Scalable Daily Human Behavioral Pattern Mining from Multivariate Temporal Data

Reza Rawassizadeh, Elaheh Momeni*, Chelsea Dobbins*, Joobin Gharibshah, and Michael Pazzani

Abstract—This work introduces a set of scalable algorithms to identify patterns of human daily behaviors. These patterns are extracted from multivariate temporal data that have been collected from smartphones. We have exploited sensors that are available on these devices, and have identified frequent behavioral patterns with a temporal granularity, which has been inspired by the way individuals segment time into events. These patterns are helpful to both end-users and third parties who provide services based on this information. We have demonstrated our approach on two real-world datasets and showed that our pattern identification algorithms are scalable. This scalability makes analysis on resource constrained and small devices such as smartwatches feasible. Traditional data analysis systems are usually operated in a remote system outside the device. This is largely due to the lack of scalability originating from software and hardware restrictions of mobile/wearable devices. By analyzing the data on the device, the user has the control over the data, i.e. privacy, and the network costs will also be removed.

Index Terms—Frequent Pattern Mining, Temporal Granularity, Multivariate Temporal Data, Human-Centric Data

1 Introduction

The computing and networking capabilities of mobile and wearable devices, makes them appropriate tools for obtaining and collecting information about user activities' (mobile sensing). This has led to a significant expansion of opportunities to study human behavior ranging from public transport navigation [1] to well-being [2]. Moreover, the advent of mobile and wearable devices enables researchers to unobtrusively identify human behavior to an extent that was not previously possible. Nevertheless, there is still a lack of wide acceptance of mobile sensing applications in real-world settings [3].

There are different reasons for this mismatch between capability and acceptance. Firstly, the limitation of resources and a lack of accuracy in the collected contextual data, especially is a challenge with regard to the battery life [4]. Furthermore, the small size of sensors that are dealing with radio frequency, i.e. Bluetooth, WiFi and GPS, affects the quality of their data [5] (the smaller the device, the less accurate the data). For instance, Figure 1 visualizes two days of data from two users. As it can bee seen, the location data (●), WiFi data (▲) and other data objects are not available at all the time. The next reason is the proximity of the smartphone to the user, because these devices are not always carried by their owners [6]. However, smartwatches and wearables are body-mounted and thus the proximity problem is less challenging. Lastly, the operating system restrictions of mobile devices, which removes background services when the CPU is under a heavy load (in order to preserve the battery life). As a result, there is no ideal data collection approach that can sense and record individuals' information 24/7 with no data loss or uncertainty.

Existing works that support a mobile data mining [7], [8], [9], [10] have offered very promising results. However, these studies employ specific hardware, which is known for data quality among users [7], [8], or they analyze data offline outside the device [9], [10]. We believe there is lack of scalable data mining methods that can handle the uncertainty. In this work, we introduce scalable algorithms[1] that utilize a *variety of sensors* e.g. WiFi, location, etc. that are available on the device. By leveraging collected multivariate temporal data our algorithms can identify *frequent human behavioral patterns* (FBP) with a time estimation (*temporal granularity*), similar to the human perception of time. We have tested our algorithms, and their scalability, on two real-world datasets, and two small devices, i.e. a smartphone and smarwatch.

Identification of frequent patterns in human behavior has applications in several domains, which vary from recommendation systems to health care and transportation optimization. For instance, a health care application can monitor a user's physical activity routine. However, if there is a change in their routines, which is not recognized or

- *R. Rawassizadeh is with the Department Computer Science, Dartmouth College, 03755, NH, US*
  E-mail: rrawassizadeh@acm.org
- *C. Dobbins is Department of Computer Science in Liverpool John Moores University, Liverpool L33AF, UK*
  E-mail: c.m.dobbins@ljmu.ac.uk
- *E. Momeni is with Faculty of Computer Science in University of Vienna, Vienna 1090, Austria*
  E-mail: elaheh.momeni.roochi@univie.ac.at
- *J. Gharibshah and M. Pazzani are with the Department Computer Science, University of California Riverside, 92521, CA, US*
  E-mail: joobin.gharibshah@email.ucr.edu, michael.pazzani@ucr.edu

** These authors contributed equally.*

1. The UbiqLog dataset is available online at this link: https://goo.gl/rXxfnu. Due to licensing constraints we cannot make the Device Analyzer dataset available, but an access can be requested here: https://deviceanalyzer.cl.cam.ac.uk. If the readers are interested in the UbiqLog cleaning codes and visualization please contact authors.

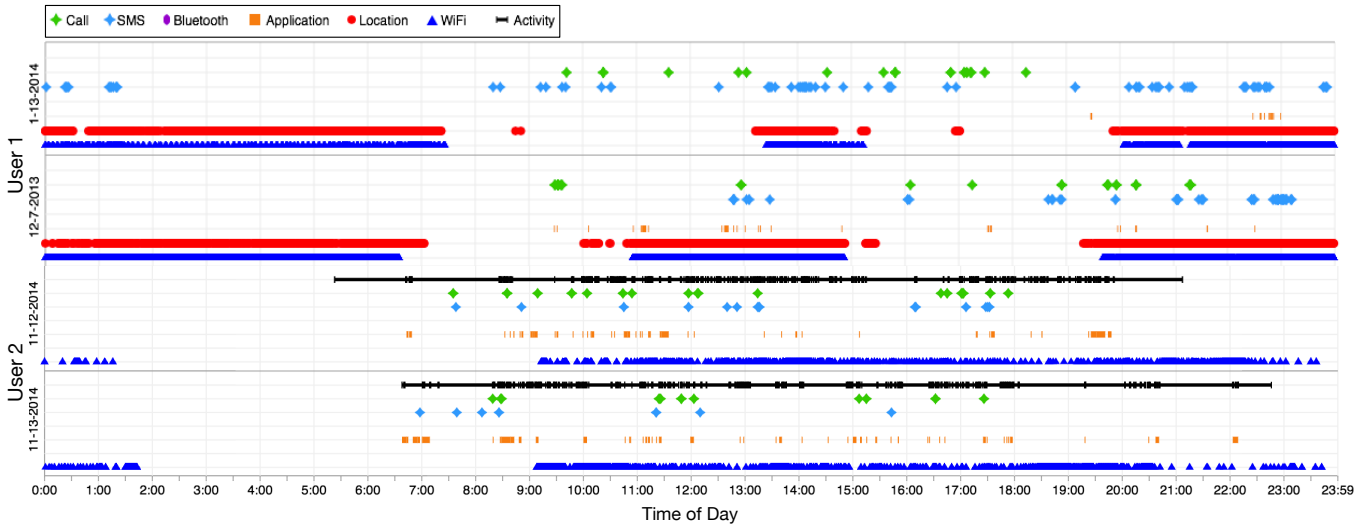

Fig. 1: Two days visualization for the data of two different users. The top two belong to User1 and the bottom ones belongs to User2. Location and WiFi logs from User1 between 7:00 to 8:00 and Activity logs from User 2 from 5:30 to 6:30 represent the small temporal differences in human behavior. (best viewed in color)

notified by the user (such as depression related behaviors), then the system can recognize this and notify care givers about the change. Another use-case can be transportation optimization. In order to arrive at the train station on time, a system can learn the routine commuter patterns of a user, and notify them about the appropriate time for leaving toward the station. On the other hand, the scalability (in terms of resource efficiency) enables on-device and online analysis, and therefore removes both the *network cost* and *privacy* risks of transferring personal data to the cloud.

The results of our algorithms are a set of identified FBPs, which is a combination of time stamped attribute/value (sensor/data) with a confidence level. For instance, {`confidence:60%; 15:00-16:00; call:#951603XXXX; sms:#951603XXXX`} is a user profile that includes one FBP. This example shows two repeated behaviors, which are (i) making or receiving a call and (ii) sending or receiving a text message to 951603XXXX. These two behaviors have been occurred 60% of the time, between 15:00-16:00 everyday.

The followings are characteristics and contributions of this research:

- Real-world Data: We have benefited from using two real-world datasets. One is a human-centric lifelogging dataset UbiqLog [11]. This dataset, in comparison with other mobile sensing datasets [7], [8], has been created using real-world settings. This is due to the variety of devices and the users' ability to turn on/off sensors. The second dataset that has been used, Device Analyzer [12], is hardware-centric. This is the largest real-world dataset, which has been created from Android phones. It includes timestamped hardware settings and operating system level changes of phones. Our focus is on human-centric behaviors. However, our algorithms can also be used to extract FBP in multivariate temporal data. Therefore, we have used the Device Analyzer dataset in our evaluations to demonstrate our algorithms versatility and in-dependency from the underlying data.
- Temporal Granularity: Unlike digital systems, human understanding of time is not precise. Our daily behaviors occur in time intervals. For instance, a person does not arrive at work every day at exactly the same time, or eat lunch at exactly the same time every day. A time interval always exists for routine behaviors, even if it is only a small break, e.g. five minutes. This is also true for precise time scheduled tasks, such as a meeting. Therefore, it is essential to have flexibility in temporal analysis. We have implemented this dynamic of human behavior by introducing a simple but novel human-centric *temporal granularity* method. Our algorithms use this temporal granularity instead of the original timestamp. Therefore, it should not be categorized as a time series approach.
- Scalability and Sensor Independence: A salient advantage of our approach is its scalability. It is lightweight and can be integrated into small devices with limited computing capabilities, e.g. wearables. Moreover, notwithstanding its temporal dependency, it does not consider the type of the underlying sensor data. We have converted heterogeneous sensor data into three tuples, which includes sensor name, data and discrete time. Such sensor independency makes the algorithm capable of running in settings that include temporal multivariate data, independent from any specific sensor. Furthermore, employing a combination of sensors rather than focusing on specific sensors, and using temporal granularity instead of exact timestamps, allows us to mitigate uncertainty by ignoring sensor data that is not available, and focusing instead on the available data.

Algorithm proposed in this paper has been implemented in the "insight for Wear"[2] an smartwatch app that is released into the market and benefit from predictive analytic. At the time of writing this paper, it is one of the top five lifelogging, quantified-self app in the Google Play market.

The remainder of this paper is organized as follows: First

2. https://play.google.com/store/apps/details?id=com.insight.insight

we start by formalizing the problem. Then, we describe datasets that have been utilized. Next, we describe the implementation of our algorithms; this is followed by the experimental evaluation. Afterward, we explain related work and conclude this paper.

2 Definitions & Problem Statement

We live in a spatio-temporal world and all of our behaviors occur in a specific location and time [13]. Therefore, to digitally quantify human behavior the target system should sense both time and location. Since location sensors, such as GPS, are not reliable (especially indoors) and it is not possible to collect this type of data at all time (24/7), we can only use time to link different information objects together. We define the problem as follows:

> ***Problem:*** Given timestamped activities of the user, assuming they are occurring in a routine, the goal is to efficiently create a *profile*, which summarizes *frequent behavioral patterns* of a user.

To be able to formulate the problem first we describe our definitions. Table 1 lists notations that we have used in this section.

Notation	Description
e	entity, is a tuple of $< A, D, T >$ that presents a fine-grained information unit
T	time interval of the entity based on temporal granularity, e.g. 16:00-15:00, 12:25-12:30
A	attribute name of an entity.
D	value of an entity. In this model per sensor only one data element will be used.
g	a set of similar entities (inside a window) that repeat in a consecutive days.
θ	minimum required number of entities between the same time intervals of two or more days.
$count(g)$	count the number of g appearances among all days (for each person).
λ	minimum number of repeats for a group to consider this group in the profile.
$profile$	a set of similar groups that have been repeated more than λ times.

TABLE 1: Notations and their descriptions.

Human behavior is composed of many daily activities that are distinctive and recurring. Here, these types of activities have been called "frequent behavioral patterns".

> ***Definition 1:*** Entity e, is assumed to be a fine-grained unit of human behavior and consists of a tuple of three $e =< A, D, T >$. Each entity contains a timestamp (time interval), T, attribute name, A, and attribute value (data) D.

For example, <*"activity","walking", 10:25-10:47*> is an entity and A is the "activity". The first task of quantifying a frequent behavior is to find entities that are occurring in the same time interval, in a series of consecutive days. Time intervals here refer to a normalized notion of the time, based on the temporal granularity. For example, the given time of 10:25-10:47 will be normalized to 10:00-11:00. In order to check if two time intervals of two (or more than two) days are similar. The number of equal entities in all time intervals, should be equal or greater than a threshold, which we call a **minimum entities threshold**, θ. In other words, θ is the minimum number of *similar entities* that should exists, in a *specific time interval* between two or more consecutive days. For example, assume θ has been set to two and we are comparing two days. In one day we may have <*activity, walking, 10:40-11:00*>, <*app,* ***skype****, 10:50-11:00*> then for the next day we have <*activity, walking, 10:40-11:00*>, <*app,* ***whatsapp****, 10:50-11:00*>. Since θ is set to two, at least two of these entities should be completely similar between 10:30-11:00. However, in the given example only one of them is similar, because there are different data, i.e. D (whatsapp & skype) for the "app" attribute A. Therefore, the 10:30-11:00 time interval, and its data, will not be counted as a frequent pattern between two days. We have introduced θ because some sensors, such as WiFi, have significantly more records than other sensors. Consequently, because of the similar WiFi records, there will be too many similar entities in each time interval, and not other sensors. Therefore, we define θ as a filter to force the similarity calculation to operate with better precision (more than one similar sensor). Here similarity calculation returns "true" for exact equality, otherwise "false" (not euclidean numerical similarity calculation).

> ***Definition 2:*** Group g, is a collection of similar entities, for a specific time interval, in a set of consecutive days. Therefore, $g = \{e_1, e_2, ..., e_k\}$, $e \in g$. In simple terms, if the number of entities in a specific time interval are greater or equal than θ, then they will be collected in a set and this set is being called $group$. T_c is a time interval that is constant among all entities of a group. In other words, groups are FBPs and the notation of a group is as follows: $g = \{\forall e : e_i(t) = T_c, \sum_{i=0}^{k}(e) \geq \theta\}$

k is the number of entities, which is always greater or equal to θ. $e_i(t)$ presents the time interval of the ith entity in the group. After the groups have been identified, the window moves to another set of days. To reduce the number of comparisons windows are disjointed and do not overlap.

We can simply compare entities together without creating groups. However, group based comparison avoids computational complexity. Comparing entities for all days together (without groups) creates a huge burden on performance $O(2^n)$, assuming n is the number of all days. To avoid this complexity we use the sliding window approach. The sliding window first compares m (window size) number of days together, as shown in Figure 3 a. Then it compares the windows' results together, i.e. m' (assuming there will be m' number of windows). This means the complexity is $O((m)^2 + m')$. m is the size of the window and it is significantly smaller than n, which is the number of all existing days. At the end, all of the results from each sliding window will be compared together to construct the profile that will be explained later. Moreover, the results that came from windows includes a fewer amount of entities than simply comparing all existing entities of between days. Therefore, the number of comparisons will be significantly reduced and the computational complexity will become near linear. We will demonstrate this impact in the evaluation section later (section 5.2).

The next step is to identify similar groups that have been repeated frequently among all days (compare results of windows together). Our initial experiments have resulted a

large number of groups that have been created by comparing between few number of all days. However, the lifetime of these groups are too short, and thus we can not literally call them a "frequent behavior". To remove these groups we have defined another threshold: **lifetime confidence threshold**, $\boldsymbol{\lambda}$. If the number of identified groups, among all days, is equal or greater than λ, then they will be considered as frequent patterns and will appear in the Profile. For instance, within six days worth of data, a windows size of two (two days will be compared together each time) has been used and λ of three. The result of each windows is as follows: *Window1:g1,g2, Window2:g2,g3,g4, Window3:g2,g3* . Since this example uses $\lambda = 3$, only $g2$ will be considered as a frequent behavior, and all other groups will be neglected. The profile is described as follows:

> ***Definition 3:*** Profile, is characterized by a set of repeated similar groups g which have been identified more than or equal λ times, i.e. $Profile = \{g_1, g_2, ..., g_k\}$. We can formalize profile as: $Profile = \left\{\bigcup_{i=0}^{k} g_i \ , \ if(count(g_i) \geq \lambda)\right\}$

In other words, Profile is a container of groups (FBPs) for a person or the union between k number of groups. If the count is greater or equal to λ, then these groups stay in the profile. The $count(g_i)$ function counts the occurrences of a group g_i, among other windows and stores this group in the profile.

This process results in a single (or multiple if we do the same for weekends or other specific days) profile for each user. Each group in the profile has a confidence in percentage, similar to the example that has been used in the introduction. The confidence presents the ratio of repeat for the target group during the course of analysis. Using the confidence enables the system to prioritize groups based on their repeat frequency.

3 Dataset

As previously noted, the development and testing of our work has benefited from access to two real-world datasets, UbiqLog [11] and Device Analyzer [12]. In contrast with previously considered smartphone datasets, i.e. Reality Mining [7] (uses Nokia N6600) and Nokia's Lausane data campaign [8] (uses Nokia N95), these datasets were collected in real-world settings, and have taken into account the device variety of users. Moreover, in contrast to previous datasets, they are both contemporary, and thus they consider the recent trend in the heavy usage of smartphones.

UbiqLog: The open source[3] UbiqLog [11] application, has been used to create the UbiqLog dataset. This application relies on participants' smartphones and has collected more than two months of lifelogging data from each participant. To preserve participants' privacy, UbiqLog is designed in such a way that participants can disable or enable sensors at any time. Participants have been asked to enable sensors that have been listed in Table 2. Activity data have been obtained from Google Play Services, Contact numbers in Call and SMS were stored with pseudonymization and SMS content was completely removed. The data collection process was performed in 2013 and 2014, on 35 participants. More description about the UbiqLog dataset and its participants' characteristics is provided in [14]. Table 2 shows a general overview about the data that has been collected from the participants. With the exception of WiFi and Bluetooth, which were sampled every six minutes, all other sensor data objects have been collected as they became available. Figure 1 presents a visualization of four days of data for two users, which we created to gain a high level view of the data. This figure represents the small temporal differences between user's activities, within two consecutive days and illustrates the need for temporal granularity.

Sensor Name	Num. of Instances
WiFi	8,750,111
Location	725,560
SMS	28,849
Call	99,022
Application Usage	45,803
Bluetooth Proximity	117,236
Activity State	15,641
All Data	9,782,222

TABLE 2: UbiqLog dataset records for each sensor. All records are semantically rich and are human readable records. Therefore, there is no raw sensor data, such as accelerometer data, in this dataset.

Sensor Name	Num. of Instances
WiFi	2,288,642
Application	98,392,622
Phone	15,719,384
SMS	104,643
Bluetooth	9,620
Analytics	2,910
Power	5,716,330
System	1,051,175
Audio	4,839,668
CPU	1,143,736
Image	2,281,293
Video	152,397
Memorycard	83,572
Net	232,954
HF	16,687
All Data	132,035,633

TABLE 3: 35 random users' records from Device Analyzer dataset.

Device Analyzer: Device Analyzer [12] is the largest dataset available that contains hardware statuses and device configurations of Android smartphones. It collects data for about 23,000 users and it includes more than 10 billion records of "raw" sensor data. This is a promising real-world dataset. However, unlike the UbiqLog dataset, Device Analyzer's focus is on hardware-specific information collection and not user-centric data. Therefore, we cannot perform user-centric analysis, using this dataset. Nevertheless, since it includes multi-variate timestamped data, we can use this dataset to demonstrate the scalability of our approach on other multivariate timestamped data. 35 random users (equal to the number of UbiqLog participants) have been chosen for our experiments. In total 132 million raw data records have been processed. These data objects include timestamped information about hardware-related data, such as network usage, WiFi connections, system processes, high frequency background services (HF), etc. Table 3 shows the number of records that we have used in our analysis from 35 users of the Device Analyzer dataset.

4 Frequent Behavioral Pattern Identification Algorithms

In order to implement our algorithms for the problem described above, firstly the data format should be converted from heterogeneous data to machine-processable data, i.e.

3. https://github.com/rezar/ubiqlog

the raw data needs to be converted to the previously described entity format. As previously stated, the data has been collected from heterogeneous sources. Some sensors have multiple values, for instance WiFi has BSSID, SSID and Capalities (WPA, PSK, etc.). Nevertheless, for each sensor our model chooses only one value. In particular, each sensor (attribute) A, requires a single data point (value) D. Therefore, "BSSID" has been chosen for WiFi and Bluetooth, the pseudonymized phone number for SMS and Calls, "process name" for Application and tilting, in-vehicle, on-bicycle, walking, still, and unknown for the activity sensors (UbiqLog uses Google play services for activity recognition and therefore there is no raw accelerometer data inside the dataset). A similar approach has also been used for the Device Analyzer dataset, which we do not report it here to preserve space.

During the second step, we propose an algorithm that identifies the movement (based on location changes) state, which will be used to enrich the semantics of the data within the notion of the location. In third step, we need to convert the timestamp to a time similar to the human perception of time. Afterward, in the fourth step we describe the behavior similarity and FBP detection algorithms. Figure 2 presents the flow of FBP detection from raw, heterogeneous sensor data.

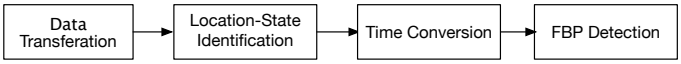

Fig. 2: FBP extraction flow from raw hetregenous sensor data.

4.1 Location State Estimation

Red dots (•) in Figure 1 are not just GPS data. They could also be a combination of Cell-ID, GPS, and any third party service that provides geographical coordinates. Lamentably, in a real-world setting, 24/7 geographical coordinates identification is not possible, especially in indoor environments and due to battery limitations, GPS is usually turned off. Cell-ID does provide geographical coordinates and it is more frequently available, but it is too imprecise for location recognition [15]. Figure 1 shows the uncertainty that has been existed among all sensor data. On the other hand, location state such as being in home, at work, etc. is widely used for human behavior detection [7], [16], [17], [18].

Previously, there have been a few works [15], [19] that focus on extracting location from a *combination* of different data sources. In contrast, there are several other works that focus on extracting location from a *single* source of information [20], [21], [22], [23] and provide promising results. Nevertheless, as it has been show in Figure 1 in a real-world setting we have *sparse* set of geographical coordinates but we can benefit from *combination of sensors* (WiFi, GPS, CellID). Therefore, we need a novel algorithm to transform these data to movement state, which could be *moving*, *stationary* or *unknown*. Our notion of location (movement state) is more limited than other spatial based research efforts, which consider the geographical locations or trajectories of users. However, our definition has the two advantages of simplicity and greater availability. The contribution of location annotation in this work is to be as used as a supplementary element to improve the probability of the FBP detection.

Occasionally, geographical coordinates is completely unavailable, as is the case with the Device Analyzer dataset. In these cases, the WiFi data has been used for estimating the location. Based on this description, the location estimation algorithm must be able to identify location changes from a combination of sensors (sensor fusion). Furthermore, it is important to note that our focus is on the data that is being collected from the users' device (user-centric) and not a third party service, such as a call detail record (CDR) [24], [25]. Algorithm 1 presents our location state estimate. The

Algorithm 1: Location state (based on movement) estimation from different signals.

```
   Data: entities,signalType
   Result: results
1  if (isWiFi = signalType) then
2      forall the (locations in entities) do
3          moving ← contDiff(locations);
4          if (moving != ∅) then
5              results.add(moving) ;
6          else if ( moving = ∅ &
7          contSim(locations) != ∅ ) then
8              results.add(stationary);
9          else
10             results.add(unknown);
11 else
12     forall the (locations in entities) do
13         locstate ← parseGPS(locations);
14         if (locstate = ∅) then
15             locstate ← parseOtherSignals(locations);
16         if (locstate = moving) then
17             results.add(moving) ;
18         else if ( locstate = stationary & contSim(locations) != ∅
           ) then
19             results.add(stationary);
20         else
21             results.add(unknown);
22 return results;
```

algorithm receives a set of *entities* and a *signal type* as inputs, and it returns a list of entities with a location state in the *results*. Entities with location states include additional data, which is the location state. Therefore we have a four tuple entity.

As the first step, the algorithm checks $signalType$, line 1. If the signal type is only WiFi, then it returns true; otherwise, there is a combination of location signals, and the algorithm continues from line 11. The $contDiff$ method, at line 3, searches for a sequence of continuous WiFi BSSIDs, which have different names. If a sequence exists, and if no WiFi BSSID has been repeated in the sequence, this is a sign of a moving event. Therefore, a *moving* event is created and appended to the $results$ list (line 5). Otherwise, if there is a sequence of WiFi BSSIDs, but at least one of them is repeated (they are not unique), the $contSim$ method returns them, and a *stationary* event will be created and appended to the event list ($results$) at line 8. For instance, in the sequence $w_1, w_2, w_3, w_4, w_1, w_3$ includes two repeated elements, w_1 and w_3. In this case, the algorithm will create a *stationary* event from this sequence. In line 10, if there is no WiFi signal at all, then an *unknown* event will be created and appended to the $results$ list. In summary, the algorithm checks WiFi

BSSID data objects that have appeared sequentially and if they are not unique, then it creates a *stationary* event. Otherwise, if there are unique elements, the algorithm creates a *moving* event, and if none of these cases exists, then it creates an *unknown* event.

If geographical coordinates exist, then the status of the location is easily recognized. To calculate location state from geographical coordinates the algorithm checks the differences between two consecutive points and calculates the state (if it is moving or stationary). Method $parseGPS$, line 13, implements this scenario.

Nevertheless, occasionally the GPS might not be turned on and so there will be very few GPS logs (mostly when users are navigating). In the UbiqLog dataset, most location logs will be from Cell-ID. As it has been described, it is not possible to precisely identify if the user's location has changed or if the cell tower has been switched. Therefore, the calculation should be flexible with 800 to 1000 meter accuracy [20]. To cover this precision problem, instead of calculating the distance between two consecutive points (geographical coordinates), we calculate the distance between three consecutive points. If the distance between the first and third point is more than 800 meters, then we can conclude the user is *moving*. Method $parseOtherSignals$, line 15, implements the location (movement state) calculation from Cell-ID data objects.

The complexity of algorithm 1 is linear, $O(n)$, because even if we assume all coordinates are Cell-IDs there is no need for a comparison between each element and its two previous ones. Therefore, in a worst case scenario, we will have a $3n$ comparison, which is still a linear complexity.

Afterward, a file is created for each user, which includes their data. Each of these records contains four elements: attribute name, timestamp and attribute data, which is a presentation of a three-tuple entity and location.

4.2 Temporal Granularity

According to Poidevin [26], we do not perceive time in and of itself, but rather, we perceive changes or events in time. To be able to model human behavior, a precise machine timestamp should be transferred to a format similar to the way humans perceive time. In a more technical sense, humans perceive events in relation to both location and time [13]. In contrast to the location, all existing mobile and wearable devices can record information with a timestamp. In order to simulate the human perception of time temporal granularity [27] (TG) will be used. Setting the TG is also depends on the target application. Here we attempt to make a TG for the daily behaviors. For instance, every evening a user may make a phone call. However, it is unlikely that s/he will call every day exactly at 5:00pm; s/he could call one day at 5:21pm and another day at 4:53pm. As a result, we define TGs based on *common daily time scheduling*, and a use rounding algorithm to convert times based on the given precision.

Algorithm 2 is the algorithm for calculating TG. It is simple and deals only with a timestamp conversion, based on given precision and predefined rules, thus its computational complexity is also $O(n)$. Since it could be read easily we save space and do not describe it in detail. In short, it receive a day, D_{in}, with all entities inside that day, and the $precision$ for TG such as one hour, half an hour, etc. It then iterates through time element entities for the given day, and creates two normalized time ($TmpCeil$ and $TmpFloor$) based on ceiling and flooring the given time. Next, it calculates the distance between the original time, i.e. $D_i(T)_H$, and both ceil and floor. Afterward, it returns the shortest one that is either ceil or floor. The returned time objects now substitutes the original time of the entity.

The TG creation algorithm, from the timestamp, can work

Algorithm 2: Temporal granularity calculation.

Data: D_{in}, $precision$
Result: D_{out}
1 //iterate through entities of a date **for** $(i{=}0\,;\,(i < D_{in}.e(length)))$ **do**
2 // read hour and minutes of current entity
3 $TmpCeil \leftarrow ceil(D_i(T)_H, precision)$;
4 $TmpFloor \leftarrow floor(D_i(T)_H, precision)$;
5 $T_{abs} \leftarrow distance(D_i(T)_H, TmpCeil, TmpFloor)$;
6 $D_i(T) \leftarrow T_{abs}$;
7 $D_{out}.add(D_i(T))$
8 return D_{out};

with different timeframes. However, in our experimental evaluation, we define six time frames, which have been used in daily communication: Five minutes (for time-sensitive tasks such as attending a meeting), a quarter of an hour, half an hour, an hour, one and half hours and two hours.

This temporal similarity transformation can handle uncertainty by focusing on similar data in a perceptible time interval (i.e. a quarter of an hour, half an hour, etc.). Therefore, using TG reduces uncertainty originating from different times of routine behaviors.

4.3 Frequent Behavioral Pattern Detection

After the data has been transformed and its timestamp has been converted, then the similarity detection algorithm starts to build groups of similar entities. First, we introduce the group creation algorithms from similar entities, then we describe the method that builds users' profiles by filtering groups. Figure 3 visualizes algorithm 3 that we have

Algorithm 3: Group creation from similar entities.

Data: D_{ins}, ws,θ
Result: All Detected Groups in a Window
1 $grpAll, grpPrev \leftarrow \varnothing$;
2 $entArr, entArrNext \leftarrow \varnothing$;
3 **while** $((D_{ins}.hasNext) < ws)$ **do**
4 //reading entities of current day
5 $entArr \leftarrow D_{ins}.current.e$;
6 //reading entities of next day ;
7 $entArrNext \leftarrow D_{ins}.next.e$;
8 //compare and collect similar entities ;
9 $entSimilar \leftarrow compare(entArr, entArrNext, \theta)$;
10 // add similar entities into a group;
11 $grpTmp.add(entSimilar)$;
12 **if** $(grpPrevious.containsData())$ **then**
13 $grpPrevious \leftarrow getSimilar(grpTmp, grpPrev)$;
14 $grpAll.add(grpPrev)$;
15 **else**
16 $grpAll.add(grpPrev)$;
17 return $groupAll$;

proposed for group creation. The window size is set to be three (Figure 3 a), one day as a weekend (green boxes) will be neglected[4], and θ is equal to two. Figure 3 b. shows that

4. The target city of the experiment only has a one-day weekends.

two entities, in each time frame will be compared between two consecutive days. By comparing two days, D1,W1, with D2,W2, two groups, G1 and G2, have been extracted. For the sake of brevity, we have not visualized a comparison between more than two days. Algorithm 3 receives the input days, D_{ins}, window size ws, and minimum threshold θ. In line 3, it iterates through the days, D_{ins}, and reads entities for each day. It then compares the current entities to the entities of the next day, using the $compare$ method and keeps the similar ones in a temporary group $grpTmp$ (lines 9 to 11). If a previous similarity group exists, $grpPrev$, then it updates that group via the $getSimilar$ method, in line 13. This process is repeated for the given learning days and all similar groups in the given window size. Results will be collected and returned in an array called $grpAll$. In summary, each window returns a set of groups. Since

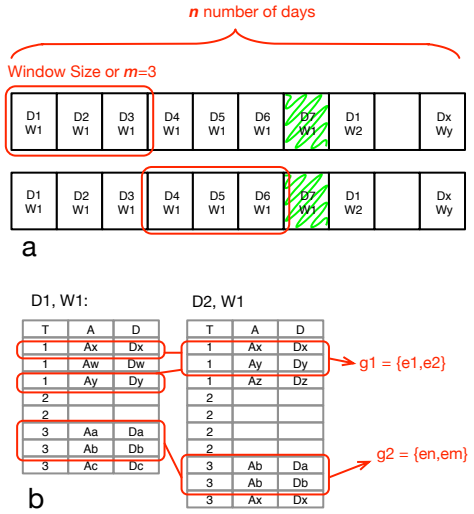

Fig. 3: Group creation based on similarities between entities. 'D' presents day and 'W' presents week. Figure a. presents a sliding window with a size of three. Figure b. presents similar entities that have been detected between two days; window size and θ both are equal to two.

behaviors are just combinations of groups, we can add them all together to have one set that includes group objects. This set will be called $Profile$. Algorithm 4 summarizes the collected groups and returns the profile object. In line 3, it iterates through the objects of a given group array. It increases the confidence of repeated group objects and removes them from the array in line 5 to 8. Next, it calculates an intersection between groups, and if the appearance of a group is more than the λ threshold (line 10), then this group will be added to the user's profile. At the end, it returns the $Profile$ object. Both algorithms 3 and 4 are linear (as it has been described in section 2).

Existing works [9], [28] provide association rule mining to identify correlation between contextual data. However, our work aims to identify human behavior, instead of the unique contextual data approach.

4.4 Algorithms Limitations

It is important to note that we still cannot map these information objects onto all existing real-life events, such as nested events, e.g. being at work includes drinking coffee, using a printer, etc. Nevertheless, our work offers

Algorithm 4: Creating profile from groups.

```
   Data: Groups, λ
   Result: Profile
1  Profile ← ∅;
2  // finding similar groups ;
3  while (Groups.hasNext) do
4      // two groups are equal ;
5      if (Groups.next = Groups.current) then
6          // increase the confidence of the current group
           Groups.current.confidence + 1 ;
7          // remove the repeated group ;
8          Groups ← remove(Groups.next) ;
9  // prune groups confidence based on λ ;
10 while (Groups.hasNext) do
11     if (Groups.current.confidence ≥ λ) then
12         Profile.add(Groups.current) ;
13 return Profile;
```

a significant step toward more intuitive understanding of human behavior (especially with the TG we are using). Moreover, our approach does not rely on a unique source or sensor; therefore, data is extracted from multiple sources. Therefore, if a single sensor fails, its impact is insignificant. This helps mitigate the problem of uncertainty originates from the nature of contextual data.

Another important issue with our approach is its behavioral scope limitation. The model we have proposed here is *time dependent*. This approach can identify daily consecutive behaviors but not all routine behaviors.For instance, going to the cinema every two or three months or going to a campaign once a year, is not going to be identified by our approach. These are routine behaviors, but our approach can not identify them. A solution to that problem is to collect all anomalous behaviors and apply the algorithm on the collection of those behaviors. This solution has not been explored in this work, because we focus on daily routine behaviors.

Furthermore, if a behavior occurs in a sparse temporal settings, such as calling a person every day, at different times of the day, then it is not considered as a routine behavior. The current version of our algorithm limits the behavior comparison in a scope of the sliding window and temporal granularity. In other word, the routineness of a behavior should be within the scope of a temporal granularity. Nevertheless, based on the experiment described in [14] there are very few daily routine behaviors that have sparse temporal settings.

5 Experimental Evaluation

The first step in our experimental evaluation is the creation of a ground truth dataset that can help us in estimating the *accuracy* of algorithms. In particular, we evaluate the accuracy of the FBP identification algorithms based on (i) different segments of the day, (ii) different TGs, and (iii) we report about the accuracy of our approach in comparison to other algorithms. We use Apriori [29], FP-Growth [30] as a baseline, and MTK [31] and estDec+ [32] as state-of-the-art algorithms. Apriori is a baseline algortihm for frequent itemset mining; FP-Growth is a well-known baseline for fast itemset mining. MTK (Memory-constraint Top-K frequent-pattern mining) is scalable and can operate in limited memory environments. estDec+ is a new fast and memory

efficient algorithm that uses a weighted-based approach for item set mining, which is similar, but significantly more advanced, than the weighted algorithm has been used by MobileMiner.

The following experiments describe the next phase of our evaluation and demonstrate the utility and efficiency of our algorithms: Firstly, we demonstrate the scalability of the FBP detection algorithm by analyzing the impact of window size (WS) and grouping on the execution time. Our FBP approach is lightweight enough to be used in wearable and mobile devices. These devices have limited resources [5], thus investigating the execution time is very crucial in demonstrating if our approach is scalable on these devices. Next, we report about the execution time of our approach and compare it with other algorithms on both mobile phones and smartwatches. Moreover, we compare FBP battery and memory utilization with these algorithms.

We then report about the sensor impact on FBP identification. This could assist us in identifying the most influential device sensors that can be used to identify FBPs and remove unnecessary sensor data collection to preserve disk space and battery. Finally, we present a statistical overview of the impact of changing the thresholds (i.e. λ and θ), TGs and sensors on the FBP identification. This can help developers to identify boundaries to configure these variables so that they are effective for the sensors.

It is important to note, the focus of this research is on end-users' behavior and thus the device setting data are not necessarily representing the users' behavior. Nevertheless, we have used the Device Analyzer dataset to demonstrate the scalability of our algorithm on multivariate temporal data.

5.1 Accuracy Analysis

5.1.1 Ground Truth Dataset

In order to evaluate the *accuracy* and *quality* of the identified FBPs, we have created a ground truth dataset, which is composed of more than 5,000 identified entities (that participate in FBPs), from five users. It contains randomly identified FBP data that has been labeled by the users as either true or false. The number of identified entities in the profile objects are different among users, but each user has labeled about 1,000 entities that belong to him/her self. Users were asked to label if they agree (true) with each of their identified entities in their profile or if they do not agree (false). Each user only labels his or her entities and not other users' behaviors (profile, groups, entities).

5.1.2 Accuracy of Identified FBPs

After collecting the labels, we carefully examined the accuracy of our algorithms using three temporal segments of the day: 0:00-07:59 (0-8), 08:00-15:59 (8-16) and 16:00-23:59 (16-24) and different TGs. This time based segmentation has been inspired by similar work in mobile data mining [10], but it is more accurate than the two divisions proposed by Ma et al. [10].

The result of labeling shows that FBPs with more than 20% confidence were mostly labeled as positive results, and lower than 20% confidence were labeled as negative results. 20% seems to be a low confidence level and we believe this is because of the short lifetime of FBPs that have originated by the sparsity of the data.

Based on one hour TGs, Figure 4 shows two segments of the day 0-8 and 8-16 and contains more accurately identified FBPs than from the 16-24 segment. This could be attributed to the fact that between 0-8 is usually a time when a user is asleep (a very routine behavior) and 8-16 is usually the time when a user is at work/school (also a routine behavior). Surprisingly, 16-24 (leisure time), has a low likelihood of having routine behaviors and its accuracy is lower than the other segments. This is in contrast with our initial hypothesis that 8-16 might have the lowest number of routine behaviors than the other two segments, because participants in the UbiqLog dataset were students, and should not have a very different behaviors from 16-24.

Table 4 reports about the accuracy of the identified FBPs, based on labels, with different TGs and not using a TG at all (baseline). The results of this analysis shows that FBP identification accuracy is influenced by different values of TG and the segmentation of the day. Table 4 shows that identifying FBPs using "an hour" as the TG improves the accuracy of the FBP identification, compared to other TGs. In other words, one hour TG has the highest accuracy among other TGs. Nevertheless, choosing 15' ,30' ,90' and 120' as TG performs almost the same or slightly lower than 60' but better than 5'. The inaccuracy of five minutes is due to the fact that this TG is too precise for an application to model human behavior. We can conclude that most routine human behaviors that can be identified from a smartphone have a one hour approximation. Nevertheless, we should consider that the sensitivity of TG is application specific. For instance, an application can identify FBPs by using smartwatch heart-rate and physical activity sensors. In this instance, it should have a short TG so that it can recognize anomalous hear-rate activities during a routine exercise.

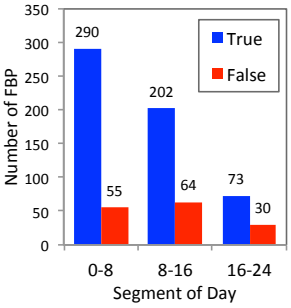

Fig. 4: Correct and incorrectly labeled FBPs based on time segment for TG=60'.

TG	Temporal Segment			All
	0-8	8-16	16-24	
0'	0.52	0.56	0.46	0.48
5'	0.66	0.62	0.64	0.62
15'	0.79	0.71	0.72	0.65
30'	0.84	0.74	0.70	0.69
60'	**0.84**	**0.76**	**0.71**	**0.77**
90'	0.80	0.75	0.71	0.75
120'	0.81	0.73	0.70	0.75

TABLE 4: FBP identification accuracy with different temporal segments of a day and different TGs. 0' is the baseline.

Our other evaluation uses the same users to annotate the results delivered by similar algorithms, and compares the accuracy of our FBP algorithm with Apriori, FP-Growth, MTK and estDec+. Table 5 shows the labeling results of users. With no TG (baseline) or low (5') and high (120') TG, other methods perform better than FBP. For the rest of the TGs, FBP outperforms other methods. This accuracy originates in the use of λ and θ. Other algorithms are very useful for the general problem domain. Nevertheless, for multivariate temporal data, defining a minimum required number of entities (θ) and filtering noise based on the minimum number of repeated groups (λ), results in improved

accuracy. Unlike Apriori, both MTK and estDec+ apply another level of filtering such as considering top frequent itemsets by MTK or recent ones by estDec+. Therefore, baseline algorithms that do not filter have superior accuracy.

TG	Apri	FP-Growth	MTK	estDec+	FBP
0′	**0.55**	0.54	**0.55**	0.51	0.48
5′	0.66	0.62	0.62	**0.67**	0.63
15′	0.58	0.60	0.63	0.62	**0.65**
30′	0.65	0.63	0.66	0.67	**0.69**
60′	0.69	0.71	0.68	0.73	**0.77**
90′	**0.75**	0.71	0.70	0.71	**0.75**
120′	**0.78**	**0.78**	0.77	**0.78**	0.75

TABLE 5: Accuracy of our FBP algorithm in comparision to Apriori, FP-Growth, MTK and estDec+ algorithms, with different TGs.

5.2 Scalability

Our algorithms must be capable of being integrated into small devices, which have restricted computational resources compared to desktop computers. In order to demonstrate that our algorithms are lightweight, we have measured the execution time, which is the representation of *scalabilty*. Moreover, we have chosen to evaluate the execution time of our algorithm among Apriori and FP-Growth as baseline algorithms and MTK and estDec+ as state-of-the-art algorithms. Apriori is the well-known algorithm for frequent itemset mining. It is not the fastest or most resource efficient but we have chosen it as a baseline algorithm and it has been used in other mobile data analysis works [9], [28]. In contrast, FP-Growth is scalable and is known as a baseline of the fastest frequent itemset mining algorithms. Furthermore, MTK and estDec+ both also have been used as state-of-the-art algorithms that are scalable and fast.

5.2.1 Sliding Window Impact on the Execution Time

Execution time is directly correlated to scalability and scalability is a major contribution of this work. It has been achieved through (i) the adoption of a sliding window and (ii) the reduction of the number of comparisons via utilizing a group based comparison. To demonstrate scalability, firstly

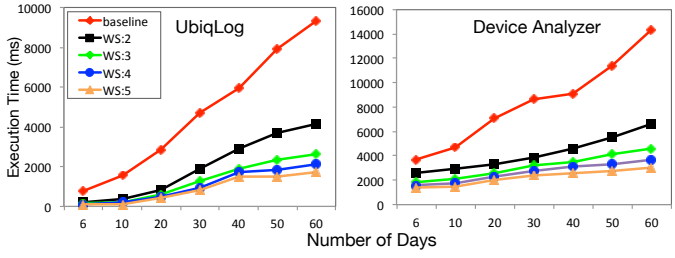

Fig. 5: The effect of window size (WS) on the execution time performance.

we have analyzed the execution time performance of the FBP algorithm with different window sizes for 60 days. This time frame has been utilized as it covers a significant period of time so that the capability of the FBP algorithm can be fully tested. Dealing with a large number of days is an important requirement in Lifelogging systems, which use small devices [33]. The baseline here is not using the window, and it compares each day with all the other days. Figure 5 summarizes these performance changes for both the UbiqLog and Device Analyzer datasets. The legend on the top-left side in Figure 5 shows the window size (WS). Since weekend behaviors are different than weekday behaviors, we recommend to compare them separately from the weekday data. In particular, we recommend not using a window size larger than five or six days. However, this depends on the weekend duration, i.e. if weekends are two days or one day. Therefore, the upper bound could be the number of the weekdays.

The results illustrate that increasing the window size significantly improves the execution time performance. In other words, a smaller slope means better performance, and increasing the window size decreases the slope significantly in both datasets. Even increasing the number of days, does not affect the performance of the FBP algorithm. The results depicted in Figure 5 belong to one user, for 60 days, and have been measured on a MacBook with Intel Core 2 Duo 2.4 GHz CPU and 8 GB RAM. Another factor that affects the scalability is the use of grouping instead of simple comparisons. FBPs are designed for multivariate temporal data. Most of the similar algorithms to FBP are frequent itemset mining algorithms.

Num Days	UbiqLog					Device Analyzer				
	Ap.	FG	MTK	eD+	FBP	Ap.	FG.	MTK	eD+	FBP
2	0.36	**0.25**	0.34	0.45	0.47	**0.33**	0.56	0.46	0.51	0.60
4	0.57	**0.29**	0.35	0.67	0.53	**0.44**	0.81	0.76	0.57	0.79
6	1.05	0.64	**0.40**	0.83	0.98	1.19	**0.98**	0.88	0.99	1.21
8	1.81	1.09	**0.57**	0.93	1.25	1.93	1.47	1.56	1.26	**1.01**
10	2.32	1.40	**0.68**	1.02	1.81	2.17	1.98	1.69	1.34	**1.32**

TABLE 6: Execution time (in seconds) of the five algorithms, while "not" using the sliding window, on the Macbook.

Table 6 reports on the execution time differences (in seconds) between the algorithms from 2 to 10 days on the same MacBook machine, *without* using the sliding window. Our algorithm outperforms Apriori (Ap.) but it is quite similar to FP-Growth (FG). Nevertheless, MTK and estDec+ (eD+) both outperform FBP. Note that the sliding window has been disabled in this experiment. The next experiments report with the sliding window enabled.

5.2.2 Comparison with Other Algorithms

We have compared the FBP *execution time*, *memory* and *battery utilization* of a sample of user data (one user from each dataset) on both smartphones and smartwatches. Based on default SPMF library settings [34] minimum support for all algorithms have been set to two. λ, θ have been set to two and a window size of three for FBP has been used. To port these algorithms we have used the implementation from SPMF [34], and we have adapted them for Android devices. For the smartphone test we have used a Nexus 5, with 2.26 GHz quad-core Krait 400 CPU with 2GB Ram. For the smartwatch test, we have used a Sony S3, with four core ARM Cortex-A7 1.2GHz CPU and 512MB Ram. However, due to small RAM and CPU of smartwatches, a maximum of 10 of days worth of data has been considered for the analysis. Otherwise, similar to other resource intensive operating systems, the Android Wear timeouts the long running process based on its resource preservation policy.

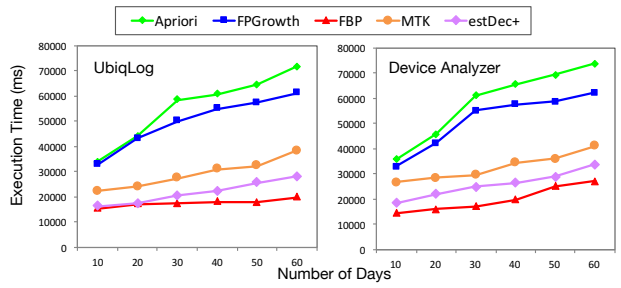

Fig. 6: Execution time comparision between FBP and other algorithms on the smartphone.

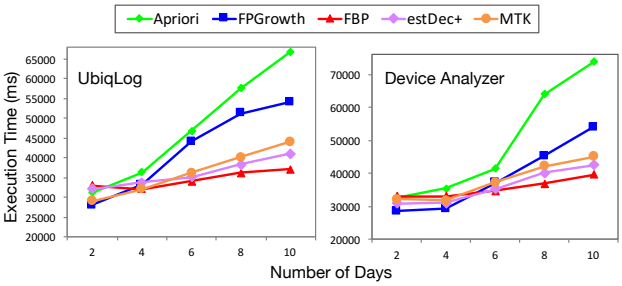

Fig. 7: Execution time comparision between FBP and other algorithms on the smartwatch.

Figure 6 shows the execution time of running FBP in comparison to other algorithms on the smartphone, within the described settings. Figure 7 shows the execution time of FBP in comparison to other algorithms on the smartwatch. For more than six days of analysis, FBP execution time performance is faster than other algorithms. This has been highlighted especially as the number of days increases, the FBP execution time does not change significantly and stays at a near constant value. It has even outperformed state-of-the-art algorithms, which are known to be fast and scalable algorithms that operates on limited memory. However, for a smaller amount of data, FBP does not perform better than MTK or estDec+.

Table 7 shows the comparison between FBP and other algorithms for battery utilization on the smartphone. Respectively Table 8 shows the similar data on the smartwatch. Table 7 and Table 8 show FBP battery utilization is lower than all other algorithms, but only with larger than 10 days of data. In particular, for a small number of days (less than 10 days), there is no significant battery utilization differences between algorithms, and FBP does not perform as efficient as others.

It could be argued that a fast execution time can be easily achieved through increasing the memory usage, which is an important resource. To demonstrate the scalability based on memory use, Figure 8 shows the maximum allocated heap used on the smartphone for both datasets, and it compares FBP memory use with other algorithms. Respectively, Figure 9 shows FBP the maximum allocated heap memory on the smartwatch. Figure 8 and Figure 9, show that when the number of days increases FBP is more memory efficient than other algorithms. When there are a small number of days all other algorithms outperforms the FBP. Therefore, from a memory usage perspective, when there are more than

Num Days	UbiqLog					Device Analyzer				
	Ap.	FG	MTK	eD+	FBP	Ap.	FG	MTK	eD+	FBP
10	245	**208**	220	212	212	360	149	**113**	128	115
20	255	218	225	218	**215**	625	262	**231**	234	239
30	247	234	232	218	**220**	861	347	352	368	**303**
40	285	228	239	221	**221**	911	389	399	413	**329**
50	318	268	251	**226**	227	1032	428	417	489	**337**
60	346	271	263	240	**236**	1428	445	470	519	**349**

TABLE 7: A comparison between battery utilization in micro-Amper-hour (mAh) of the five algorithms on the smartphone, for both datasets. Ap. stays for Apriori, FG for FP-Growth and eD+ for estDec+.

Num Days	UbiqLog					Device Analyzer				
	Ap.	FG	MTK	eD+	FBP	Ap.	FG	MTK	eD+	FBP
2	0.03	**0.01**	0.02	0.10	0.018	0.34	**0.02**	**0.02**	0.04	0.05
4	**0.03**	**0.03**	0.05	0.12	0.31	0.72	**0.06**	0.08	0.11	0.07
6	**0.04**	0.09	0.09	0.13	0.73	0.81	**0.09**	0.11	0.12	0.19
8	**0.09**	0.18	0.14	0.16	0.76	0.89	0.14	**0.11**	0.16	0.21
10	1.27	**0.23**	**0.23**	0.25	0.79	0.94	0.19	**0.17**	0.21	0.25

TABLE 8: A comparison between battery utilization in micro-Amper-hour (mAh) of five algorithms on the smartwatch, for both datasets. Ap. stays for Apriori, FG for FP-Growth and eD+ for estDec+.

10 days of data available, FBP is more efficient than other algorithms. Nevertheless, similar to the battery utilization, calculating for small number of days, shows that the memory usage for all algorithms is very insignificant and not worth for further investigation.

These evaluations demonstrate the scalability of our algorithms, while *preserving accuracy*. From a technical perspective, this superiority is because of: (i) using a sliding window that filters most of the irrelevant entities, and thus reduces comparisons significantly. (ii) theta and lambda that propose two layers of hierarchical filtering and result in both improving accuracy and reduction in the search space, which overcomes the state-of-the-art methods [31], [32].

5.3 Users Characteristics

As it has previously been stated, resource utilization is a challenge on small devices. To mitigate this issue, the system should be prevented from continuously running our algorithms. Instead, it is more important to know *when* the most appropriate time to run the algorithms are. In particular, we should know the *frequency* that these algorithms should be run. For instance, a group of users could have routine behaviors during the evening and not many routine behaviors during the day; if a system learns this, then it will execute the algorithms only in the evening.

To achieve this goal, we have analyzed the temporal differences among users in terms of their routine behaviors. Identification of these temporal differences enables the target system to decide about the *optimal execution time*.

Figure 10 a. shows the distribution of FBPs detected in these three temporal segments, for all 35 users. The stack bar plot in Figure 10 a. has been ordered based on the number of FBPs detected between 0-8. This figure does not visualize FBP confidence. As previously stated, based on the users' labels, FBPs that have more than 20% accuracy are highly frequent behavioral patterns.

After this initial step, the second task is to identify if we can generalize users' characteristics on the described temporal

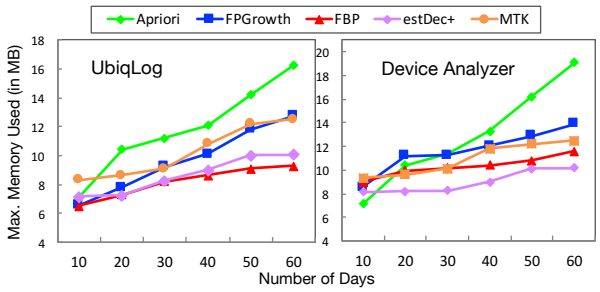

Fig. 8: Maximum heap allocation memory size (in MB) comparision between FBP and other algorithms on the smartphone.

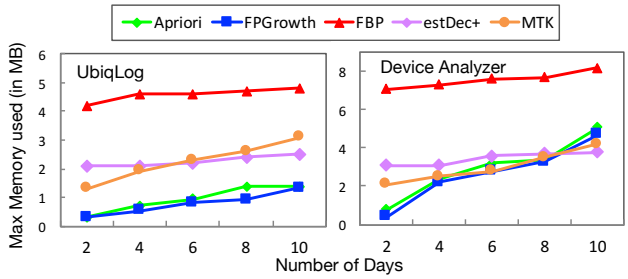

Fig. 9: Maximum heap allocation memory size (in MB) comparison between FBP and other algorithms on the smartwatch.

segment. In this instance, we have used a topic modeling approach, latent semantic indexing (LSA) [35], to cluster users based on their temporal FBPs, within their confidence. Our approach assumes users as *documents* and numbers of FBPs within their temporal segment plus confidence as *terms*. These are terms: `0-8 & <20%`, `0-8 & >20%`, `8-16 & <20%`, `8-16 & >20%`, `16-24 & <20%`, `16-24 & >20%`. Figure 10 (b) shows a multidimensional scaling [36] that has

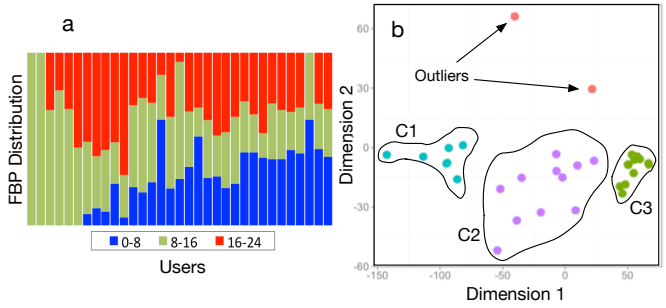

Fig. 10: (a) FBP distribution among users in three temporal segments (0-8, 8-16, 16-24). (b) Multidimensional scaling of clusters of users based on the temporal distribution of their FBPs. We have colored the cluster elements based on the characteristics of the users.

been performed on the results of our LSA clustering. The two red dots on the top are outliers, which have significantly different behaviors than other users. These results illustrates three clusters. C3 presents higher density and it include users who provide fewer FBPs during 16-24 with a larger and higher confidence FBPs during 8-16. C2 illustrates users who have an average number of FBPs (with both low and high confidence) distributed among two segments: 8-16 and 16-24. C1 presents the third cluster. This cluster has more identified FBPs at 0-8 temporal segment and fewer FBPs in the other two segments.

Understanding the temporal segment with the highest FBP detection rate enables the system to identify the best execution time for FBP detection. The fist conclusion we can draw is that our algorithms could be executed in the time segment that users have more FBPs (based on the user cluster) and not in two other time segments. Therefore, we preserve resources by not executing these algorithms frequently. The second conclusion is that, this clustering approach assists the system to reduce the search space through filtering data that is not being used for the FBP detection. In other words, if a system knows the target user's cluster, the FBP detection algorithms can be applied to only the related temporal segment(s) and not all of the segments.

5.4 Thresholds and Sensors Effects

This section first provides an overview about *sensors effectiveness* in FBP detection, based on the time of the day. Next, it reports about *parameter sensitivities*. Our approach is multivariate and sensor independent, but continuously collecting data and thus analyzing large amounts of information is a resource intensive process. Therefore, identifying the effective sensors in FBP detection, based on time of the day, could assist us in trimming the data source and thus not using all of the existing sources 24/7. To gain such an overview we report on sensors that have participated in FBPs, based on the *time of day*. Figure 11 a. shows the distribution of sensor data based on the time of the day, in the UbiqLog dataset. Since the number of WiFi and location logs are significantly larger than other sensors, WiFi and location have been shown separately in Figure 12 a. Figure 11 b. shows the identified FBPs based on the time of the day and the number of sensor appearances in FBPs. Similarly, Figure 12 (b) shows the number of WiFi and Location records (extracted from Algorithm 1) that have participated in the creation of FBPs.

For each sensor we have divided the quantity of that sensors in the FBP logs by the overall number of that sensors in the dataset. This provides us with the impact ratio (effectiveness) of the sensor for FBP identification. For instance, the effectiveness of the activity sensor for FBP detection can be calculated as: $2,174 \div 15,641 = 0.139$. 2,174 are the number of FBP logs that include activity sensors and 15,641 are the overall number of activity sensor logs in the dataset. In the UbiqLog dataset, Activity has the highest ratio, followed by SMS, Location state or movement (based on Algorithm 1), Application usage, Calls, WiFi and Bluetooth which has the lowest ratio.

It is notable that the ratio that we have calculated considers each sensor in abstract, with the creation of our FBPs being based on a combination of different sources. Therefore, we do not recommend focusing only on one sensor. Moreover, we can not generalize these findings to all other ubiquitous human-centric datasets. Bluetooth data collection, in the UbiqLog dataset, is based on scanning the environment every six minutes and logging all available Bluetooth proximities, which might not be applicable in other datasets. This Bluetooth data collection policy results in sparse Bluetooth logs, and thus Bluetooth data is ineffective in FBP detection. Activity and SMS have a significant role in FBP identification. At the time of running our data collection experiment, not all devices were equipped with the Google

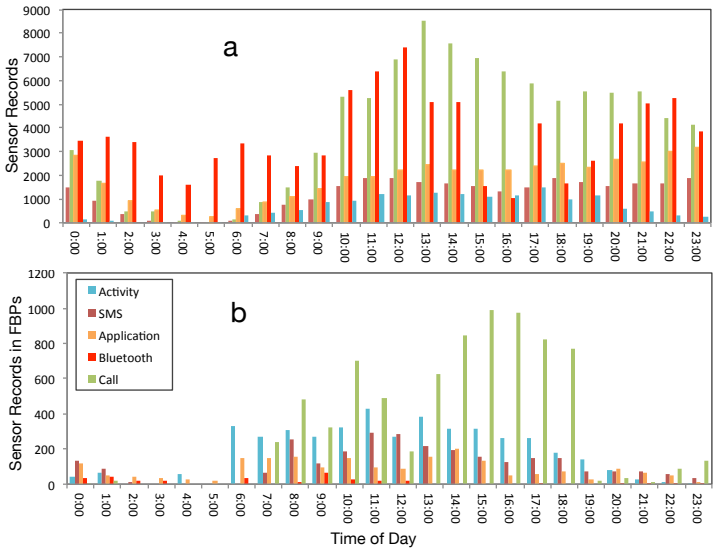

Fig. 11: (a) collected sensor data based on the time of day. (b) identified FBPs based on sensors and the time of the day.

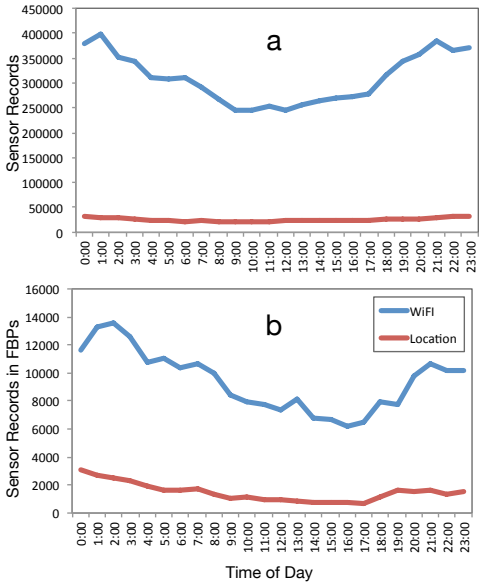

Fig. 12: (a) collected WiFi and Location logs (b) identified FBPs based on WiFi alone and the identified location state (not original location).

Play API, which has been used for the activity recognition. Nevertheless, a large number of FBPs have been created because of activity data.

The second evaluation has analyzed the impact of variables

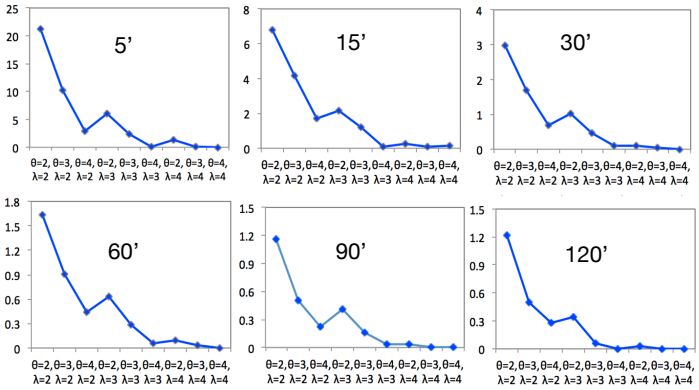

Fig. 13: Average number of FBP (X Axis) for each temporal granularities, based on different minimum entities threshold(θ) and lifetime confidences threshold(λ) (Y axis).

on FBP detection. "Minimum entities threshold", "lifetime confidence threshold" and "temporal granularity" are three configurable variables that have been used in our approach. We have tested our FBP detection and profiling algorithms within six different types of TGs: 5', 15', 30', 60', 90', 120' and 0', which is not using TG. Figure 13 shows the average number of detected FBPs for each TG, with different minimum entities thresholds (θ) and lifetime confidence threshold (λ) in the UbiqLog dataset. Figure 13 shows that increasing λ and θ detects a fewer number of FBPs. Moreover, it shows that the maximum limit for λ is four and setting theta to four is also leaning toward its maximum. Clearly, minimum for both λ and θ is two, because one means that they are not in use. As an example for setting these variables consider a scenario where a system tries to quantify a user's gym attendance behavior. If only detecting the location is enough then θ=1. However, the system may try to detect routine activities of the user inside the gym. In this case, both location and activity should be detected then θ=2).

However, results in Figure 13 show that increasing both λ and θ to more than three reduces the chance of detecting any FBPs. This is due to the fact that increasing these variables increases the precision, but lowers the probability of FBP identification. Based on the maximum number of identified motifs (θ =2) in Figure 13, we demonstrate that it is not feasible to model and predict human behavior 24 hours a day, via a smartphone. These findings are in line with [6], which argues that the smartphone's proximity to the user, restricts a 24/7 behavior observation. In addition to this, Figure 13 shows a possible maximum for the θ. Furthermore, there will be very few FBPs identified with a θ larger than three. The value of λ is associated with the number of identified FBPs, and similar to θ increasing the λ reduces the number of FBPs. Nevertheless, λ is not as effective as θ. Note that θ depends on the number of available sensors on the device. For instance, in the "insight for Wear" example we have used 2 for θ. This means two equal sensors in each temporal granularity represents a routine behavior. λ is also application dependent, we use 2 for λ with window size 3, to consider a behavior in a week as a routine behavior.

6 Related Work

A major contribution of this research is a generic mobile data mining system. We claim it is generic because of its multi-sensor support and application independence. Our secondary contribution is frequent itemset mining algorithms and their sub components such as analyzing the temporal aspect of human behavior. Moreover, we discuss algorithms for location estimation based on users' smartphone data. Therefore, three categories of related works have been studied: mobile data mining efforts that focus on device data collection (not 3rd party providers), frequent itemset mining algorithms, and location estimation from smartphone data.

6.1 Mobile Data Mining

Research that relies on collecting data from users' mobile devices is mostly application-specific and focuses on predicting one element of data (single sensor). For instance, a category of research explores activity recognition from accelerometer data [37], [38]. Recent approaches [37] have tried to employ a data dictionary and use semi-supervised learning to learn human activities. This makes the data mining process lightweight as well as scalable for implementation on mobile devices. MobileMiner [28] and ACE [9] are two works that are particularly relevant to our research. Both studies are very similar and consider the co-occurrence patterns in human behavior, via mobile phones, through association rule mining. Their approach is realistic in terms of deployment. However, they focus on co-occurrences of more than one data object. In contrast, we have identified FBPs and not just co-occurrences. Likewise, since we aim for human behavior detection we benefit from the temporality of behavior, and thus there is no need to have at least two data objects available for prediction (one is enough if the application uses $\theta = 1$). Another similar work is [10], which extracts users' routine behavior by identifying application usage correlation with time and location. This work transforms geographical coordinates based on the time of the day to "work" or "home". Our location transformation (movement state estimation) is more accurate than this transformation, and we include precise time of the day, while transforming the location.

6.2 Frequent Itemset Mining and Temporal Granularity

As it has been previously stated, our work digitally maps timestamps for human activities onto human temporal perception. The work most similar to ours is [39], which focuses on mining users' daily location patterns via trajectory mining and defines the TG as a day. Another approach for identifying daily behavior is within [10], who tries to match the daily location of users to the applications that they use. They converted a day into two segments (8:00-18:00 and 18:00-8:00) and model application usage in each of these segments.

It is important to note that we are dealing with temporal events that fit into the Allen's interval algebra [40], and so it is not about time series analysis [41]. Time series are dealing with continuous sequences of precisely timestamped data. For instance, [42] proposed a language for expressing temporal knowledge in time interval. The authors proposed TKSR (Time Series Knowledge Representation) representation as a resolution to resolve Allen's logic limitations such as handling the ambiguity that exists in overlaps of Allen's relation. This work proposes a five stage datamining approach based on TSKR representation, i.e. (Time Series Knowledge Mining) TSKM, which focuses on mining coincidences and partial orders. Our approach treats data in a similar fashion but it has focuses on frequent behavior mining.

Other scalable approaches which are fast and memory efficient. For instance, the DCI-CLOSED [43] algorithm uses depth-first visits of the search space and adopts a vertical bitmap representation of the dataset. The scope of our behavior identification is a time interval of a temporal granularity, therefore our algorithms does not consider if an itemset is a closed itemset, or not. Two other state-of-the-art algorithms have also been used in our experimental evaluation. MTK [31] is an itemset mining algorithm that operates based on the given memory constraint. It resolves the issue of creating the FP-Tree in the memory by providing a customized search δ-stair search, which limits the number of candidates that are generated-and-tested in each database scan. estDec+ [32] uses the compressible-prefix tree and to stay memory efficient it keeps only recent frequent itemsets and neglects the previous ones. Since it is both fast and memory efficient we use it in our evaluation as one of the state-of-the-art algorithms.

6.3 Location Estimation from Smartphone

There are several research benefits from utilizing smartphone location logs, i.e. GPS, WiFi and Cell-ID, to identify locations of interest and daily movement patterns. Reality mining [7], is one of the first efforts toward identifying behavior from smartphone contextual data. Their benchmark location dataset has been used widely in other works [16], [17], [18]. These works use Reality Mining location data to identify daily location change patterns. However, recently, the uncertainty of a realistic deployment has been taken into account and there are some works have been trying to support this uncertainty while mining for location data that has originated from unreliable sensors [18]. Semantically, location is the most valuable piece of information in digital human behavior identification, and therefore these studies map location onto human behavior. However, we believe that human behavior is not just based on changes in location, and studies should also include activities that are happening within the location. Therefore, our interpretation of human behavior is different from other works. Since our research can use all of the existing sensors on a device, it can be extended to any type of human behavior analysis application. In other words, we benefit from a *combination of sparse information sources* and not just one information source. There is another category of work, which uses trajectory of location [44], [45] for movement pattern prediction and classification. Since they use geographical coordinates their notion of location is different than ours.

7 Conclusion & Future Work

In this paper, we have proposed a *scalable* approach for daily behavioral pattern mining from multiple sensor information. This work has been benefited from two *real-world* datasets and users who use different smartphone brands. We use a novel *temporal granularity* transformation algorithm that makes changes on timestamps to mirror the human perception of time. Our *frequent behavioral pattern* detection approach is generic and not dependent on a single source of information; therefore, we have reduced the risk of uncertainty by relying on a combination of information sources to identify frequent behavioral patterns. Furthermore, our approach is lightweight enough that it can be run on small devices, such as smartwatches, and thus reduces the network and privacy cost of sending data to the cloud. Results of the experimental evaluation shows our algorithm outperforms the baseline and two state-of-the-art algorithms

in both execution time and accuracy. Moreover, converting raw timestamps to temporal granularities increase the accuracy of the FBP identification, which is influenced by different values of temporal granularity, the segment of the day and the sensor type. These findings assist the system in identifying the appropriate run time and sensor impact of the behavioral pattern identification.

In our future work, we are trying to model concept drift and its relation with forgetting or churn that is in the nature of human behavior. Moreover, we plan to compare the performance of the sliding window with the performance of the damped window.

REFERENCES

[1] S. Foell *et al.*, " Micro-navigation for Urban Bus Passengers: Using the Internet of Things to Improve the Public Transport Experience," *Urb-Iot '14*.

[2] R. Dobbins and R. Rawassizadeh, "Clustering of Physical Activities for Quantified Self and mHealth Applications," in *IUCC 15*.

[3] A. Campbell and T. Choudhury, "From Smart to Cognitive Phones," *IEEE Pervasive Computing '12*, vol. 11, no. 3, pp. 7–11.

[4] D. Ferreira *et al.*, "Understanding Human-Smartphone Concerns: a Study of Battery Life," *Pervasive Computing '11*.

[5] R. Rawassizadeh *et al.*, "Wearables: Has the Age of Smartwatches Finally Arrived ?" *Communications of the ACM '15*, vol. 58, no. 1, pp. 45–47.

[6] A. Dey *et al.*, "Getting Closer: An Empirical Investigation of the Proximity of User to Their Smart Phones," *UbiComp '11*.

[7] N. Eagle and A. Pentland, "Reality Mining: Sensing Complex Social Systems," *Personal and Ubiquitous Computing '06*, vol. 10, no. 4, pp. 255–268.

[8] N. Kiukkonen *et al.*, "Towards Rich Mobile Phone Datasets: Lausanne Data Collection Campaign," *ICPS '10*.

[9] S. Nath, "ACE: Exploiting Correlation for Energy-Efficient and Continuous Context Sensing," *MobiSys '12*.

[10] H. Ma *et al.*, "A Habit Mining Approach for Discovering Similar Mobile Users," *WWW '12*.

[11] R. Rawassizadeh *et al.*, "Ubiqlog: a generic mobile phone-based life-log framework," *Personal and Ubiquitous Computing '13*, vol. 17, no. 4, pp. 621–637.

[12] D. Wagner *et al.*, "Device Analyzer: Large-scale Mobile Data Collection," *SIGMETRICS Perform. Eval. Rev. '14*, vol. 41, no. 4, pp. 53–56.

[13] R. Rawassizadeh and A. Tjoa, "Securing Shareable Life-Logs," *SocialCom '10*.

[14] R. Rawassizadeh *et al.*, "Lesson Learned from Collecting Quantified Self Information via Mobile and Wearable Devices," *Journal of Sensor and Actuator Networks*, vol. 4, no. 4, p. 315, 2015.

[15] J. Paek *et al.*, "Energy-efficient Positioning for Smartphones Using Cell-ID Sequence Matching," in *MobiSys '11*.

[16] K. Farrahi and D. Gatica-Perez, "A Probabilistic Approach to Mining Mobile Phone Data Sequences," *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 223–238, 2014.

[17] J. Zheng and L. Ni, "An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data," *UbiComp '12*.

[18] T. Bhattacharya *et al.*, "Automatically Recognizing Places of Interest from Unreliable GPS Data Using Spatio-temporal Density Estimation and Line Intersections," *Pervasive and Mobile Computing '14*.

[19] N. Deblauwe and P. Ruppel, "Combining GPS and GSM Cell-ID Positioning for Proactive Location-based Services," in *MobiQuitous '07*.

[20] C. Zhou *et al.*, "Discovering Personal Paths from Sparse GPS Traces," in *JCIS '05*.

[21] N. Mamoulis *et al.*, "Mining, Indexing, and Querying Historical Spatiotemporal Data," in *KDD '04*.

[22] Y. Li *et al.*, "Mining Probabilistic Frequent Spatio-Temporal Sequential Patterns with Gap Constraints from Uncertain Databases," in *ICDM '13*.

[23] M. Ye *et al.*, "On the Semantic Annotation of Places in Location-based Social Networks," in *KDD '11*.

[24] D. Wang *et al.*, "Human Mobility, Social Ties, and Link Prediction," in *KDD '11*.

[25] S. Isaacman *et al.*, "Identifying Important Places in Peoples Lives from Cellular Network Data," in *Pervasive Computing '11*, pp. 133–151.

[26] R. Poidevin, "The Experience and Perception of Time," 2009, http://plato.stanford.edu/entries/time-experience.

[27] C. Bettini *et al.*, *Time Granularities in Databases, Data Mining, and Temporal Reasoning*. Springer, 2000.

[28] V. Srinivasan *et al.*, "Mobileminer: Mining your Frequent Patterns on Your Phone," in *UbiComp '14*.

[29] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," in *VLDB '94*, 1994, pp. 478–499.

[30] J. Han, J.Pei, and Y. Yin, "Mining Frequent Patterns Without Candidate Generation," in *SIGMOD '00*, 2000, pp. 1–12.

[31] K. Chuang *et al.*, "Mining Top-K Frequent Patterns in the Presence of the Memory Constraint," *The VLDB Journal*, vol. 17, no. 5, pp. 1321–1344, 2008.

[32] S. Shin *et al.*, "CP-tree: an adaptive synopsis structure for compressing frequent itemsets over online data streams," *Information Sciences*, vol. 278, pp. 559–576, 2014.

[33] C. Dobbins *et al.*, "The Big Data Obstacle of Lifelogging," in *WAINA '14*.

[34] P. Fournier-Viger *et al.*, "SPMF: a Java Open-source Pattern Mining Library," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 3389–3393, 2014.

[35] S. Deerwester *et al.*, "Indexing by Latent Semantic Analysis," *JASIS '90*, vol. 41, no. 6, pp. 391–407.

[36] J. Kruskal, "Nonmetric Multidimensional Scaling: a Numerical Method," *Psychometrika '64*, vol. 29, no. 2, pp. 115–129.

[37] S. Bhattacharya *et al.*, "Using Unlabeled Data in a Sparse-Coding Framework for Human Activity Recognition," *Pervasive and Mobile Computing*, vol. 15, pp. 242–262, 2014.

[38] O. Incel, *et al.*, "A Review and Taxonomy of Activity Recognition on Mobile Phones," *BioNanoScience '13*, vol. 3, no. 2, pp. 145–171.

[39] Y. Ye *et al.*, "Mining Individual Life Pattern Based on Location History," *MDM '09*.

[40] J. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[41] E. Keogh and S. Kasetty, "On the Need for Time Series Data Mining Benchmarks: a Survey and Empirical Demonstration," *Data Mining and Knowledge Discovery*, vol. 7, no. 4, pp. 349–371, 2003.

[42] F. Mörchen and A. Ultsch, "Efficient Mining of Understandable Patterns from Multivariate Interval Time Series," *Data Mining and Knowledge Discovery*, vol. 15, no. 2, pp. 181–215, 2007.

[43] C. Lucchese *et al.*, "Fast and Memory Efficient Mining of Frequent Closed Itemsets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 1, pp. 21–36, 2006.

[44] H. Tsai *et al.*, "Mining Group Movement Patterns for Tracking Moving Objects Efficiently," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 2, pp. 266–281, 2011.

[45] J. Lee *et al.*, "Mining Discriminative Patterns for Classifying Trajectories on Road Networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 5, pp. 713–726, 2011.

Reza Rawassizadeh is a post-doctoral associate within the Department of Computer Science at Dartmouth College. He received his BSc in Software Engineering. Master of Computer Science, and PhD in Computer Science in 2012, in University of Vienna, Austria.

Chelsea Dobbins is a senior lecturer within the Department of Computer Science at Liverpool John Moores University (LJMU). She received her BSc (Hons) in Software Engineering, and PhD in Computer Science, from LJMU in 2014.

Momeni Elaheh is a research associate of the Faculty of Computer Science at the University of Vienna. She received a Master of Computer Science from the Vienna University of Technology and PhD in Computer Science at the University of Vienna in 2014.

Joobin Gharibshah is a Ph.D. student in Computer Science at UCR. He received his B.Sc. in Computer Engineering (Software) from Iran University of Science and Technology, and his M.Sc. in Artificial Intelligence in 2012 from Sharif University of Technology.

Michael Pazzani is Vice Chancellor for research and Economic Development and also a professor of Computer Science at University of California, Riverside. He received his Ph.D. in Computer Science from UCLA and was an assistant, associate and full professor at the University of California, Irvine, where he also served as Chair of Information and Computer Science.