Published in final edited form as:
Pervasive Mob Comput. 2014 February 1; 10(Pt B): 138–154. doi:10.1016/j.pmcj.2012.07.003.

Activity Recognition on Streaming Sensor Data

Narayanan C Krishnan and **Diane J Cook**
School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752, USA

Abstract

Many real-world applications that focus on addressing needs of a human, require information about the activities being performed by the human in real-time. While advances in pervasive computing have lead to the development of wireless and non-intrusive sensors that can capture the necessary activity information, current activity recognition approaches have so far experimented on either a scripted or pre-segmented sequence of sensor events related to activities. In this paper we propose and evaluate a sliding window based approach to perform activity recognition in an on line or streaming fashion; recognizing activities as and when new sensor events are recorded. To account for the fact that different activities can be best characterized by different window lengths of sensor events, we incorporate the time decay and mutual information based weighting of sensor events within a window. Additional contextual information in the form of the previous activity and the activity of the previous window is also appended to the feature describing a sensor window. The experiments conducted to evaluate these techniques on real-world smart home datasets suggests that combining mutual information based weighting of sensor events and adding past contextual information into the feature leads to best performance for streaming activity recognition.

Keywords

streaming; online; real-time; activity recognition; mutual information

1. Introduction

Advances in pervasive computing have resulted in the development of unobtrusive, wireless and inexpensive sensors for gathering activity information, which when coupled with state of the art machine learning algorithms are critical to the development of a wide variety of applications. One such application area is that of smart environments where activity information is used to monitor and track the functional status of residents. A good number of on-going projects in smart environment and activity recognition such as the CASAS project [1], MavHome [2], PlaceLab [3], CARE [4] and the Aware Home [5] stand testimony to the importance of this research area. The need for the development of such technologies is underscored by the aging population[6], the cost of health care [7] and the importance that individuals place on remaining independent in their own homes [8]. Individuals need to be able to complete Activities of Daily Living (ADLs) such as eating, grooming, cooking,

Email address: ckn, cook@eecs.wsu.edu (Narayanan C Krishnan, Diane J Cook).

drinking and taking medicine, to lead a functionally independent life. Thus automating the recognition and tracking of these ADLs is an important step toward monitoring the functional health of a smart home resident, which has also been recognized by family and caregivers of Alzheimer's patients. This is the primary motivation behind much of the activity recognition research in smart environments.

Activity recognition (AR) is a challenging and well researched problem. The different approaches proposed in the literature differ primarily in terms of the underlying sensing technology, the machine learning models and the realism of the environment in which activity information was gathered. Irrespective of the sensing technology and machine learning model, literature is abundant with AR techniques that work extremely well on scripted or pre-segmented sequences of activity. While this is a first step toward developing AR, real-world deployment of these systems require AR techniques to work on streaming/online data among other scenarios such as concurrent and interleaved activity execution. This is also important in the context of developing assistive technologies for the elderly that can help them in completing ADLs (such as prompting systems [9], [10]). There is a need for online activity recognition techniques that can classify data as it is being collected which is the basis for tracking the progress of the activity. This is a challenging problem as data that completely describe an activity is not generally available in such situations and the algorithm has to rely on the partially observed data along with other contextual information to make a decision on the activity being performed.

The work presented in this paper attempts online AR on discrete binary motion sensor data obtained from real-world smart homes. The approach classifies every sensor event based on the information encoded in a sliding window of preceding sensor events. It explores both fixed static window size and dynamic varying window size, along with investigating modifications to the sliding window protocol that takes into account the temporal and spatial relationships between the different sensors. It also encodes the context of the window in terms of the classification probabilities of activities in the preceding window and the previously recognized known activity. This methodology is evaluated on data collected from three smart apartments over a period of six months. These datasets represent the activities performed by a single resident of the smart home. One of the facets of the work presented in this paper that sets it apart from other related work is the dataset that is used to evaluate the algorithms. Our dataset reflects the complexities of unconstrained real-world data that cannot be observed in other datasets. We present the first application of the sliding window method for dealing with discrete motion sensor events. Another factor that distinguishes our work from the rest is the inclusion of sensor events that do not belong to any of the known activity labels for performance evaluation. This is a common problem that one faces when scaling the AR approaches to real-world settings. It makes the first attempt at trying to understand how the state of the art techniques perform in complex real-world settings, where the subject is living in their natural habitat and conducting their daily routine with no instructions of what so ever from the researchers. The focus on the evaluation is to study the effectiveness of the activity models, trained and tested on data collected from the same smart home. We are not trying to study how well the activity models generalize across different apartment layouts and different residents.

The rest of the paper is organized as follows. Section 2 discusses briefly the related work on AR. A discussion on the different methodologies for processing streaming data is presented in Section 3. The sliding window methodology adopted in this paper along with the accompanying modifications are described in Section 4. Section 5 presents the experimental setup for evaluating the proposed methodology along with a description of the smart apartment dataset. The results are presented and discussed in Section 6. Section 7 summarizes the work presented in the paper along with providing directions for future work.

2. Related Work

The goal of activity recognition is to recognize human physical activities from data collected through different sensors. It is a well researched area and hence there exists a number of approaches for activity recognition [11]. These approaches vary depending on the underlying sensor technologies that are used for gathering the activity data, the different machine learning algorithms used to model the activities and the realism of the environment in which the activity data is collected and AR is performed.

2.1. Sensors for AR

Advances in pervasive computing have seen the development of a wide variety of sensors that are useful for gathering information about human activities. Wearable sensors such as accelerometers are commonly used for recognizing activities that are primarily defined by ambulatory movements (such as walking, running, sitting, standing, lying down and falling) as demonstrated by earlier efforts [12][13]. More recently researchers are exploring smart phones equipped with accelerometers and gyroscopes to recognize ambulatory movements and gesture patterns [14][15]. Most of these approaches have been able to recognize activities primarily characterized by movements in real time [16] through a sliding window protocol. Since the movement information related to activities is typically well represented within a window of the data from accelerometers with a high sampling rate, a sliding window based approach is appropriate for recognizing these activities in real-time.

Environment sensors such as infrared-based motion detectors or reed switches based door sensors have also been used for gathering information about a more general set of ADLs such as cook, leave home, sleep, eat, etc; as explored by others [17][18][19][20]. These sensors are adept in performing location based activity recognition in indoor environments just as GPS is used for outdoor environments [21]. Some actives such as wash dishes, take medicine, use phone, etc are characterized by unique objects of interaction. Researchers have explored the usage of RFID tags and shimmer sensors for tagging these objects of interaction and thus be able to perform AR. For instance, Philipose et al.[22] use the count of objects of interaction obtained through the activation of RFID tags to decipher the activities being performed in the environment and Palmes et al. [23] mine the web to determine which objects are essential for recognizing a particular activity and use this information to build activity models.

Researchers have also used data from video cameras monitoring and recognizing different activities [24][25]. The use of video cameras for AR is very prevalent in security related applications. However, their usability in the context of smart homes for monitoring the activities of residents is debatable as study subjects uniformly believe that it intrudes into their privacy. There are many other challenges with video based AR such as illumination variations, occlusion and background changes that make it somewhat impractical in certain scenarios. The data used for the experiments conducted in this paper are primarily from binary discrete passive IR sensors that can be easily embedded in a smart environment.

2.2. Machine Learning Approaches for AR

There have been a number of machine learning models that have been used for AR akin to the variety of the sensor technologies. These can be broadly categorized into template matching or transductive techniques, generative and discriminative approaches. Template matching techniques employ k-NN classifier on either distance computed between a test window and training windows through Euclidean distance in the case of fixed window size [26] or dynamic time warping in the case of varying window size [27]. Generative approaches to AR such as simple naive Bayes classifiers, where the activity samples are

modeled using Gaussian mixtures, have yielded promising results for offline learning of activities when large amount of data is available for training [19][28][29]. Generative probabilistic graphical models such as hidden Markov models and dynamic Bayesian networks that are known for their representational power have also been used to model activity sequences. HMMs have been used to model the activity sequences from data obtained from both wearable accelerometers for recognizing dietary activities [30] and from environmental sensors for recognizing ADLs [31]. HMM is employed as a post processing tool by others [32] to smooth out the recognition results of an AdaBoost classifier for detecting human activities using data from on-body sensors. Discriminative approaches that model the boundary between the different classes have also been popular for AR. Decision trees such as C4.5 [33] are natural choices for models that classify various activities based on different threshold or properties of the associated features [28],[34]. Meta classifiers based on boosting and bagging have also been experimented for AR [29][35]. Support vector machine based activity models have been experimented for AR using accelerometer data [13]. Discriminative probabilistic graphical models such as conditional random fields have been explored for AR using motion sensor data [18],[19].

There are also a number of unsupervised activity discovery methods that mine for frequent sensor sequences [36], or discontinuous activity patterns [37]. An activity discovery algorithm based on compression of sensor data is presented in [38]. Most of these approaches for AR predominantly work on pre-segmented activity sequences that have been collected in controlled settings. More recently Wang et al. [39] propose a hierarchical approach for activity recognition from data collected through body sensor networks. The fundamental difference in our approach and that of Wang et al. lies in the underlying sensing technology that is used for collecting activity data. While in the proposed approach we use binary motion sensors that are triggered by human motion, Wang et al use inertial sensors that have a constant sampling rate, which is well suited for sliding window based on time. Secondly, the evaluation of Wang et al's approach is conducted on data that is collected in a laboratory setting does not reflect the performance of the approach in unconstrained real-world settings. We evaluate our approach on real-world data that is collected from the house of volunteers.

2.3. Experimental Settings for AR

The most common type of experiment is to ask subjects to perform a set of scripted activities, one at a time in a laboratory setting [13][31][28][39]. In this case the activities are well segmented, which facilitates the researchers to focus on the task of mapping the pre-segmented sensor sequences to activity labels. With the ability of the algorithms to perform well in these scripted settings, researchers are now more focused on algorithms for more realistic and complex activity tasks. These include recognizing activities that are performed with embedded errors, interleaved activities, concurrent activities performed by multiple residents and activities with multiple goals. However the data for these settings are still restricted to a laboratory/supervised setting where the activities are performed in a controlled manner [40][41]. The next leap taken by the researchers is to recognize activities in unscripted settings, an example of which is a smart home while residents perform their usual daily routine, without being given explicit instructions on how to perform the activity. However, these approaches include an offline annotation process that make use of human annotators to analyze, segment and label the data. Activity recognition is then performed only on the pre-segmented data [19] [17]. While this approach brings activity recognition closer to practical everyday usage, to make it even more realistic, the activity recognition algorithms will have to move away from pre-segmented sequences and focus directly on the sensor streams. The work presented in this paper is an attempt in this direction, where AR is performed on streaming data.

Activity recognition from sensor streams has been explored with wearable sensor technology [16]. The data from wearable sensors such as accelerometers are divided into chunks of equal length, which are then classified as one of the activities by the classifier. This is made possible by the fact that sensors provide a continuous sequence of data through time (constant sampling rate) and most of the activities discerned from these sensors have sufficient information characterizing them within a small time window. It is an interesting problem to try to extend this approach to relatively sparse sensor streams, where sensor events are triggered only because of human activities, such as motion sensor sequence from a smart home. This is the primary feature that distinguishes the work presented in this paper with other related work.

3. Processing Streams

This section briefly describes the three common approaches in the literature for processing of streaming data. These three approaches are discussed in the context of the smart home dataset used in this paper. The sensors embedded in the smart apartments are primarily motion and door sensors that are in two states - 'ON' and 'OFF' for the motion sensors and 'OPEN' and 'CLOSED' for the door sensors. These sensors have a varying sampling rate (when they switch to the 'ON' state) which is dependent on the human activity. Figure 1 is an abstract illustration of the sequence of these sensor firings (represented as the vertical lines). The data chunks that each of the three approaches analyze are then presented one below the other in the figure. The underlying activity sequence that results in these sensor firings is A_1,A_4,A_2,A_3. Each activity results in different number and type of sensor firings.

3.1. Explicit Segmentation

Some methodologies [42] adopt a two step approach for streaming activity recognition. In the first step the streaming sensor events are segmented into chunks, each chunk possibly corresponding to an activity and perform the classification of each of the chunk in the second step. An example of this process is illustrated in Figure 1. The segmentation process results in chunks C_1,C_2, . . . , C_6. The first thing to notice is that the process does not result in exact activity boundaries (Activity A_1 is broken down to chunks C_1 and C_2), which is a common property of the segmentation algorithms. While the segmentation process could lead to chunks representing specific activities, it has some drawbacks. The first and the foremost problem is trying to find the appropriate chunk size for learning the activity models during the training phase. Typically a pre-segmented sequence of sensor events that correspond to an activity is used to train the classifier. However during testing, since the chunks do not necessarily represent the entire sequence of sensor events for a particular activity, the performance of the classifier is thus lowered. Secondly, the approach has to wait for future data to make a decision on past data rendering it a somewhat non-streaming approach. Furthermore because of its dependency on the future data, large temporal gaps in successive sensor events which are realistic in every day routines will result in the approach waiting for a long time to make a decision on the past events. Finally the two level approach also leads to additional complexity issues of the segmentation process to be dealt with such as splitting and merging chunks. If the activities have very distinct boundaries, then this will be a good approach, which is not the usual scenario.

3.2. Time based windowing

The second approach to handling streaming data would be to divide the entire sequence of sensor events into equal size time intervals as illustrated in Figure 1 by the chunks denoted by T_1, T_2, . . . , T_9. This approach has been adopted by many researchers [43][13][39]. This technique offers a simpler approach to learn the activity models during the training phase over the explicit segmentation approach. It further reduces the computational complexity of

the explicit segmentation process. This is a good approach when dealing with data obtained from sensors that operate continuously in time. Data for every time interval is always guaranteed in such a scenario. This is a common approach with accelerometers and gyroscopes, where data is sampled at a constant rate from the sensors. However, one has to deal with the problem of selecting the optimal length of the time interval. If a very small interval is chosen, there is a possibility that it will not contain any relevant activity information for making any useful decision. If the time interval is too wide, then information pertaining to multiple activities can be embedded into it and the activity that dominates the time interval will have a greater influence in the classification decision. This problem manifests itself when dealing with sensors that do not have a constant sampling rate. In the current context of motion and door sensor events, it is very likely that some time intervals do not have any sensor events in them (e.g., T_6 in Figure 1) . Then heuristics have to be developed to extend the activity occurring in the previous time intervals to the current time interval.

3.3. Sensor event based windowing

The third approach for sensor stream processing is to divide the sequence into windows containing equal number of sensor events. This is illustrated in Figure1 by the chunks S_1, $S_2, \ldots, S_{26}$. It is evident that the windows appear to vary in their duration. This is fine considering that during the performance of activities, multiple sensors could be triggered, while during silent periods, there will not be many sensor firings. The sensor events preceding the last event in a window define the context for the last event. This method too has some inherent drawbacks. For example, consider the chunk S_{26}, in Figure 1. The last sensor event of this chunk corresponds to the beginning sensor event of activity A_4. There is a significant time lag between this event and its preceding sensor event. The relevance of all the sensor events in this chunk on the last event might be small if the time lag is large. Thus treating all the sensor events with equal importance will not be a good approach. In the presence of multiple residents, sensor firings from two different activities performed by the different residents will be grouped into a single chunk, thereby introducing conflicting influences for the classification of the last sensor event. While by itself this approach may not be alluring, modifying it to account for the relationship between the sensor events is a good method to process the stream of sensor events. This approach offers computational advantages over the explicit segmentation process and does not require future sensor events for classifying past sensor events. The methodology presented in the paper adopts this approach for processing sensor streams in conjunction with three modifications that capture the relation between the sensor events in a window and between multiple windows. The results obtained on real-world datasets discussed in the paper advocate the advantage of this methodology.

4. Methodology

This section discusses in detail the windowing methodology adopted in the work along with the modifications. Let us begin by defining some of the notations used to describe the approach. Let $s_1, s_2, \ldots, s_N$ represent the sequence of sensor events. Example of these events are motion and door sensor events as depicted from the sample data in Figure 2 collected from one of our smart home testbeds. Associated with each sensor event is the calendar day, time of the day, sensor status. As a preprocessing step, we mapped the actual sensor tags to their functional areas (a total of 15). The sensor locations in Figure 2 correspond to these functional areas. The objective of the proposed approach is to classify every single sensor event with a corresponding activity label to the best possible extent. Activity information of the individual sensor event is essential in some of the applications

such as prompting systems that will be built on this foundation. We first describe the basic sensor windowing approach followed by the modifications.

4.1. Sensor windowing

A simple approach to provide context to a single sensor event is to consider the sensor events that have preceded it. Formally, the sequence $s_1, s_2, \ldots s_N$ is divided into windows of equal number of sensor events $S_1, S_2, \ldots, S_M$ and the S_i window be represented by the sequence $[s_{i-\Delta s}, s_i]$. The value of the parameter Δs varies depending on the experimental context. It can be derived through an empirical process by studying the effect of the different values of Δs on the performance of the classification system. Among the different factors that influence the value for the parameter Δs is the average number of sensor events that span the duration of different activities. At one end of the spectrum are activities such as 'Leave Home' that are defined by rapid firing of a small set of sensors as illustrated in Figure 2, while at the other end is the activity 'Sleep' that continues for hours, but typically leads to occasional firing of one or two sensors as illustrated in Figure 2. Ideally the number of sensor events within the window should be sufficient enough to define the context of the last sensor event. Heuristics such as the average length of the longest activity can be used to bound the window size. Later in this section we describe a method to dynamically determine the window size.

Once the sensor window S_i is defined, the next step is to transform this window into a feature vector that captures its information content. We perform this step by constructing a fixed dimensional feature vector xi explicitly capturing the time of the first and last sensor events, the temporal span of the window S_i and a simple count of the different sensor events within the window. With 15 different sensors, the dimension of the x_i will be 18. Each x_i is tagged with the label y_i of the last sensor event (s_i) in S_i. Each label y_i corresponds to an activity class. A collection of x_i and the corresponding y_i then become the training data that is fed into a classifier to learn the activity models in a discriminative manner. This will be our **Baseline** approach against which we compare the proposed enhancements.

One of the problems associated with windowing based on sensor events for sensors that do not have constant sampling rate is that windows could contain sensor events that are widely spread apart in time. An illustration of this problem is presented in Figure 3(a). This is an example of a sequence of sensor events from our dataset. Notice the the time stamp of the last two events in this sequence. There is a gap of nearly one and half hours between these sensor events. All the sensor events that define the context of the last event have occurred in the 'distant' past. Thus in the absence of any weighting scheme, even though the sensor event corresponding to the end of the 'Personal hygiene' activity occurred in the past, it has an equal influence on defining the context of the event corresponding to 'Enter home'. In order to overcome this problem, we propose to use a time based weighting scheme (described in Section 4.2) which takes into account the relative difference in the time the sensors were triggered.

In situations when the sensor event corresponds to the transition between two activities (or in other settings when multiple activities are performed by more than one resident in parallel), the events occurring in the window might not be related to the sensor event under consideration. An example of this situation is illustrated in Figure 3(b). This particular sequence of sensor events from one of the testbeds represents the transition from the 'Personal Hygiene' activity to the 'Leave Home' activity. Notice that all the initial sensor events in the window come from a particular functional area of the apartment namely the *Bathroom*, whereas the second set of sensor events are from an unrelated functional area of the apartment namely *FrontDoor*. While this certainly defines the context of the activity, since the sensors from a particular activity dominate the window, the chances for a wrong

conclusion about the last sensor event of the window are higher. We try to overcome this problem by defining a weighting scheme based on a mutual information type measure between the sensors as described in Section 4.3.

4.2. Time Dependency

As described earlier, with a fixed length sliding window over sensor events, it is possible for two sensor events that are spread apart in time to be a part of the same window. In order to reduce the influence of such sensor events for deciding the outcome of the last sensor event, we use a time based weighting factor for each sensor event relative to the time of the last sensor event.

Formally, let $\{t_{i-\Delta s}, t_{i-\Delta s+1}, \ldots, t_i\}$ represent the timing of the sensor events in the i^{th} window. We take into account the difference in the time for each sensor event with respect to t_i, for computing the feature vector describing the window. In particular we use an exponential function to compute the weights. Thus, the contribution of sensor event k, within the interval Δs to the feature vector describing the ith window is given as

$$C(i,j)=exp(-\chi(t_i - t_k)) \quad (1)$$

The simple count of the different sensor events within a window is now replaced by a sum of the time based contributions of each of the different sensor event within the window. The value of χ determines the rate of decay of the influence. Figure 4 shows the effect of the χ on the rate of decay. If $\chi > 1$, then it is only the sensor events that are temporally very close to the last event that contribute to the feature vector. With $0 < \chi < 1$, the feature vector is under the influence of a temporally wider range of sensor events. When $\chi = 0$, the temporal distance has no influence on the feature vector, making it a simple count of the different sensor events. We henceforth refer to this approach as **SWTW** that stands for Sensor Windows Time Weighting.

4.3. Sensor Dependency

The work presented in this paper uses a mutual information based measure between the sensors, to reduce the influence of sensor events from very different functional areas on the feature vector defining the last sensor event within a window. Mutual information is typically defined as the quantity that measures the mutual dependence of two random variables. In the current context, each individual sensor is considered to be a random variable that has two outcomes, namely 'ON' and 'OFF'. The mutual information or dependence between two sensors is then defined as the chance of these two sensors occurring consecutively in the entire sensor stream. If S_i and S_j are two sensors, then the mutual information between them *MI(i, j)* is defined as

$$MI(i,j)=\frac{1}{N}\sum_{k=1}^{N-1}\delta(s_k,S_i)\delta(s_{k+1},S_j) \quad (2)$$

where

$$\delta(s_k,S_i)=\begin{cases} 0 & \text{if } s_k \neq S_i \\ 1 & \text{if } s_k=S_i \end{cases} \quad (3)$$

The summation term takes a value of 1, when the current sensor is S_i and the subsequent sensor is S_j. If two sensors are adjacent to each other, such that triggering of one sensor is

most likely to trigger the other sensor, then the mutual information between these two sensors will be high and similarly, if the sensors are far apart such that they do not often occur together, then the mutual information between them will be low. Note that the computation of mutual information using this bi-gram model depends on the order of occurrence of the sensors.

The mutual information matrix is computed offline using the training sensor sequence. It is then used to weigh in the influences of the sensor events in a window while constructing the feature vector. Similar to the time based weighting, each event in the window is weighted with respect to the last event in the window. Thus instead of the count of different sensor events, it is the sum of the contribution of every sensor event based on mutual information that defines the feature vector. We denote this approach as Sensor Windows Mutual Information (**SWMI**) for future references.

4.4. Past Contextual Information

The methods described previously only take into account the sensor events in the current window that is being analyzed. There is no information about the past that is encoded into the feature vector. The information about the activity in the previous window, or the previous occurrence of an activity are important factors that determine the activity in the current window. In our dataset, there are certain activities that have a well defined past activity; 'Enter Home' is an instance of such an activity that occurs only after the 'Leave Home' activity. Thus adding the past activity information to the feature vector defining the activity in the current window, will enhance the description of the current window.

One can simply add the ground truth of the previous window and the previous occurring activity to the current sensor window, but then the approach would not generalize well as we do not have ground truth about the past activity information during online recognition. Thus, one has to rely on the model predictions of the previous window to obtain the past activity information, which is similar to a semi-supervised learning paradigm. We incorporate this into our learning problem in two steps as illustrated in Figure 5. In the first step, activity models are learned by a classifier using training data that does not contain the past activity information. Each of the training instances are then fed into this activity model to obtain the probability of each window corresponding to the different activities. In the second step, these classification probabilities of the immediately preceding sensor window are appended to the feature vector describing the current sensor window along with the last predicted activity (not the activity in the immediate preceding window). In the example illustrated in the Figure 5, let us consider the sensor window S_{i+2}. The feature vector describing this window is appended with the probability of classification of the S_{i+1} sensor window, along with the activity that occurred last, which in this case is am. These newly appended feature vectors are then used to train another activity model. During the test phase, the feature vector describing the test window is fed into the first activity model; the probability outputs of this model are appended to the feature vector and the new feature vector is passed to the second model to obtain the predicted activity. We call this technique as Previous Window and Previous Activity (**PWPA**) for future references.

4.5. Dynamic Window Size

The previously described approaches employ a fixed window size for computing the feature vectors. An important challenge with this approach is identifying the optimal window size. Various heuristics such as the mean length of the activities and sampling frequency of the sensors can be employed to determine the window size. However the best approach would be to automatically derive the window size using a data-driven approach.

We use a probabilistic approach to derive the window size for every single sensor event. We begin with first defining a fixed number window sizes $\{w_1, w_2, ..., w_L\}$, where $w_1 = \min\{ws(A_1), ws(A_2), ..., ws(A_M)\}$ and $w_L = \text{median}\{ws(A_1), ws(A_2), ..., ws(A_M)\}$. $ws(A_m)$ corresponds to the mean window size of activity *Am*. The intermediate window sizes between w_1 and w_L are obtained by dividing this interval into equal length bins. After identifying the possible window sizes, we then compute the most likely window size for an activity A_m

$$w* = argmax_{w_l} \{P(w_l/A_m)\} \quad (4)$$

We also determine the probability ($P(A_m/s_i)$) of an activity A_m being associated with the sensor s_i. Thus given the sensor identifier s_i for the event under consideration, we can determine the most likely activity A^* associated with it as

$$A* = argmax_{A_m} \{P(A_m/s_i)\} \quad (5)$$

Thus for the sensor event s_i the optimal window size can be determined by combining the two equations according to the following factorization

$$w* = max_{w_l} P(w_l/s_i) = max_{w_l}[P(w_l/A_m) * P(A_m/s_i)] \quad (6)$$

The different probability vectors are computed using the training data. The window sizes for the sensor events in the train and test data are then computed using these probabilities. Considering that there are no significant changes to the routine of the residents of our smart apartments and that the training data typically consists of over 400K sensor events, we believe that there will not be significant differences to the probability vectors when computed using train and test data independently. We call this approach as Dynamic Window **DW** for future references.

4.6. Time Windows

We also evaluate our proposed approaches against the time window approach that is commonly found in the literature dealing with wearable sensors such as accelerometers. While there are different ways of defining the time windows, to align with our goal to classify individual sensor events, we define time windows for every sensor event. Thus for a sensor event s_i, the time window consists of all sensor events that fall within the time interval of the fixed duration starting from the time stamp of s_i. The sensor events of a particular time window are then aggregated to form the feature vector using the same process that was adopted for sensor windows. The time window is labeled using the activity associated with the last sensor event of the time window. We refer to this approach as Time Window **TW** for future discussion.

4.7. Handling the 'Other' class

Most of the current AR approaches ignore the sensor events that do not correspond to any of the known activity classes. To elaborate further, activity models that are trained for 'Cook', 'Eat' and 'Sleep' are experimented on data points that correspond to these three activities only. However this does not correspond to a real-world setting, where data points can correspond to other well defined activities such as 'Bathing' or transitions between different activities. Typically data points from these other activity classes tend to dominate the real-world data as illustrated by the high number of samples corresponding to the 'Other Activity' in Table 1. Current AR approaches tend to ignore the presence of these other

activity classes while evaluating the performance of the algorithms. However, in this work we include the sensor events from these 'Other Activity' classes as well to determine the most realistic activity recognition performance.

There are two methods to handle the 'Other' class events. The first method relies on the probability of classification obtained from models that have been trained only on events belonging to the known classes. A threshold on this probability determines if the event belongs to the other class. While this is a simple approach, it is hard to set the threshold as there is no learning involved in determining its value. In the second approach an explicit model for the other class events is learned by training the classifier on the events that do not fall under any of the known classes. The increase in the computational complexity of this method due to explicit learning of a new class can be forgone by considering that it is an offline process.

Explicitly modeling the other class events brings the activity recognition system one step closer to stream processing and hence is an important step. Table 1 shows that nearly half of the sensor events in each of the smart apartment belong to the other class, further motivating the need to learn a model for this class, which is the approach adopted in this paper.

5. Experiments

5.1. Dataset

Each smart environment is treated as an intelligent agent that perceives the state of its residents and the physical surrounding using sensors and acts on the environment using controllers in such a way that specified performance measures are optimized [44].

We test the ideas described in this paper on sensor event datasets collected from three smart apartment testbeds which we denote as *B1*, *B2* and *B3*. Figure 6 shows the floor plan and sensor layout for the three apartments. Each of the smart apartments housed an elderly resident. During the six months that we collected data in the apartments, the residents lived in these apartments and performed their normal daily routines. Figure 6 shows the occurrences of activities in each of the testbeds for a sample of the data. Note that each of the individuals had a fairly consistent daily routine.

Each of the smart home residents completed a generic questionnaire that asked them to mark the time they performed the activities under consideration. Human annotators analyzed a 2D visualization of the sensor events and used the information from the questionnaires to mark the beginning and ending for each occurrence of the 11 activities listed in Figure 6. Each smart home dataset was annotated by different human annotators. The annotations reflect the activity associations for the sensor events as perceived by the annotator based on the ground truth provided by the smart home resident. Thus there are possibilities for annotation errors. While the labeling process is cumbersome and prone to error, as part of the future work we will explore strategies to compensate and correct these errors.

The characteristics of each of these datasets are presented in Table 1. Reflective of the real-world nature of the dataset, the imbalance in the number of data samples for the different activities can be observed; with nearly 50% of the data samples belonging to the 'Other' class.

5.2. Classifier Model

In this paper we use support vector machines (SVM) as the choice for the classifier for learning the activity models. In the past we have successfully tried other models such as naive Bayes, hidden Markov models (HMM) and conditional random fields (CRF) [19].

While the discriminative CRF yielded better performance than naive Bayes or HMM, the enormous training cost of CRF made us choose SVM over it. We used the LibSVM implementation of Chang et al [45]. This implementation uses a one vs one SVM classification paradigm that is computationally efficient when learning multiple classes with class imbalance.

For computing the classification probabilities, the margin of the SVM classification was converted into probability values using Platt's scaling by fitting a logistic function [46]. A five fold cross-validation strategy was employed to obtain the generalized performance of the model on the three datasets. A radial basis function kernel was used with a width parameter of 1. The penalty parameter was set at 100. These are the default parameter values for the LibSVM classification implementation. The data samples were normalized before being fed into the classifier. Since we are primarily interested in the performance on the activities excluding the 'Other' activity, we computed the predefined activity classification accuracy as well as the *F score* as measures to compare the performance of the different methods.

The predefined activity classification accuracy is computed as follows. Let the total number of sensor windows associated with a predefined activity A_m (such as 'Cook') be denoted as N_{A_m} and the number of correctly classified windows for this predefined activity be TP_{A_m}. Then the predefined activity classification accuracy is defined as

$$Accuracy = \sum_{m=1}^{|A|} \frac{TP_{A_m}}{N_{A_m}} \quad (7)$$

where $|A|$ are the total number of predefined activities excluding the 'Other' activity. Let P and R represent the precision and recall for activity Am, then the *F score* for this activity is computed as

$$F\ score = 2 * \frac{P * R}{P + R} \quad (8)$$

6. Results and Discussion

In this section we present the results obtained on the three data sets using the different proposed techniques. These techniques and their notations have been summarized in Table 2 for clarity.

We begin by studying the performance of the Baseline approach of sliding window of sensor events for different number of sensor events per window. The results for this experiment are summarized in Table 3. It is interesting to note that the performance is poor when the length of window is greater than or equal to median of all the activities (presented in the second column of Table 3). Furthermore, the performance drops significantly beyond 20 sensor events per window for B1 and B2; much before the median value. It can be observed that the performance peaks between 10 and 20 sensor events per window. We choose the number of sensor events in a window to be the higher value, 20 as our modifications to the baseline method ensures that the relevant information within this window is captured to define the context for the last event of the window.

Having set the number of sensor events per window, we first check the performance of the baseline approach with time based weighting of sensor events(SWTW). The summary of these experiments are presented in Table 4. We did not explore values of χ greater than 1, as

they do not change the weights significantly. We varied the value of χ from 1 to 2^{-6} to give non-zero weights to sensor events between 1 and 64 seconds. This means that with $\chi = 1$, the sensor window consists of events that happen within 1 second of last sensor event of the window and similarly with $\chi = 2–6$ the sensor events within the window within 26 seconds of the last sensor event have non-zero weights. The accuracy for $\chi = 1$ is the lowest as it captures a very small number of sensor events preceding the last event of the window. However as we decrease the value of χ, information from a larger number of preceding sensor events are being used to characterize the sensor window leading to an increase in the classification accuracy. This improvement occurs only till a certain value of χ, which corresponds to sensor events within 64–128 seconds preceding the last event of the window, after which the performance drops. There is no change in the accuracy beyond this temporal span. Comparing the results obtained from the time based weighting scheme and the baseline method leads to mixed observations. While there is a significant improvement of 10% for B3, B1 shows an improvement of only 4% and the performance deteriorates in B2 by 3%. This indicates that time-based weighing scheme of sensor events within a window is not uniformly effective.

We compare the effectiveness of the time based weighting method against a method that determines sensor windows based on time interval (TW). We varied the time interval from 5 seconds to 60 seconds. Prior literature [39] suggests 15 seconds to be the optimal time interval. The results of this experiment are presented in Table 5. It can be observed that the accuracies do not vary significantly across the different time intervals. For testbeds B1 and B3, the accuracies are comparable to that obtained by SWTW approach with $\chi = 2^{-3}$. However the TW approach results in markedly improvement in the accuracy for testbed B2. While this is promising, our experiments with other approaches result in an improvement over the TW approach.

The MI based similarity measure between the different sensors for the each of the three testbeds is illustrated in Figure 7. Since we encode the actual sensor identifiers by the functional areas, it is evident from the figure that each of the functional areas are very dissimilar to each other. Furthermore the relatively strong diagonal elements indicate that higher chances of self transition of sensors, instead of transitioning from one sensor to another. Since each of the sensors is triggered by the human motion, it implies within the current context that the residents of the testbed tend to remain at a single location in their home more often than moving around. There are a couple of subtle observations that can be made from Figure 7. For example consider the similarities between the sensors 5 and 6 for B1. These two sensors correspond to the *front door* and *kitchen* sensors that are geographically close to each other in the testbed (refer Figure 6 B1). This implies that whenever the resident is entering the testbed; in addition to triggering the front door sensor, the resident is also likely to trigger the *kitchen* sensors. However the *kitchen door sensors*(sensor number 7) do not get triggered. Another subtle observation is the relatively high similarity between the *medicine cabinet* sensor (sensor number 13) and *kitchen* (sensor number 6) and the *kitchen door* sensors (sensor number 7) for B1. This is because the resident of this testbed stores the medicines in the kitchen. Thus when he/she takes the medication, they are likely to trigger the other *kitchen* sensors as well.

Modifying the baseline approach by using the MI measure for weighting the sensor events boosted the overall performance uniformly across the three testbeds by an average of 7% ($p < 0.01$) with respect to the baseline. These accuracy values are presented in Table 6. We also inlcude the best accuracies obtained by Baseline, SWTW and TW approaches for the different parameter choices. In comparison to the time based sensor weighting scheme, the MI based scheme outperforms only for testbed B1 and B2. There is a marginal dip in the performance for testbed B3. However the consistent improvement in the accuracy of

predefined activities over the baseline approach suggests the context of the last sensor event of a sliding window can be enhanced by considering the relationship between the different sensors. Further it is interesting to note that combining the MI based scheme with the time based weighting scheme with the best performing χ factor (SWTW+SWMI) resulted in a performance that is closer to the time based weighting scheme. The results for this experiment are presented in Table 6. It can be seen from the accuracy values that the combination model was more influenced by the time based weighting scheme. This result is understandable considering that the combination scheme gives equal importance to both the sets of weights and as a result, the set of weights that are numerically significant (time based weights) tend to dominate the other set (MI based weights).

The accuracies of the predefined set of activities for the third proposed modification that integrates the previous contextual information in the form of classification probabilities of the previous window and previous recognized activities (PWPA) are presented Table 6. Here too, there is an improvement in the performance by an average of 4% ($p < 0.01$) over the baseline approach. However, this approach does not match up to the performance of MI based weighting scheme. Finally we combine this method with MI based weighting scheme. These results for this approach (SWMI+PWPA) are presented in Table 6. It can be noticed that combining these two methods results in a marginal improvement in the performance over SWMI approach. However this is the highest performance obtained among all the techniques for B1 and B2.

The results for the DW approach that used dynamic window sizes is presented in the last row of Table 6. It is evident that this approach performs better than the Baseline method that used a fixed window size. This clearly indicates the advantage of using varying window size. However the resulting performance is not better than the SWMI+PWPA approach indicating scope for further improvement.

While measuring the accuracy of predefined activities is one metric to evaluate the algorithms, the skewed data distributions for the different activity classes necessitates other measures such as Fscore. The average *F score* over all the activities obtained for the different methods is summarized in Table 6 in parenthesis. It can be observed that SWMI +PWPA approach increases the *F score* over the Baseline approach by 10% for B1 and B2 and about 8% for B3. The *F score* obtained by SWMI+PWPA is comparable to the SWMI approach. These results are along the lines of what we had observed when comparing the accuracies of the predefined activities.

Going further we wanted to study which activities benefited by the proposed modifications. For this we plotted the individual *F score* for each activity that is summarized in Figure 8. We observed that ‘Leave Home’ and ‘Enter Home’ activities benefited the most by adopting the SWMI+PWPA approach. This resulted in improving the *F score* on an average by 16% for ‘Leave_Home’ and 41% for ‘Enter_Home’ activity across the three testbeds. The significant improvement for ‘Enter_Home’ is understandable as it has a well defined past context. The best performance for these activities was observed using the SWMI+PWPA approach.

The variation in the performance of the different methods for each activity across the three testbeds can also be observed from Figure 8. There are some activities like ‘sleep’, ‘personal hygiene’ and ‘bathing’ that are recognized relatively better than the rest of the activities. ‘Leave Home’ is the most difficult activity to discern uniformly across all the three testbeds. This can be attributed to the fact that these activities are performed in well-defined locations and/or static time during the day. For example, the activity ‘bathing’ occurs in the morning

in the ‘bath tub’ region of the testbed. Each testbed has a specialized sensor that monitors movement in the ‘bath tub’ region and therefore, acts as a robust marker for the activity.

An interesting observation can be made with respect to the ‘Eating’ activity. This activity has high *F score* for testbeds B2 and B3, and very low values for B1. While this is a universal activity performed by all the residents in all the three testbeds, the results suggests that residents in testbeds B2 and B3 performed this activity in a structured manner, meaning the activity ‘Eating’ consistently took place in the ‘Dining Room’. However the annotations for testbed B1 suggest that the resident carried out the activity in other locations of the apartment as a result of which it was often misclassified.

There is no significant change in the *F scores* for the ‘Other’ activity with different approaches across all the three testbeds. Furthermore the *F scores* for this activity average around 0.8, which is relatively high suggesting the ability of the current techniques to handle the data from this class. However notice from Table 1 the skewness towards the number of samples in the ‘Other’ class. Nearly 50% of the sensor windows belong to the ‘Other’ class. Thus a high true positive rate can boost the *F score* for this class. We computed the confusion matrix for each of the testbed using the approach that resulted in the best classification accuracy of predefined activities(SWMI+PWPA) to study the impact of the large ‘Other’ activity class. These are summarized in the form of normalized confusion matrices presented in Figure 9. Normalization was performed row-wise. Blocks that are darker imply less confusion between the corresponding activities and vice versa for the lighter blocks. The lighter diagonal entires indicate a high true positive rate for each of the activities. The large number of darkly shaded blocks dominating the rest of matrix except the last column and diagonal entires suggests that the proposed technique results in low confusion amongst the known set of activities. However, the last column of each of the three confusion matrices is signficantly lighter than the rest of the matrix, indicating that many sensor windows corresponding to known activities are being misclassified as ‘Other’ activity.

The proposed approach took on an average 4 days to learn the different activity models for each technique. Though this value is high, it does not have an impact on real-time recognition as training is typically performed offline. All the approaches were able to classify test samples at the rate of more than 100 samples per second implying online recognition.

7. Summary, Limitations and Future Work

In order to provide robust activity related information for real-world applications, researchers need to design techniques to recognize activities in real-time from sensor data. Activity recognition approaches have so far experimented on either a scripted or pre-segmented sequence of sensor events related to activities. In this paper we propose and evaluate a sliding window based approach to perform activity recognition in an on line or streaming fashion; recognizing activities as and when new sensor events are recorded. To account for the fact that different activities can be best characterized by different window lengths of sensor events, we incorporate the time decay and mutual information based weighting of sensor events within a window. Additional contextual information in the form of the previous activity and the activity of the previous window is also appended to the feature describing a window. These techniques are evaluated on three real-world smart home datasets collected over a period of 6 months. While each of these modifications show improvement over the baseline approach, we observe that combining mutual information based weighting of sensor events and adding past contextual information into the feature leads to best performance for streaming activity recognition.

While the recognition accuracies of this sliding window approach over the entire dataset is lower than the sensor windows corresponding to only known activity sequences, it is a promising step in the direction of developing online activity recognition. A limitation of the proposed approach is its inefficiency in modeling the 'Other' activity class. The current approach associates all sensor windows that do not correspond to any of the known activity as a single 'Other' activity resulting in skewness in the data. The activity models learned from this categorization are inherently biased due to this skewness as evidenced by the results that show a high degree of confusion between windows corresponding to known activities and the 'Other' activity. Reducing this confusion by exploring other ways of modeling the 'Other' activity class is a future direction that we plan to pursue. In the current evaluation methodology, the train and test sensor windows are drawn from the samples of the same smart apartment. As part of the future work, we will also evaluate the effectiveness of the proposed approach on train and test data being sampled from different smart apartments.

Acknowledgments

We would like to acknowledge support for this project from the National Science Foundation (NSF grant CNS-0852172), the National Institutes of Health (NIBIB grant R01EB009675), and the Life Sciences Discovery Fund.

References

1. Cook D, Schmitter-Edgecombe M, Crandall A, Sanders C, Thomas B. Collecting and disseminating smart home sensor data in the casas project. Proceedings of the CHI Workshop on Developing Shared Home Behavior Datasets to Advance HCI and Ubiquitous Computing Research. 2009
2. Cook, DJ.; Youngblood, M.; Heierman, E.; Gopalratnam, K.; Rao, S.; Litvin, A.; Khawaja, F. Mavhome: An agent-based smart home; Proceedings of the International Conference on Pervasive Computing; 2003.
3. Intille S, Larson K, Tapia EM, Beaudin J, Kaushik P, Nawyn J, Rockinson R. Using a live-in laboratory for ubiquitous computing research. Proceedings of Pervasive. 2006:349–365.
4. Krse, BJA.; Kasteren, TLMV.; Gibson, CHS.; den Dool, TV. Care: Context awareness in residences for elderly; Proceedings of Sixth International Conference of the International Society for Gerontechnology; 2008.
5. Kidd, CD.; Orr, RJ.; Abowd, GD.; Atkeson, CG.; Essa, IA.; Mac-Intyre, B.; Mynatt, E.; Starner, TE.; Newstetter, W. The aware home: A living laboratory for ubiquitous computing research; Proceedings of the Second International Workshop on Cooperative Buildings; 1999.
6. Vincent G, Velkoff V. The next four decades - the older population in the united states: 2010–2050. US Census Bureau. 2010
7. House A, Fox-Grage W, Gibson M. State by state long term health care costs. AARP. 2009
8. Gross J. A grass-roots effort to grow old at home. The New York Times. 2007
9. Pollack ME, Brown LE, Colbry D, McCarthy CE, Orosz C, Peintner B, Ramakrishnan S, Tsamardinos I. Autominder: an intelligent cognitive orthotic system for people with memory impairment. Robotics and Autonomous Systems. 2003; 44:273–282.
10. Das, B.; Chen, C.; Seelye, A.; Cook, D. An automated prompting system for smart environments; Proceedings of the International Conference on Smart Homes and Health Telematics; 2011.
11. Kim E, Helal A, Cook D. Human activity recognition and pattern discovery. IEEE Pervasive Computing. 2010; 9:48–53. [PubMed: 21258659]
12. Maurer, U.; Smailagic, A.; Siewiorek, D.; Deisher, M. Activity recognition and monitoring using multiple sensors on different body positions; Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks; p. 113-116.
13. Krishnan, NC.; Panchanathan, S. Analysis of low resolution accelerometer data for human activity recognition; International Conference on Acoustic Speech and Signal Processing, ICASSP; 2008.

14. Gyorbiro N, Fabian A, Homanyi G. An activity recognition system for mobile phones. Mobile Networks and Applications. 2008; 14:82–91.
15. Kwapisz, JR.; Weiss, GM.; Moore, SA. Activity recognition using cell phone accelerometers; Proceedings of the International workshop on Knowledge Discovery from Sensor Data; 2010. p. 10-18.
16. Krishnan, NC.; Colbry, D.; Juillard, C.; Panchanathan, S. Real time human activity recognition using tri-axial accelerometers; Sensors Signals and Information Processing Workshop; 2008.
17. Logan, B.; Healey, J.; Philipose, M.; Tapia, EM.; Intille, S. A long-term evaluation of sensing modalities for activity recognition; Proceedings of the International Conference on Ubiquitous Computing; 2007.
18. van Kasteren, T.; Krose, B. Bayesian activity recognition in residence for elders; Proceedings of the IET International Conference on Intelligent Environments; p. 209-212.
19. Cook D. Learning setting-generalized activity models for smart spaces. IEEE Intelligent Systems (to appear).
20. Tapia EM, Intille SS, Larson K. Activity recognition in the home using simple and ubiquitous sensors. Proceedings of Pervasive. 2004:158–175.
21. Liao, IL.; Fox, D.; Kautz, H. Location-based activity recognition using relational Markov networks; Proceedings of the International Joint Conference on Artificial Intelligence; 2005. p. 773-778.
22. Philipose M, Fishkin KP, Perkowitz M, Patterson DJ, dieter Fox, Kautz H, Hahnel D. Inferring activities from interactions with objects. IEEE Pervasive Computing. 2004; 3:50–57.
23. Palmes P, Pung HK, Gu T, Xue W, Chen S. Object relevance weight pattern mining for activity recognition and segmentation. Pervasive and Mobile Computing. 2010; 6:43–57.
24. Hongeng S, Nevatia R, Bremond F. Videobased event recognition: activity representation and probabilistic recognition methods. CVIU. 2004:129162.
25. Brdiczka O, Crowley JL, Reignier P. Learning situation models in a smart home. IEEE Transactions on Systems, Man, and Cybernetics, Part B. 2009; 39
26. Foerster F, Smeja M, Fahrenberg J. Detection of posture and motion by accelerometry: a validation study in ambulatory monitoring. Computers in Human Behavior. 1999; 15:571–583.
27. Alon J, Athitsos V, Yuan Q, Sclaroff S. A unified framework for gesture recognition and spatiotemporal gesture segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2008; 31:1685–1699. [PubMed: 19574627]
28. Bao L, Intille SS. Activity recognition from user-annotated acceleration data. Pervasive Computing. 2004:1–17.
29. Ravi, N.; Dandekar, N.; Mysore, P.; Littman, ML. Activity recognition from accelerometer data; IAAI'05: Proceedings of the 17th conference on Innovative applications of artificial intelligence; 2005. p. 1541-1546.
30. Amft O, Troester G. On-body sensing solutions for automatic dietary monitoring. IEEE Pervasive Computing. 2009; 8:62–70.
31. Singla G, Cook D, Schmitter-Edgecombe M. Recognizing independent and joint activities among multiple residents in smart environments. Ambient Intelligence and Humanized Computing Journal. 2010; 1:57–63.
32. Lester, J.; Choudhury, T.; Kern, N.; Borriello, G.; Hannaford, B. A hybrid discriminative/ generative approach for modeling human activities; IJCAI'05: Proceedings of the 19th international joint conference on Artificial intelligence; 2005. p. 766-772.
33. Quinlan, JR. C4.5: Programs for machine learning. Morgan Kaufmann Publishers; 1993.
34. Chen, C.; Das, B.; Cook, D. A data mining framework for activity recognition in smart environments; Proceedings of the International Conference on Intelligent Environments; 2010.
35. Wang, S.; Pentney, W.; Popescu, AM.; Choudhury, T.; Philipose, M. Common sense based joint training of human activity recognizers; IJCAI'07: Proceedings of the 20th international joint conference on Artificial intelligence; 2007. p. 2237-2242.
36. Gu T, Chen S, Tao X, Lu J. An unsupervised approach to activity recognition and segmentation based on object-use fingerprints. Data and Knowledge Engineering. 2010

37. Pei, J.; Han, J.; Asl, MB.; Pinto, H.; Chen, Q.; Dayal, U.; Hsu, MC. Prefixspan: Mining sequential patterns efficiently by prefix projected pattern growth; Proceedings of International Conference on Data Engineering; 2001. p. 215-226.

38. Rashidi P, Cook D, Holder L, Schmitter-Edgecombe M. Discovering activities to recognize and track in a smart environment. IEEE Transactions on Knowledge and Data Engineering. 2011; 23:527–539. [PubMed: 21617742]

39. Wang L, Gu T, tao X, Lu J. A hierarhical approach to real-time activity recognition in body sensor networks. Journal of Pervasive and Mobile Computing. 2011

40. Singla G, Cook D, Schmitter-Edgecombe M. Tracking activities in complex settings using smart environment technologies. International Journal of BioSciences, Psychiatry and Technology. 2009; 1:25–35.

41. Hu, DH.; Pan, SJ.; Zheng, VW.; Liu, NN.; Yang, Q. Real world activity recognition with multiple goals; Proceedings of the Tenth International Conference on Ubiquitous Computing; 2008. p. 30-39.

42. Junker H, Amft O, Lukowicz P, Troster G. Gesture spotting with body-worn inertial sensors to detect user activities. Pattern Recognition. 2008; 41:2010–2024.

43. Kasteren TLMV, Englebienne G, Krse B. An activity monitoring system for elderly care using generative and discriminative models. Journal Personal and Ubiquitous Computing, Special Issue on Pervasive Technologies for Assistive Environments. 2010:489–498.

44. Cook D, Das SK. Smart environments: Technology, protocols and applications. 1995

45. Chang C-C, Lin C-J. LIBSVM: a library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2011; 2:1–27.

46. Platt J. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. Advances in Large Margin Classifiers. 1999:61–74.

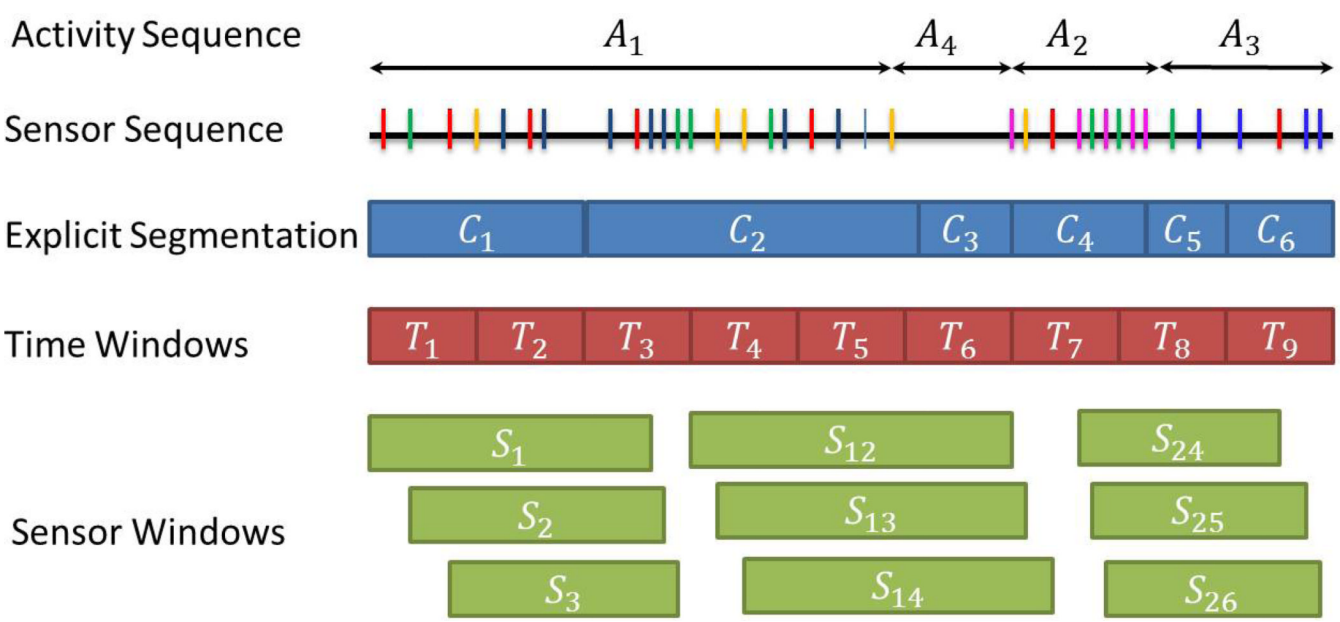

Figure 1.
Illustration of the different approaches for stream processing. The different motion/door sensor firings are depicted by the colored vertical lines. The color indicates the type and location of the sensor that was switched on. The sensor windows are obtained using a sliding window of length 10 sensor events.

```
2009-07-17 15:56:55.484001 FrontDoor OPEN Leave_Home begin
2009-07-17 15:56:58.078001 FrontDoor OFF
2009-07-17 15:56:59.796001 FrontDoor ON
2009-07-17 15:57:03.703001 FrontDoor OFF
2009-07-17 15:57:04.406001 FrontDoor CLOSE Leave_Home end
...
2009-07-17 22:14:11.250001 Bedroom ON Sleep begin
2009-07-17 22:14:15.546001 Bedroom OFF
...
2009-07-17 22:20:09.921001 Bedroom OFF
2009-07-18 00:11:32.703001 Bedroom ON
...
2009-07-18 00:14:19.296001 Bedroom OFF
2009-07-18 01:10:37.640001 Bedroom ON
...
2009-07-18 01:57:30.375001 Bedroom OFF
2009-07-18 03:34:18.000001 Bedroom ON
...
2009-07-18 05:59:16.234001 Bedroom OFF
2009-07-18 06:34:37.687001 Bedroom ON
...
2009-07-18 06:35:13.296001 Bedroom OFF
2009-07-18 06:35:16.859001 Bedroom ON Sleep end
```

Figure 2.
An example of a sequence of sensor events from one of the smart home testbeds.

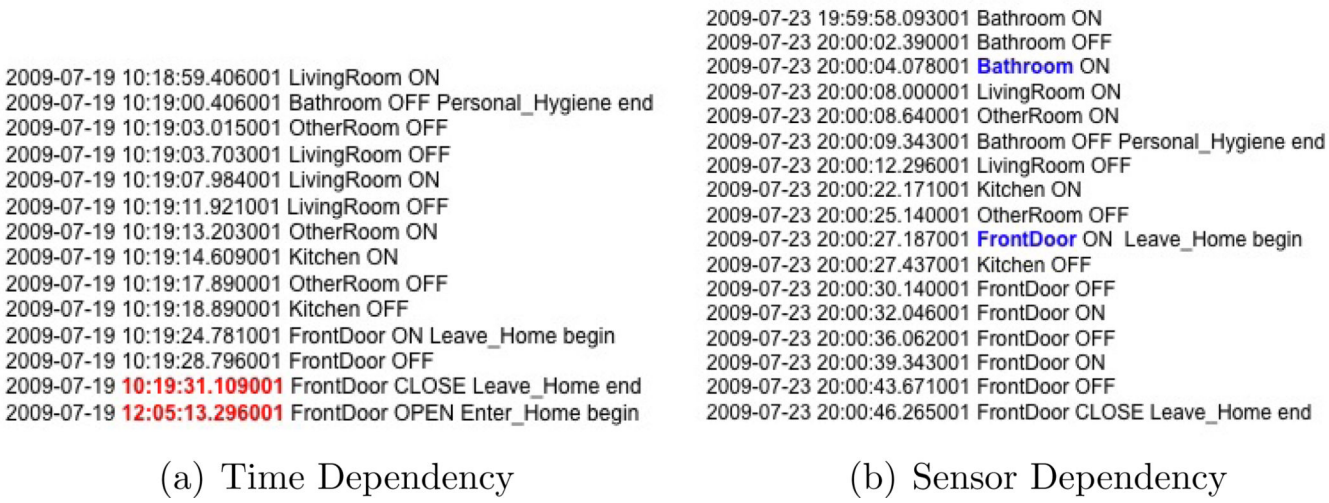

Figure 3.
Illustration of the different dependencies when considering sensor based windows.

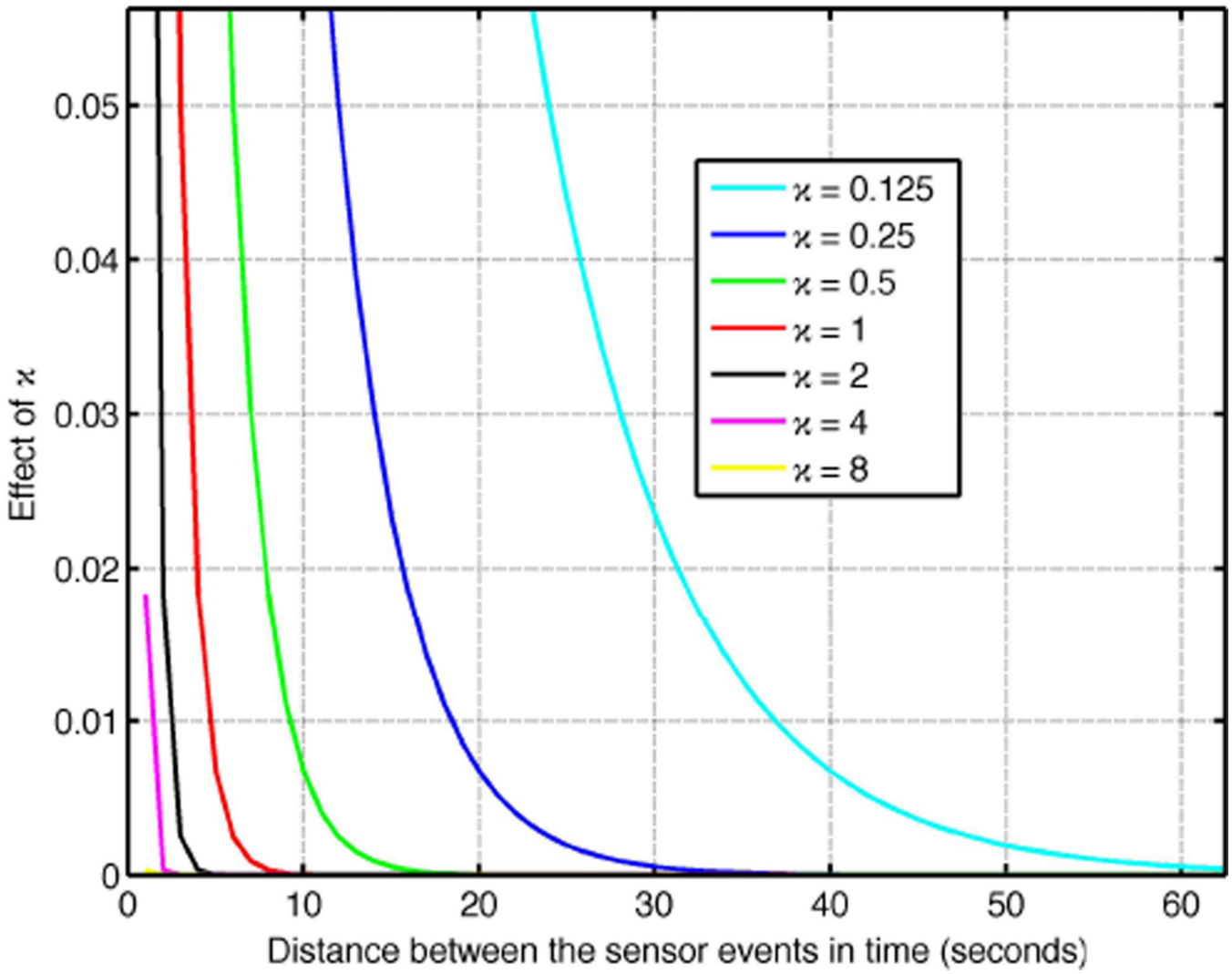

Figure 4.
Effect of χ on the weights.

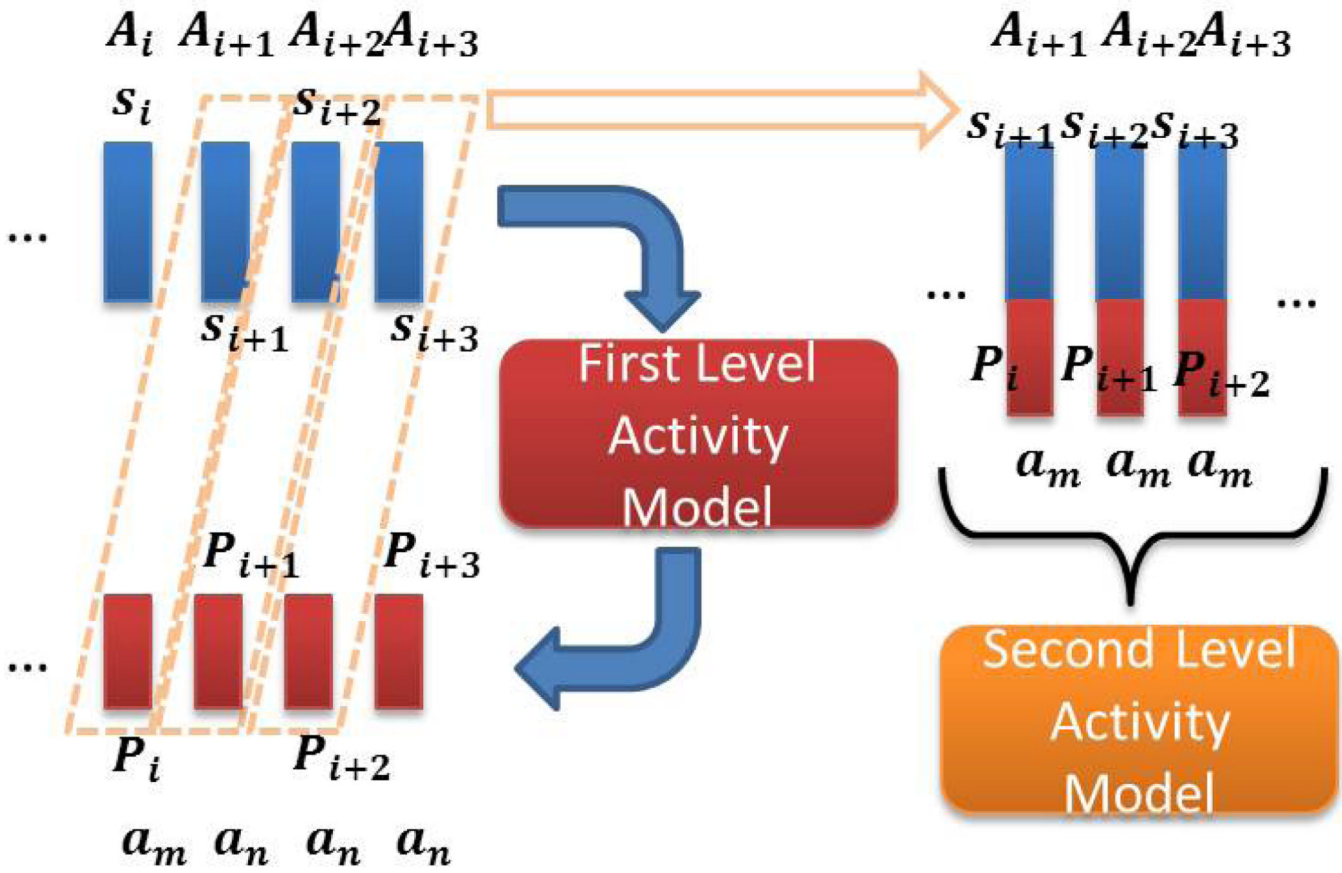

Figure 5.
Illustration of the two phase learning process that includes past contextual information

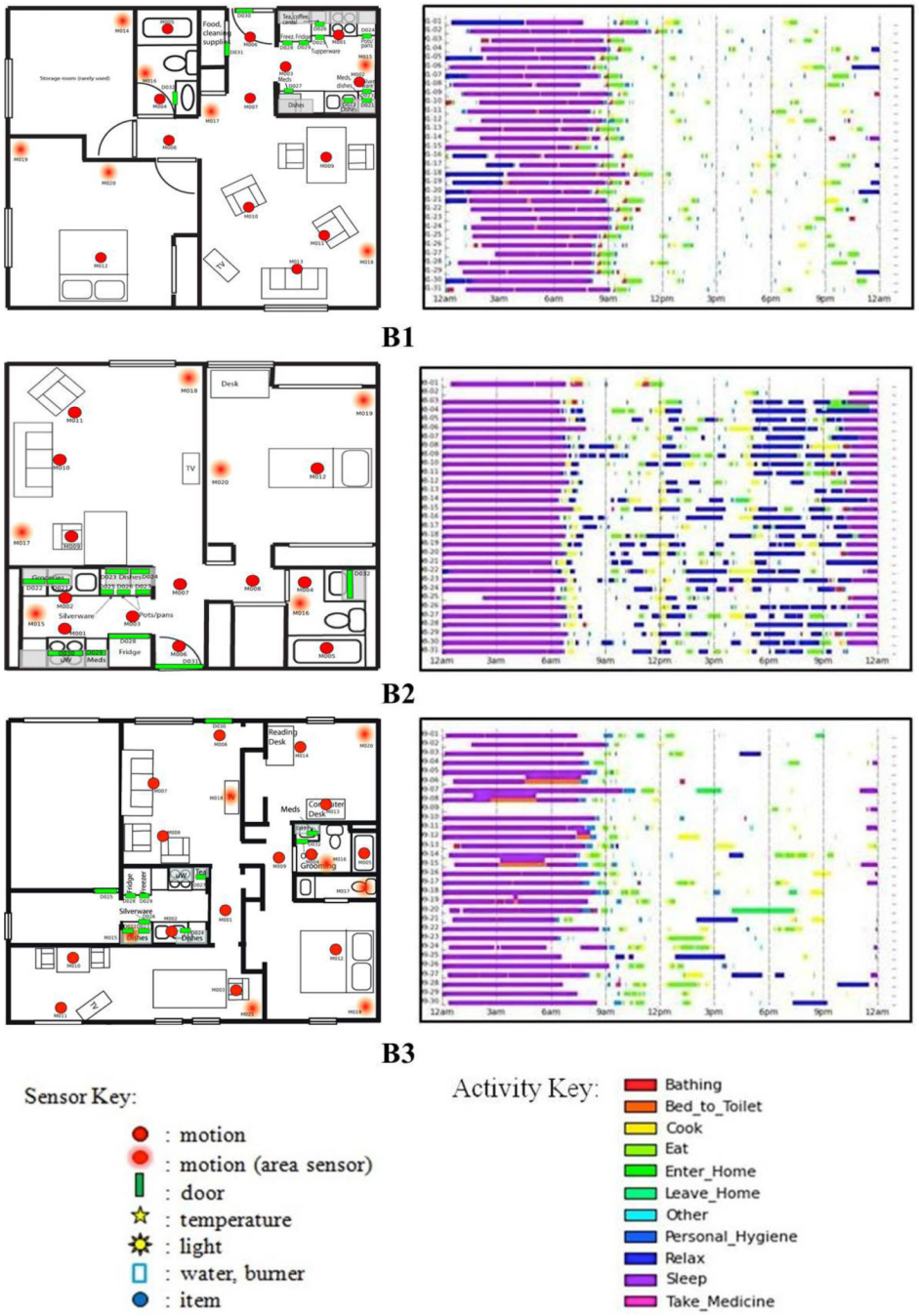

Figure 6.
The occurrences of the different activities in each of the smart apartment.

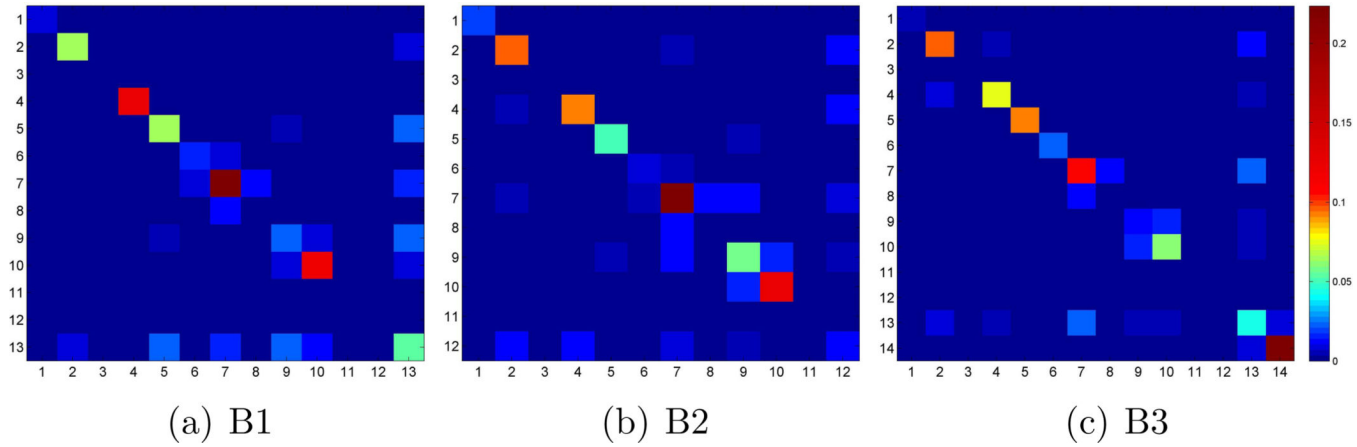

Figure 7.
Mutual Information between every pair of sensors for the three smart apartment testbeds.

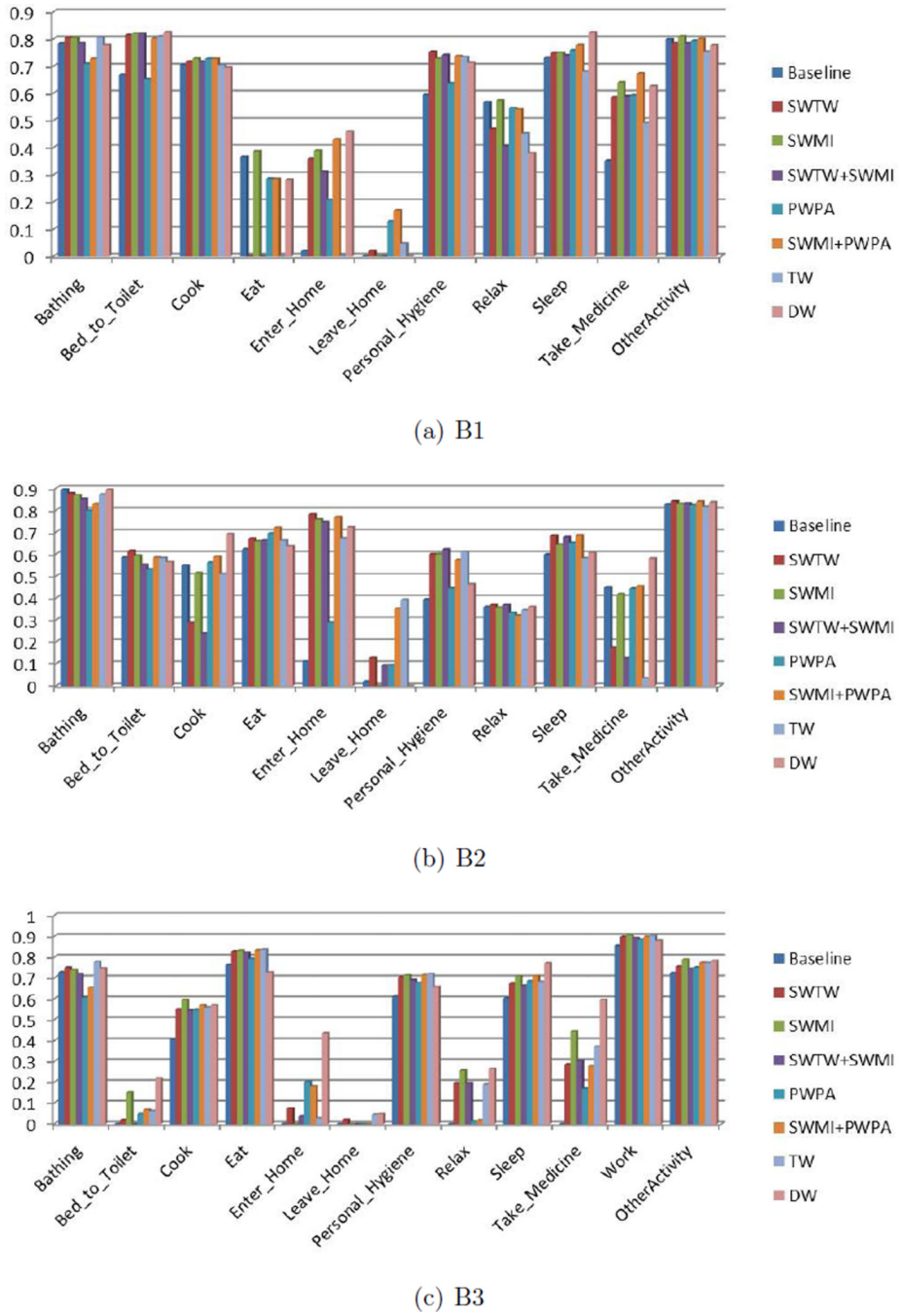

Figure 8.

F scores for the individual activities for each testbed as obtained by the different approaches.

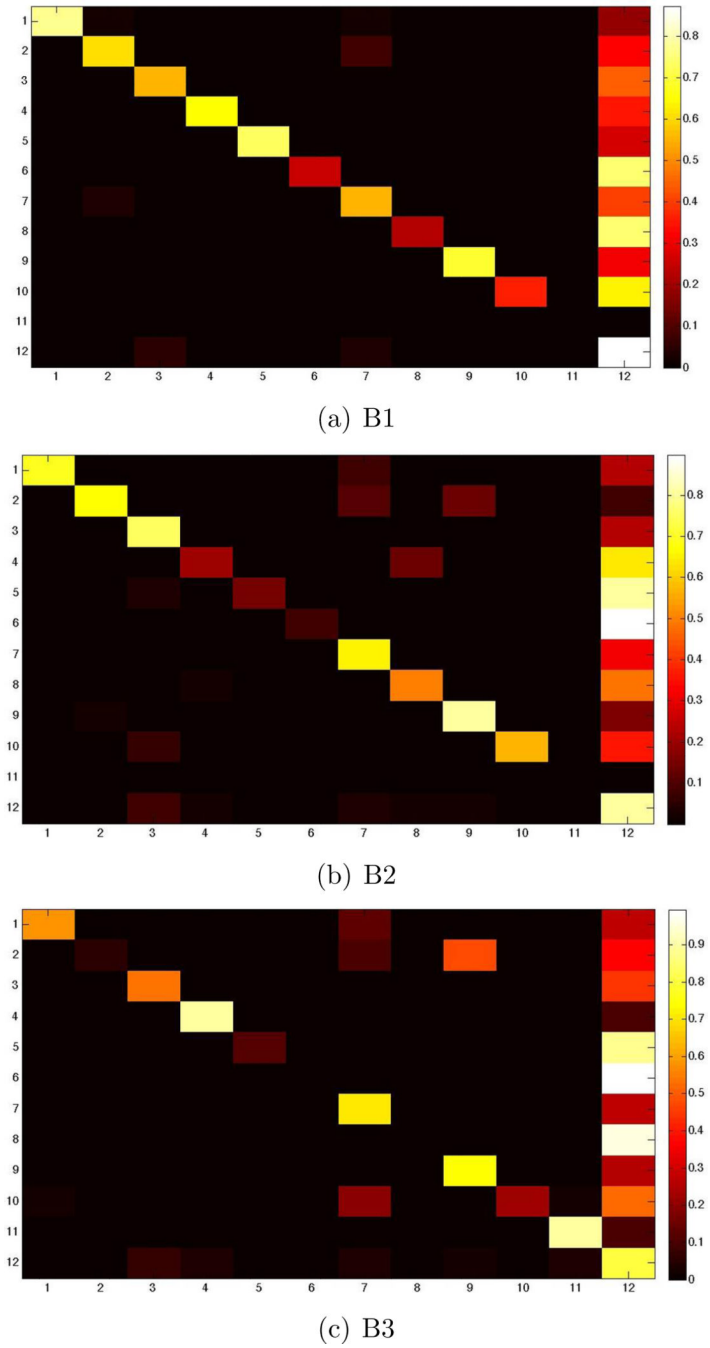

Figure 9.
Normalized confusion matrix for each of the three testbeds. The labels are 1) Bathing, 2) Bed to Toilet, 3) Cook, 4) Eat, 5) Enter home, 6) Leave home, 7) Personal Hygiene, 8) Relax, 9) Sleep, 10) Take Medicine, 11) Work and 12) Other Activity.

Table 1

Characteristics of the three datasets used for this study. It presents the number of samples/sensor events associated with each activity for each testbed. Each testbed had 32 sensors deployed in the environment

Dataset	B1	B2	B3
Bathing	7198	16177	5113
Bed to toilet	4168	14418	2806
Cook	101899	55080	44824
Eat	28687	24403	39380
Enter home	3685	2240	877
Leave home	4304	2457	1246
Personal Hygiene	40506	16949	37054
Relax	39929	38879	8207
Sleep	33212	10428	20676
Take Medicine	5388	7156	700
Work	0	0	108645
Other	391443	382422	249212
Total # of Activity Events	658,811	572,255	518,759

Table 2

Notation and description of the different approaches experimented in this paper.

Notation	Description
Baseline	baseline approach of fixed length sliding windows
SWTW	fixed length sensor windows with time based weighting of the sensor events
SWMI	fixed length sensor windows with Mutual Information based weighting of sensor events
PWPA	two level fixed length sensor window approach that includes the probabilities of activities in the previous window and the previously occurred activity into the feature vector at the second level classification
DW	sensor window approach where the window length is determined dynamically
TW	fixed length time window approach. The window length is defined in terms of the time interval.

Table 3

The classification accuracy of only the predefined activities for varying window lengths. *Med* corresponds to the median of overall number of sensor events of all the activities.

Testbed	*Med*	10	20	30	50
B1	47	0.59	0.58	0.49	0.43
B2	42	0.45	0.47	0.40	0.43
B3	64	0.71	0.68	0.66	0.65

Pervasive Mob Comput. Author manuscript; available in PMC 2015 February 01.

Table 4

The classification accuracy of only the predefined activities for varying values of χ for the SWTW approach.

χ =	1	2^{-1}	2^{-2}	2^{-3}	2^{-6}
B1	0.52	0.55	0.56	0.62	0.57
B2	0.41	0.41	0.42	0.45	0.44
B3	0.78	0.78	0.78	0.77	0.69

Table 5

The classification accuracy of only predefined activities using the TW approach with varying time interval values.

Time Interval =	5s	10s	15s	30s	60s
B1	0.59	0.59	0.59	0.60	0.58
B2	0.50	0.51	0.52	0.55	0.54
B3	0.76	0.76	0.75	0.74	0.74

Pervasive Mob Comput. Author manuscript; available in PMC 2015 February 01.

Table 6

Classification accuracy for the known set of activities obtained using the different approaches. The value in the parenthesis refers to the average F Score obtained across all activities.

Dataset	B1	B2	B3
Baseline	0.58(0.51)	0.48(0.49)	0.67(0.40)
SWTW	0.62(0.55)	0.45(0.55)	0.78(0.48)
TW	0.59(0.53)	0.52(0.54)	0.75(0.50)
SWMI	0.64(0.60)	0.54(0.57)	0.75(0.51)
SWTW+SWMI	0.61(0.54)	0.44(0.52)	0.78(0.47)
PWPA	0.62(0.55)	0.50(0.51)	0.72(0.45)
SWMI+PWPA	0.65(0.61)	0.56(0.61)	0.75(0.48)
DW	0.59(0.58)	0.55(0.58)	0.72(0.56)