitoring." *Multimedia and Expo (ICME), 2015 IEEE International Conference on*. IEEE, 2015.

[5] Gerald Friedland and Ramesh Jain, Multimedia Computing, Cambridge University Press, New York, NY, USA, 1 edition edition, July 2014.

[6] Laleh Jalali, Da Huo, Hyungik Oh, Mengfan Tang, Siripen Pongpaichet, Ramesh Jain. "Personicle: Personal Chronicle of Life Events." *Workshop on Personal Data Analytics in the Internet of Things*, VLDB 2014.

[7] Mittal, Sangeeta, Krishna Gopal, and S. L. Maskara. "A Versatile Lattice Based Model for Situation Recognition from Dynamic Ambient Sensors." *International Journal on Smart Sensing and Intelligent Systems* 6.1 (2013): 403-432.

[8] Mittal, Sangeeta, Alok Aggarwal, and S. L. Maskara. "Situation recognition in sensor based environments using concept lattices." *Proceedings of the CUBE International Information Technology Conference*. ACM, 2012.

[9] Seth, Parul. "Personicle: Contextual and Actionable Chronicle of a Person's Life," Master thesis, University of California, Irvine, 2014.

[10] Movable Type - Information Design & Management, [online], http://www.movable-type.co.uk/scripts/latlong.html (Accessed: 13 April 2015)

[11] Ganter, Bernhard, and Rudolf Wille. *Formal concept analysis: mathematical foundations*. Springer Science & Business Media, 2012.

[12] Yevtushenko Serhiy et al, "Concept Explorer. The User Guide", [online], http://www.comp.dit.ie/pbrowne/compfund2/UserGuide.pdf (Accessed: 20 April 2015)

[13] Lin, Jessica, et al. "A symbolic representation of time series, with implications for streaming algorithms." *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*. ACM, 2003.

[14] Marx, Morris L., and Richard J. Larsen. *Introduction to mathematical statistics and its applications*. Pearson/Prentice Hall, 2006.

[15] Katz S, Ford AB, Moskowitz RW, Jackson BA, Jaffe MW (1963) Studies of illness in the aged. the index of ADL: a standardized measure of biological and psychological function. J Am Med Assoc 185:914–919 2.

[16] Katz S (1983) Assessing self-maintenance: activities of daily living, mobility, and instrumental activities of daily living. J Am Geriatr Soc 31(12):712–726.

[17] Sandberg, S., et al. "Asthma exacerbations in children immediately following stressful life events: a Cox's hierarchical regression." *Thorax* 59.12 (2004): 1046-1051.

[18] McVeigh-Schultz, Joshua, et al. "Extending the lifelog to non-human subjects: ambient storytelling for human-object relationships." *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012.

[19] McVeigh-Schultz, Joshua, et al. "Vehicular lifelogging: new contexts and methodologies for human-car interaction." *CHI'12 Extended Abstracts on Human Factors in Computing Systems*. ACM, 2012.

[20] Mennicken, Sarah, et al. "Smart for Life: Designing Smart Home Technologies that Evolve with Users." *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*. ACM, 2015.

[21] Carpineto, Claudio, and Giovanni Romano. *Concept data analysis: Theory and applications*. John Wiley & Sons, 2004.

[22] Chen, Chien Chin, Meng Chang Chen, and Ming-Syan Chen. "An adaptive threshold framework for event detection using HMM-based life profiles." *ACM Transactions on Information Systems (TOIS)* 27.2 (2009): 9.

[23] Weng, Jianshu, and Bu-Sung Lee. "Event Detection in Twitter." *ICWSM* 11 (2011): 401-408.

[24] Xu, Changsheng, et al. "Live sports event detection based on broadcast video and web-casting text." *Proceedings of the 14th annual ACM international conference on Multimedia*. ACM, 2006.

[25] Ke, Yan, Rahul Sukthankar, and Martial Hebert. "Efficient visual event detection using volumetric features." *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 1. IEEE, 2005.

[26] Dougherty, James, Ron Kohavi, and Mehran Sahami. "Supervised and unsupervised discretization of continuous features." *Machine learning: proceedings of the twelfth international conference*. Vol. 12. 1995.