Holmes: A Comprehensive Anomaly Detection System for Daily In-home Activities

Enamul Hoque
University of Virginia
eh6p@virginia.edu

Robert F. Dickerson
University of Virginia
rfd7a@virginia.edu

Sarah M. Preum
University of Virginia
preum@virginia.edu

Mark Hanson
BeClose
MHanson@beclose.com

Adam Barth
BeClose
ABarth@beclose.com

John A. Stankovic
University of Virginia
stankovic@cs.virginia.edu

ABSTRACT

Advances in wireless sensor networks have enabled the monitoring of daily activities of elderly people. The goal of these monitoring applications is to learn normal behavior in terms of daily activities and look for any deviation, i.e., anomalies, so that alerts can be sent to relatives or caregivers. However, human behavior is very complex, and many existing anomaly detection systems are too simplistic which cause many false alarms, resulting in unreliable systems. We present *Holmes*, a comprehensive anomaly detection system for daily in-home activities. *Holmes* accurately learns a resident's normal behavior by considering variability in daily activities based not only on a per day basis, but also considering specific days of the week, different time periods such as per week and per month, and collective, temporal, and correlation based features. This approach of learning complicated normal behaviors reduces false alarms. Also, based on resident and expert feedback, *Holmes* learns semantic rules that explain specific variations of activities in specific scenarios to further reduce false alarms. We evaluate *Holmes* using data collected from our own deployed system, public data sets, and data collected by a senior safety system provider company from an elderly resident's home. Our evaluation shows that compared to state of the art systems, *Holmes* reduces false positives and false negatives by at least 46% and 27%, respectively.

Keywords

Human Behavior Modeling, Anomaly Detection

1. INTRODUCTION

Due to increasing numbers of elderly people living alone, widespread ubiquitous deployment of sensors has become prominent for monitoring in-home activities of daily living (e.g., eating, sleeping, exercising, and entertainment). Research in accurate detection and summarization of these daily activities has progressed significantly over the last decade [19, 34, 23, 35, 9] which enables long-term monitoring of a resident's in-home activities, learning normal behavior, and detecting deviation from normal behavior i.e., anomalies. Reliable anomaly detection in daily in-home activities is the most important component of many home health care applications such as assessing behavioral rhythms [32, 10], and monitoring cognitive decline [14, 24]. Existing research has emphasized minimizing false positives and false negatives for reliably detecting anomalous events in assisted living systems [33, 20, 27, 7]. Moreover, reducing false positives in assisted living facilities is also desirable as it raises anxiety levels of patients and their relatives and often causes real alarms to be ignored [22, 12]. Some studies have found false positives as the major cause of client and caregiver dissatisfaction [12].

An anomaly detection system needs to model a resident's regular behavior accurately, and define types of deviation from the model that would be considered anomalies. Regular behavior may depend on day of the week, season, and other events of a resident's life (e.g., special occasions, visitors, medications). Anomalies can be classified as point, collective and contextual anomalies [6]. In the domain of in-home activities, point anomalies consider each instance of each activity independently and decide whether that instance is normal or not. Collective anomalies consider groups of activity instances together to determine whether the group is normal or not. Contextual anomalies consider activities under some context (e.g., day of week, under medication). Since human behavior is complex, an anomaly detection system for in-home activities should be designed to address these multiple types of anomalies. Otherwise, an anomaly detection system would become unreliable due to excessive false positives and false negatives. Existing anomaly detection systems [13, 26, 25, 24, 18, 17] either consider each activity instance as an independent data point (point anomalies) or consider sequence of activities together (collective anomalies). However, none of the existing algorithms combines context with point and collective anomalies.

Regular behavior may consist of different features for different in-home activities. For some activities, features generated from individual activity instances may be important. However, for some other activities, we may need to combine multiple activity instances from a day / week / some other period to construct useful features that represent regularity. However, existing anomaly detection systems do not address this issue; they use features from individual activity

instances. Moreover, anomaly in an activity may be caused by a previous activity on which the later activity is temporally dependent. In such cases, reporting anomalies for the later activity would generate false alarms. Also, a resident's behavior may also be affected by different explainable scenarios (e.g., new medication, special occasions, visitors). We argue that there must be a mechanism in an anomaly detection system that can learn such explainable scenarios to reduce false alarms.

We present *Holmes*, a comprehensive anomaly detection system for daily in-home activities. *Holmes* learns a resident's normal behavior in terms of in-home activities from training data. Note that based on the training data, each activity may have multiple models as part of normal behavior (e.g., weekdays, weekends, Fridays). *Holmes* automatically learns these multiple models for each activity using a novel context-aware hierarchical clustering algorithm. *Holmes* also learns temporal relationships (e.g., if two activities often happen concurrently, if one activity often happens before another) among multiple activities using sequential pattern mining and itemset mining algorithms. A highly accurate normal behavior assessment system forms the critical base upon which to detect anomalies.

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

The main contributions of *Holmes* are:

- A fully implemented system composed of a comprehensive and multi-model solution for modeling regular resident behaviors, carefully integrated with a sophisticated multi-level, semantic anomaly detection system.
- The novelty in modeling regular behavior includes the collection of following techniques: (i) hierarchically merging clusters from different days of the week ensuring that the merged clusters do not become too generalized compared to the original ones, (ii) preservation of noise points found in training for future use during re-training, (iii) the use of a combination of features extracted from both individual activity instances and group of activity instances combined over a specific period, and (iv) use of sequential pattern mining and itemset mining algorithms to learn groups and sequences of activities that are part of a resident's regular behavior (i.e., temporal correlations among activities).
- The novelty in the anomaly detection arises due to the combination of the following: combining point, collective and context anomalies to ensure reliability, use of semantic rules that represent logical deviation from regular behavior to reduce false alarms, and learning new semantic rules based on resident / expert feedback.
- Evaluation of *Holmes* using 1) six months of activity data collected from one single-resident home; 2) two publicly available data sets each of which has six months of user-labeled in-home activity data; and 3) one data set from *BeClose* (a commercial senior safety system provider [2]) that contains four months of activity data in one single-resident home. Our evaluation shows that *Holmes* reduces false positives by at least 46% and false negatives by at least 27% compared to three state of the art anomaly detection systems. Also, the use of semantic rules reduces false positives by an additional 20%.

2. RELATED WORK

Many current solutions use training data as a baseline and use statistical or clustering based anomaly detection approaches to find point anomalies. Such as, Virone et al. [32] monitored 22 patients in an assisted living facility for two weeks that was treated as a baseline behavior. Then the baseline was used for the next six months to find behavioral changes in their circadian rhythms. For changes of one order of magnitude a warning was signaled, and for changes of two orders of magnitude an alarm was signaled to a caregiver. In [28, 24, 29], authors learn which rooms the resident is in during different times of day and monitor anomalies in room occupancy. However, circadian rhythms and room locations are very limited features to represent behavior.

Han et al. [13] use the mean of different features for different activities to define regular behavior and look for point anomalies based on predefined thresholds of deviation. Clustering based techniques are used in [26] to detect anomalies in timings and durations of different activities. All the above anomaly detection systems often suffer from generating numerous false positives that makes them unreliable. One reason of generating higher number of false positives is treating each activity instance independently ignoring the correlation among them.

There are existing systems that consider correlations among activities to detect collective anomalies. Anderson et al. [5] take an automata based approach to define sequence of activities as behaviors and learn those behaviors. They also support combining multiple days of activities to detect anomalies that occur over the time. However, they do not consider durations of each of the activities or the intervals among activities. These features are very useful for many health care applications. Jakkula et al. [18] use temporal mining to learn different temporal relations among different activities. In [17], authors use support vector machines to detect anomalies in sequences of activities. Authors use unsupervised pattern clustering techniques [31] to identify behavior model of the resident. However, none of these collective anomaly detection techniques considers differences in daily routines in different days of the week and specific features related to individual or group of activity instances which may cause both false positives and false negatives.

A survey of anomaly detection techniques in various application domains is presented in [6] where the authors explain how context plays an important role in detecting anomaly. None of existing anomaly detection systems for in-home activities addresses the effect of context (e.g., day of the week, weather) on daily activities. *Holmes* is novel compared to existing systems in this respect. Another shortcoming of the above techniques is the lack of semantic rules to filter out false positives as logical deviation from normal behavior.

3. SYSTEM DESCRIPTION

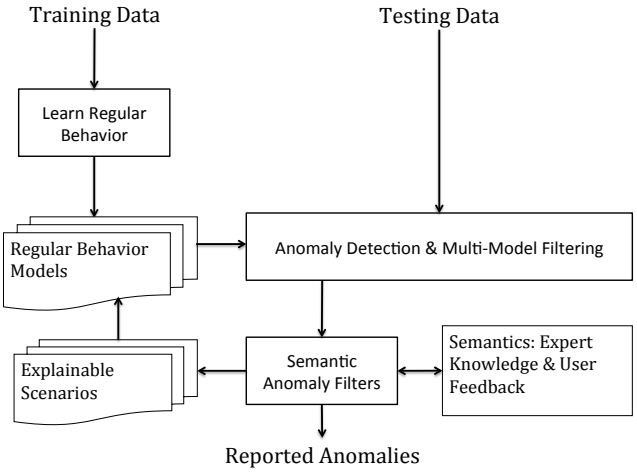


Figure 1: *Holmes* Framework for Anomaly Detection

3.1 Holmes Framework

The first step of any anomaly detection system is to learn the normal behavior based on training data. Normal behavior consists of what in-home activities a resident performs, which time of day each activity takes place, how long an activity lasts, and relationships among different activities. For some activities, features generated from individual activity instances may be important. However, for some other activities, we may need to combine multiple activity instances from a day / week / some other period to construct useful features that represent regularity. Moreover, anomaly in an activity may be caused by a previous activity on which the current activity is temporally dependent. In such cases, reporting anomalies for the later activity would generate false alarms. Therefore, a holistic approach is necessary to consider the different kinds of anomalies (e.g., point, collective, contextual anomalies) and filter false alarms that may be caused by other anomalies. Once normal behavior is modeled and labeled, an anomaly detection system can detect any deviation from the normal behavior that is represented by and learned from the training data. Different applications may need to monitor a different subset of anomalies that *Holmes* can detect. It is up to the domain experts to decide what anomalies they want to detect for their applications.

Figure 1 shows the *Holmes* framework for anomaly detection. The training and testing data consist of sets of activity instances. An activity instance is represented by an activity identifier, the time when the activity starts, and the total duration of that activity; we use the notation ($activityID$, $startTime$, $duration$) for this representation. The start time and duration of each activity instance may be logged by the user, or may be detected by an activity recognition system based on in-home sensors. The function of the 'Learn Regular Behavior' component of *Holmes* (Section 3.2) is learning a resident's regular behavior from the training data. Since human behaviors are complex, this results in multiple regular behavior models. Based on these models, the 'Anomaly Detection and Multi-Model Filtering' module (Section 3.3) detects anomalies in the activity instances of the testing data first. Next, this module filters out the detected anomalies that do not satisfy the anomaly conditions in all the corresponding models learned during the 'Learn Regular Behavior' module. Only the anomalies that pass the filtering are forwarded to the 'Semantic Anomaly Filters' module.

The 'Semantic Anomaly Filters' module, described in Section 3.4, filters out those anomalies that may be explained by 'Expert Knowledge & User Feedback', i.e., semantics. For example, the user may be recovering from a major operation, or there may be a power outage which caused loss of sensor data. This module detects such scenarios and does not report them as behavioral anomalies. Such scenarios are saved in 'Explainable Scenarios'. If there is significant number of instances of a particular scenario, a new behavioral model is developed for this scenario and stored in the 'Regular Behavior Models' so that such scenarios are not detected as anomalies in future. Thus, the set of regular behavior models is continuously enriched.

3.2 Learning Regular Behavior

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

3.2.1 Modeling Each Individual Activity

One part of learning complex behavior is to model regularities in each individual activity. For example, when does the resident usually go to bed, duration of sleep, how many times a week the resident goes to the gym, whether the resident usually has lunch at home: all this information may be indicative of regular behavior and deviations from that may indicate change of lifestyle or may be due to some illness. We propose a novel context-aware hierarchical clustering algorithm to model each individual activity. The context we consider here is 'day of the week'. Our hypothesis is that people may have different routines for different days of the week. We model each day's behavior (e.g., Sunday, Monday) separately and then merge models for multiple days (e.g, merge Friday and Saturday together) if they are very similar. If adequate data is available, the algorithm can also handle other context and model behavior accordingly e.g., for different seasons, for 4th of July, for Thanksgiving.

[**Feature Selection**] For modeling each activity instance, we cluster the activities based on the $startTime$ and $duration$ of the activities. If any additional feature is available for an activity, then that can also be incorporated in the feature set. For example, sleep can have an additional feature named *sleepQuality*, breakfast can have an additional feature *calories*. We do not have these features in our data sets. For each activity, we also combine all activity instances in a day and calculate the following two features: 1) $totalCountPerDay$ represents how many times an activity takes place in a day 2) $totalDurationPerDay$ represents the total duration of an activity in a day (the sum of durations of all activity instances). Note that we also consider the days when an activity does not happen (i.e., $totalCountPerDay$ is zero). As a result, if an activity does not take place often in a specific day, then that is also represented by a cluster.

[**Context-aware Hierarchical Clustering Algorithm**] Figure 2 explains our context-aware clustering algorithm that works in a bottom-up hierarchical way. The novelties include (i) merging clusters from different days of the week hierarchically ensuring that the merged clusters are not too generalized compared to the original ones, (ii) preservation

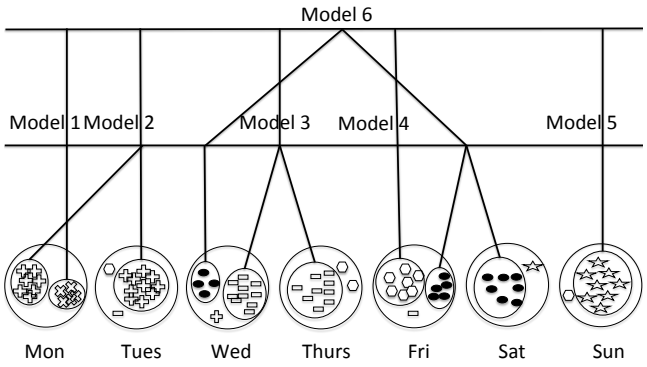


Figure 2: Per-activity Context-aware Hierarchical Clustering

of the noise points for future use during retraining, and (iii) use of combination of features extracted from both individual activity instances and group of activity instances combined over a specific period. Note that the algorithm runs separately for each activity.

At the bottom layer, for each day of the week, the algorithm combines different instances of an activity and clusters them based on their features. When we model the activity instances independently, the features are *startTime* and *duration*. When we model the collection of all activity instances of a day, the features are *totalCountPerDay* and *totalDurationPerDay*. For clustering in the bottom layer, we use the DBSCAN clustering algorithm [11] which is a density based clustering algorithm. The major advantage of DBSCAN is that we do not need to specify how many clusters there are. This is important, because we do not know how many different activities there are. DBSCAN has two parameters; one is *min_pts* which is the minimum number of points in a cluster, and the other is *Eps* which is the minimum distance between two data points for them to be considered in the same cluster. We only specify the parameter *min_pts* and use the corresponding value of *Eps* that DBSCAN suggests. In Section 5.1 we discuss the effect of varying *min_pts* on clustering performance. After clustering, DBSCAN marks each point as belonging to a cluster or as noise.

Note that there may be some activity instances that are not part of any cluster. During training, we do not consider them as part of any model i.e., regular behavior. However, we do not totally discard these activity instances. After training, when *Holmes* is used for anomaly detection, if an activity instance does not fall into any regular model, we try to combine them with similar other activity instances that were left alone during training.

From the bottom layer of Figure 2, we get one or more clusters for each day of the week. From here on, the algorithm progresses upwards like agglomerative clustering [1] i.e., it merges two clusters from the bottom layer which are closest to each other and advances the merged cluster to the upper layer. The important part of this merging step is calculating the distance between two clusters i.e., the linkage. For each cluster, we define the diameter of a cluster as the maximum distance between any two points. We merge two clusters with diameter d_1 and d_2 only if after merging the diameter of the new cluster d_{new} satisfies the relations in (1) and (2). Note that if only one of the conditions is satisfied, then we do not merge these two clusters. Merging in such cases would generalize the cluster (for which the condition

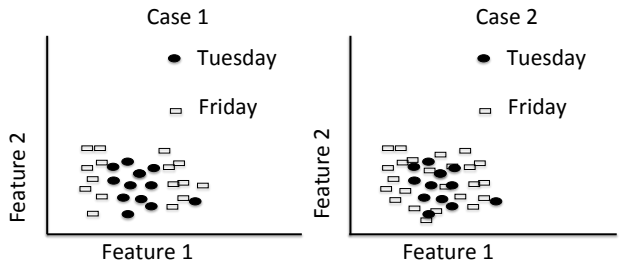


Figure 3: Sample scatter plot of instances of a specific activity for 2 different cases showing need of context-aware clustering

is not satisfied) too much. In each layer of Figure 2, a cluster can be merged with multiple clusters under the merging condition, *Holmes* selects the one with minimum d_{new}. If a cluster cannot be merged with another one, it moves to the upper layer. The merged clusters move to the upper layer together as one cluster. If at any layer no new merging is possible, the algorithm stops.

$$\frac{d_1}{d_{new}} > MERGE\_THRESHOLD \tag{1}$$

$$\frac{d_2}{d_{new}} > MERGE\_THRESHOLD \tag{2}$$

We argue that our context-aware (day of the week being the context here) clustering algorithm is necessary to model regular behavior, because without it we may not capture if the resident behaves in a specific way on specific days of the week. For example, consider the sample scatter plots of two different cases shown in Figure 3. If we cluster the activity instances without considering the day of the week, then all the activity instances of Tuesday and Friday would be clustered together in one cluster in both cases. We may separate the cluster of Tuesday in Case 1, if we specifically mention that we want three clusters. However, it is difficult to know how many different clusters there may be apriori. However, even then it won't be possible to separate instances of Tuesday in Case 2. This is why clustering activity instances from different days separately is necessary, and we go one step further by also merging days that have similar behavior.

Our algorithm creates a separate cluster for Tuesday, and two different clusters for Friday for Case 1. However, for Case 2, our algorithm creates one cluster for Tuesday and one cluster for Friday. Another point to note from Figure 3 is that there is one activity instance from Tuesday (in both cases, on the right side) which is abnormal. If we cluster all instances together, we are not able to identify it. One alternate to our solution is just to use different models (i.e., clusters) for different days of the week. We argue that such an approach would fail to capture several high level regularities in a user's behavior, such as, in which days of the week the behavior is similar.

Note that there may be activities that do not take place daily. If any of these activities happen only on specific days of the week, then it would be included in that days model. However, if any of these activities do not have any correlation with specific days of week, then there may not be enough instances of that activity in any specific day for the clustering algorithm to run. To address this, we cluster all instances of any such activity together based on *startTime* and *duration* using DBSCAN and get one or more models.

Also, for each activity instance of such activities, we calculate a new feature named *interval* which is defined as the interval of this activity instance from the last time that activity took place in terms of days. We also test if the *interval* of the activity instances follow any particular model.

3.2.2 Modeling Activities Together

Modeling activities indpendently may not represent regular behavior completely. We also need to model the temporal correlations among activities. The temporal correlation among activities can be sequential or non-sequential. For example, two or more activities may frequently happen concurrently or in a specific sequence maintaining a similar time interval, e.g., cooking dinner at Friday night is followed by watching TV. We need to model such sequential temporal correlations. Alternately, two activities can often take place in a segment of the same day of the week, but in no specific order, e.g., the duration of resident's dinner is long only during the days when he/she goes to the gym. In this section we describe the detail of modeling sequential and non-sequential temporal correlations.

Prior to finding temporal correlations among activities, we pre-process the activity instances. Note that activity instances are represented by tuples of the form (activityID, startTime, duration). From the clustering step of the previous section each activity instance belongs to a unique model (cluster), and some instances do not belong to any cluster (considered not regular). We assign the instances not belonging to any cluster a default model number. In this pre-processing step, for each activity instance, we create two new tuples: (acID_ModNo_Start, startTime) and (acID _ModNo_End, endTime).

We find the frequent sequences of activities and the normal time intervals between successive activities. We apply a sequential pattern mining algorithm [4] to learn these temporal correlations as frequent patterns. For example, for a particular day of the week, we may find a frequent pattern (ac1_Mod11_Start, ac2_Mod21_Start, ac2_Mod21_End, ac1_Mod11_End) which represents that on that day, $activity2$ frequently starts after $activity1$ starts, and ends before $activity1$ ends. We also know start and end times of all activity instances, so we can model the duration of each activity and intervals between successive events as normal distributions.

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

For each day of the week D_i, there is a set of segments $\{s_{ij}\}$, where each segment s_{ij} is a sequence of tuples of the form (acID_ModNo_Start, startTime) or (acID_ModNo_End, endTime); each such tuple is represented by (a_{ijk}, t_{ijk}). We consider this sequence $<(a_{ijk}, t_{ijk})>$ corresponding to each segment s_{ij} as one stream st_{ij}. We apply sequential pattern mining [4] on the set of streams corresponding to each day of week to get the set of frequent patterns for that day.

We use a state of the art sequential pattern mining algorithm PrefixSpan [30]. For each day of week D_i, we run PrefixSpan separately with $\{st_{ij}\}$ as input. For each member sequence $st_{ij} = <(a_{ijk}, t_{ijk})>$, the algorithm only considers the activity identifiers ($<a_{ijk}>$) and ignores the timestamps. As output, we get the set of frequent patterns $\{FP_{il}\}$, where each FP_{il} is a sequence of activity identifiers of the form $<a_{ilm}>$. A pattern is considered to be frequent if the number of different instances of the day of week D_i when the pattern occurs is more than a threshold $FP\_THRESHOLD$. For each day of week, thus we get a set of frequent patterns each of which represents a pattern of activities. For each frequent pattern, we model the duration of each individual activity (that is part of the pattern) and the interval between successive activities separately as normal distributions. This enables us to identify cases where anomaly in start time or duration in one activity may cause irregularities in later activities in a pattern.

[Modeling Out-of-sequence Temporal Correlations] Although the sequential pattern mining algorithm above enables us to understand sequential temporal correlations among activities it would ignore the group of activities that only have out-of-sequence temporal correlations. Hence, we also apply an itemset mining algorithm on the segments of each day of the week to find the non sequential temporal correlations. In our case, an itemset represents a group of activities, and the goal of the itemset mining algorithm is to find the frequent itemsets i.e., those where the group of activities often happen together within the same segment.

Similar to the sequential pattern mining algorithm, the itemset mining algorithm is applied separately on the segments of each day of the week. We apply a state of the art itemset mining algorithm named apriori algorithm [3]. As output, we get the set of frequent itemsets $\{FI_k\}$, where each FI_k is a set of activity identifiers. An itemset is considered frequent for a specific day of the week, if it's set of activities happen together more than a threshold ($ITEM\_THRESHOLD$) number of different instances.

3.3 Anomaly Detection and Multi-Model Filtering

After training, for each day of the week, each activity has one or more models (i.e., clusters). For example, in our evaluation, the number of clusters generated for all activities in different datasets range between 15 - 35 (details in Section 4). Note that some clusters are calculated based on $startTime$ and $duration$ of the activity instances for that day of the week; each activity instance (except the irregular ones) belongs to one of these clusters. And the remaining clusters are calculated based on $totalCountPerDay$ and $totalDurationPerDay$. For the later type of clusters, if an activity does not often take place in a specific day of the week, then a cluster is formed representing this so that during testing similar irregularity is not reported as an anomaly.

An anomaly score can be represented based on discrete values (e.g., 'normal', 'abnormal', 'very abnormal') or based on continuous values where a higher value would represent a more anomalous event (or the opposite). We choose a discrete representation because it is easier to interpret for the experts / caregivers. After training, for each cluster, we calculate the mean μ_i and standard deviation σ_i of each feature i based on the data points belonging to that clus-

ter. During testing, for a new data point x represented by $(x_1, x_2, ..., x_k)$, we calculate the *Mahalanobis Distance* d from a cluster according to Equation (3). If the instances in a cluster are normally distributed along the k dimensions (i.e., k features), then $d \leq \sqrt{k}$ means x is within one standard deviation from the cluster center. If x is not within two standard deviations (i.e., $d > 2 * \sqrt{k}$) from the cluster center, we consider that x does not belong to this cluster. *Mahalanobis Distance* is widely used for cluster analysis, as it normalizes variation in each feature value by its standard deviation in the training data [16].

$$d = \sqrt{\sum_{i=1}^{k} \frac{(x_i - \mu_i)^2}{\sigma_i^2}} \quad (3)$$

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

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

We also report anomalies in the set of frequent patterns and frequent itemsets for each day of the week. In this way, *Holmes* detects collective anomalies of type 'Activity Instances from Multiple Activities'. For each frequent pattern (i.e., sequence of activities), if we find one or more activities are out of sequence, we report an anomaly in that pattern to the 'Semantic Anomaly Filter' layer. The time intervals between successive activities in a pattern is modeled as a normal distribution, and if any interval in testing data falls two standard deviation outside the mean, we also report it as an anomaly. Similarly, for each frequent itemset (i.e., group of activities), if only a subset of them takes place, we report an anomaly in that itemset.

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

If *Holmes* is used to monitor symptoms of a particular disease (e.g., depression, diabetes, dementia), then an expert can select a group of activities and specific sets of frequent patterns and itemsets so that *Holmes* monitors only those anomalies. Also, an expert can assign different weights to anomalies in different activities or patterns. Note that anomaly scores are calculated on a daily basis.

3.4 Semantic Anomaly Filters

Here, the goal is to detect anomalies that may be normal under explainable scenarios. Each of these explainable scenarios is represented by a semantic rule. If *Holmes* is informed that one of the explainable scenarios is currently true, then it checks if any anomaly is related to that scenario so that the corresponding anomaly can be suppressed.

The 'Explainable Scenarios' may include: entertaining visitors, power outage, recovering from major medical operation, and extreme weather (cold/heat wave). This list is extensible over time. One of the novelties of our system is this set of semantic rules that will help to reduce false alarms in anomaly detection. For example, if our system detects that the user is out of house for more than a day, it does not look for anomalies in other activities, e.g., eating, sleeping. It only reports that user is not in house and thus reduces false positives in other activities. As another example, if it is now known that a person is suffering from prostate problems, then frequent toilet visits are highly likely and thus should not be reported as anomalies. Another type of semantic filter is defined by experts in the form "Do not report anomaly in an activity if there are less than X anomalies in Y days". *Holmes* does not claim or aim to detect these explainable scenarios automatically. However, if *Holmes* is notified about the presence of a specific scenario, it learns to look for specific anomalies that may occur during that scenario and discards them from the set of anomalies.

4. EXPERIMENTAL SETUP

4.1 Our System with Real Deployment

We have collected activity data from a single resident home for six months for a comprehensive evaluation of *Holmes*. We use the system described in [15] for obtaining the ground truth pertaining to activities occurring in the home by placing microphones in the rooms and using grammar based speech recognition for processing the speech. We deployed this system in a single resident home and collected data for six months. The resident is a graduate student. The activity labels collected during this deployment include sleeping, meal preparation, cooking, breakfast, lunch, dinner, snack, dish washing, watching TV, using laptop, going to toilet, taking showers, and leaving and entering the home. The

resident also labels all activity instances as regular or irregular. For example, if the resident most often has dinner at home, then occasional dinner out of home is marked as irregular.

4.2 Public Data Sets

We use two public data sets from the CASAS smart home data set [8]. Each data set contains four months of in-home activity data labeled by the resident. The set of activities includes sleeping, meal preparation, working, watching television, showering, leaving and entering home, and using the toilet. The data sets do not provide information on whether any of the activity instances is anomalous.

4.3 *BeClose* Commercial System

BeClose is a commercially available home monitoring technology that promotes safety and wellness for aged and disabled individuals [2]. The *BeClose* technology platform consists of wireless, battery operated ambient sensors such as passive infrared motion sensors, door and cabinet sensors, bed and chair occupancy sensors, and floor presence mats. The sensors communicate with a base station connected via a cellular link to a remote monitoring data center in the cloud. Sensor data is then contextualized to human behavior and to activities via hierarchical clustering and heuristics based approaches. Activities include sleep, sedentary behavior, entries and exits, ambulatory activity, and kitchen use. For this study the *BeClose* system was deployed in the home of one individual over the age of 70 and data was collected for four months.

4.4 Baseline Approaches

We compare the performance of *Holmes* with the following three standard techniques of anomaly detection that are widely used in the domain of behavior analysis and smart health care [32, 18].

Statistical Approach

We implement a statistical approach similar to [32]. For each activity and for each hour of the day, we calculate the mean (μ) and standard deviation (σ) of how long that activity takes place in that hour in the training data. During testing, if the duration of an activity in a specific hour of the day is outside $\mu \pm 2\sigma$, then we consider it as an anomaly. Each activity instance can generate at most one anomaly. From that probability, we get an anomaly score and calculate the mean (μ) and standard deviation (σ) of anomaly scores over all activity instances of an activity.

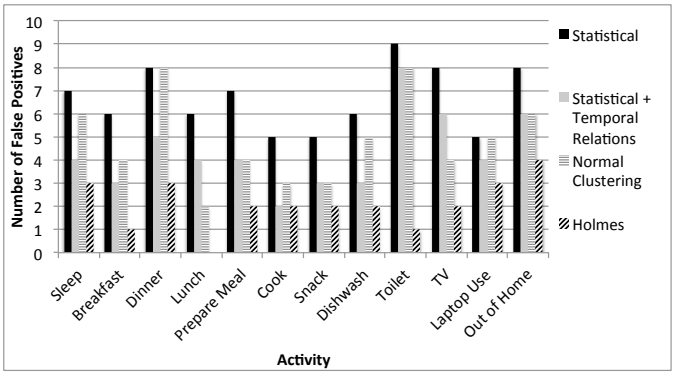
Statistical Approach with Temporal Correlations

Here, in addition to statistical measures, we implement the temporal relationships among activities as in [18] which models an activity based on what other activities happen before / after / during that activity. From the training data, a probability is calculated for each activity based on what activities happen before / after / during that activity. From that probability, we get an anomaly score and calculate the mean (μ) and standard deviation (σ) of anomaly scores over all activity instances of an activity. During testing, if an activity instance's anomaly score falls outside $\mu \pm 2\sigma$, we consider it an anomaly.

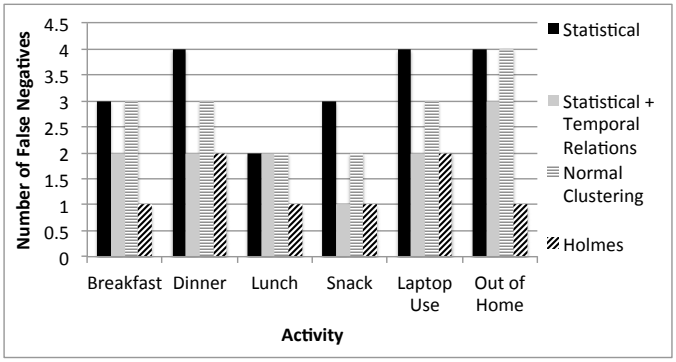
Clustering Based Approach

In this case, we cluster activity instances of each activity based on $startTime$ and $duration$ without considering the day of the week as in [26]. A new activity instance is classified as anomalous or normal based on same technique as *Holmes*. We do not consider any temporal correlation in this baseline approach.

5. EXPERIMENTAL RESULTS



(a) Avg. number of false positives per month (rounded) generated by different algorithms using cross-validation. *Holmes* generates the least number of false positives for every activity and reduces number of false positives by at least 46%.



(b) Avg. number of false negatives per month (rounded) generated by different algorithms using cross-validation. For some activities, none of the algorithms generate any false negative (e.g., sleep). Hence they are not shown here. *Holmes* generates the least number of false negatives for each activity and reduces number of false negatives by at least 27%.

Figure 4

5.1 Evaluation on Six-month Deployment

Table 1 shows a list of clusters for all activities after training with the first three months' data. Although many common activities can be learned in as little as 2 or 3 weeks of data, we perform our evaluation with 3 months of training data since we are also interested in weekly, bi-weekly and even monthly behaviors. In the subsection labelled as *effect of training size* we also show the effect on accuracy of different training periods. We describe the clusters for each activity non-technically in the table for ease of understanding. For some activities (e.g., sleeping, breakfast), the clusters consist of only per instance features (i.e., $startTime$ and $duration$); this means that there is no variation in the

collective features (i.e., *totalCount* and *totalDuration*). For some activities, both types of features are used. For example, the resident's regular behavior includes having dinner occasionally out of home on Friday / Saturday / Sunday; therefore, 'No Dinner' is included as a model for those days along with models based on timing and duration of dinner when in home. Some activities do not happen daily, they only happen every few days (e.g., cooking, shower). For such activities, the interval between successive occurrences is used in the models. Table 1 also shows us that some activities (e.g., lunch) take place in the home only on specific days (e.g., weekends).

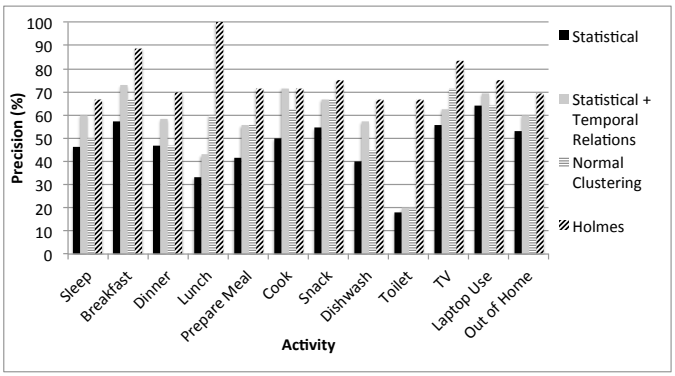
Holmes also extracts 21 frequent patterns in daily activities after training with the first three months of data. Table 2 lists some of the patterns. Each of the patterns contains temporal relationships among the activities that belong to that pattern. Patterns no. 1 and 2 show us that on weekends, the resident often has breakfast while watching TV. However, for weekdays, it is not the case. The other patterns in the table show us that while having lunch, dinner or snack, the resident often watches TV. 'dishwash2' in pattern no. 10 represents instances of dish washing that have longer duration and the pattern shows us that on days when the resident cooks, the duration of dish washing is longer.

We compare the performance of *Holmes* on this data set with the three baseline approaches described in Section 4.4. We train each algorithm on the training data, and test for anomalies on the testing data. We use a 20 fold cross validation, where in each fold we use three months data for training, and the remaining three months for testing (e.g., data from month 1, 2, 3 for training and data from month 4, 5, 6 for testing). All the activity instances are hand labeled by the resident as regular / anomaly. Based on that ground truth, we calculate the average (rounded) number of false positives and false negatives in anomaly detection over all folds by each algorithm. Figure 4a shows comparison of number of false positives of all the algorithms. The figure only shows the activities for which at least one of the algorithms have false positives. Here, we see that *Holmes* has the least number of false positives for all activities. For example, *Holmes* has no false positive for lunch, and for toileting, the standard deviation approach has nine false positives while *Holmes* has only one.

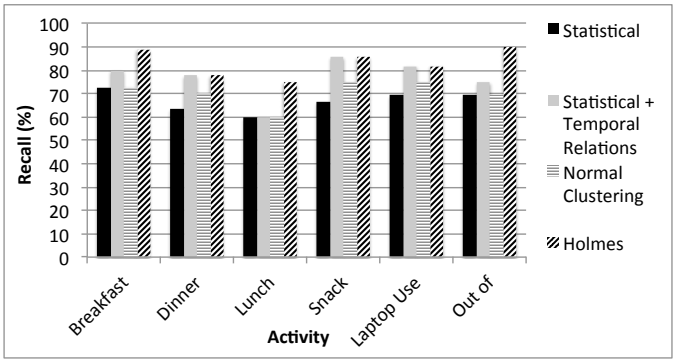
The statistical approach, used by many existing systems, performs the worst; on average, it generates almost one false positive per day. This is mainly because it does not consider the temporal relations among activities. For example, if the resident returns home later than normal one night, then all the subsequent activities are delayed. However, the only anomaly was his late arrival. The statistical approach would report anomalies in all subsequent activities. When we add the temporal relations to the statistical approach, the number of false positives decreases. *Holmes* performs even better because it learns different temporal relations in more detail by considering the sequence of activities rather than just considering temporal relations among pair of activities. The normal clustering based approach performs better than the statistical approach because of modeling the temporal regularity better. However, it does not consider temporal relations; therefore, it has many false positives. the number of false positives in 'toilet use' is high in other algorithms because the collective features are more significant in terms of representing normal behavior than exact timing of each toilet visit. On average (over all activities), *Holmes* reduces the number of false positives by 67%, 46%, and 56% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively.

Figure 4b shows comparison of number of false negatives of all algorithms; it only shows the activities for which at least one of the algorithms has false negatives. From this figure, we see that *Holmes* has the least number of false negatives for all activities. The main reason is that *Holmes* captures the variability in resident's behavior on different days of the week. There are few instances for each of these activities in Figure 4b, when the resident behaves unusually compared to his routine for a particular day of the week. *Holmes* correctly identifies such anomalies. On average (over all activities), *Holmes* reduces number of false negatives by 59%, 27%, and 51% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively.

Precision and Recall



(a) Average precision values for different algorithms using cross-validation for different activities. *Holmes* has the maximum precision for each activity and increases precision by at least 17%.



(b) Average recall values for different algorithms using cross-validation for different activities. *Holmes* has the maximum recall for each activity and increases recall by at least 6%. As some of the activities have no false negative in any algorithm, they result into 100% recall. They are not shown here.

Figure 5

Figures 5a and 5b show the corresponding average precision and recall values for different algorithms, respectively. *Holmes* has the highest precision and recall values for each activity. The relatively low precision values for some activities (even for *Holmes*) indicate that for those activities (e.g., 'Sleep', 'Toilet', 'Dishwash') the number of true anomalies

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

Table 1: List of models based on 3 months of training data for all activities.

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

Table 2: A subset of frequent patterns generated from 3 months of training. If there is no other activity between start and end of an activity, it is represented by its name.

is very low. On average, *Holmes* increases precision by at least 17%, and recall by at least 6%. However there is a trade off between precision and recall as with the increase of recall, false positives will increase and thus the precision will decrease. We can achieve the required balance between precision and recall for a specific application by tuning the values of different thresholds.

Effect of Semantic Rules

Based on the resident's feedback, we find the following semantic rules for this deployment: 1) The presence of guests causes specific variation in cooking, watching TV and the amount of time outside home; 2) The ordering of food for home delivery causes specific variations in preparing meal and dish washing; 3) Going to or coming back from a trip cause specific changes in entry or exit and sleeping; 4) Sickness causes variations in most daily activities.

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

Effect of Training Size

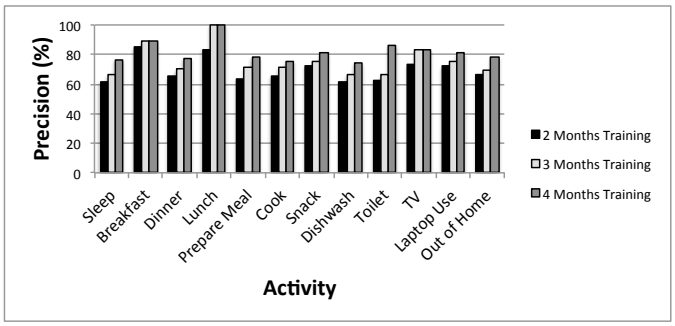
For frequent activities only 2 to 3 weeks of training are necessary. But this does not capture people's behaviors, many of which are weekly, bi-weekly or monthly. Here we show the effect on accuracy for different training periods. Figures 6a and 6b show the corresponding average cross-validated precision and recall of *Holmes* with the variation of training set size of 2, 3, and 4 months while the rest of the data is used as the test set. For all activities other than *breakfast*, an increase of 2 months in training set increase the average precision by at least 8%. For most of the activities, an increase of 2 months in training set increase the average recall by at least 13%. To minimize training time an interesting approach might be to employ a crowd-sourcing approach as found in [21].

Effect of Different Components of Holmes

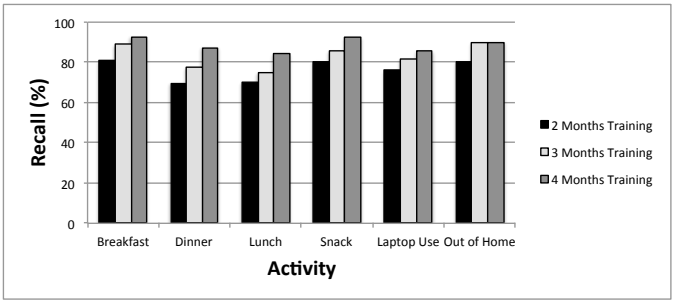
Figure 7 shows number of false alarms for different activities under three settings. In the baseline setting, we just cluster based on features of different activities without considering the context. In the second setting, we calculate the number of false alarms after applying our context aware hierarchical clustering algorithm. Finally, we apply our temporal correlation based anomaly detection in addition to the context aware clustering (as in *Holmes*). From Figure 7, we see that the context aware clustering algorithm reduces the number of false positives for sleep from 6 to 4. On average the context aware clustering algorithm reduces false alarms by 34% from normal clustering baseline. Adding the temporal correlation analysis further reduces false alarms for some activities, e.g., sleep, breakfast, dinner, TV. On average adding the temporal correlation analysis reduces false alarms by 32% from the context aware clustering algorithm.

Detecting Different Types of Anomalies

Figure 8 shows different kinds of true anomalies detected by *Holmes* in one of the cross-validation folds. For example, for the activity sleep, there is no point anomaly due to missing sleep episodes. However, three sleeping episodes are detected as point anomalies due to irregularities in temporal features. One collective anomaly is detected by combining sleep episodes of the same day when the subject took more naps than usual during that day. Finally, on two cases anomalies are detected due to irregularities in the frequent patterns containing sleep. Similarly, for each activity,



(a) Average precision values for different training size using cross-validation for different activities.



(b) Average recall values for different training size using cross-validation for different activities. For most of the activities the recall increases at least 13% when the training set size increases by two months. The activities that have 100% recall for all algorithms are not shown here.

Figure 6

Holmes detects different types of true anomalies which are summarized in Figure 8. For example, in case of preparing meal, *Holmes* finds two point anomalies due to missing activity and two point anomalies due to irregularity of schedule.

Effect of Threshold Values

There are different thresholds used in different parts of the *Holmes* framework. Based on different application domains, different values may be appropriate for these thresholds and these should be set by the experts or estimated by experiments. We experimented with different values for these thresholds and use the values that produced most accurate results in terms of reducing false positives and negatives in our data. In the following section, we discuss the effect of varying different thresholds.

Effect of Merge Threshold

Figure 9 shows the effect of $MERGE\_THRESHOLD$ on the precision and recall values. Precision values do not vary with $MERGE\_THRESHOLD$ a lot, this is because false positives are mainly reduced due to using both point and collective features, and correlations among activities. However, at higher $MERGE\_THRESHOLD$ values (more than 80%), precision values decrease due to the clusters being too specific. On the other hand, recall values generally increase with the increase of $MERGE\_THRESHOLD$. At lower values, most of the clusters for an activity are merged. Therefore, some clusters get very generalized which causes false negatives, and recall values decrease as a result. If we increase the $MERGE\_THRESHOLD$ value, such generalization of clusters does not happen. However beyond 80%, we do not see significant increase in the recall values. Therefore, we use 80% as the $MERGE\_THRESHOLD$ value.

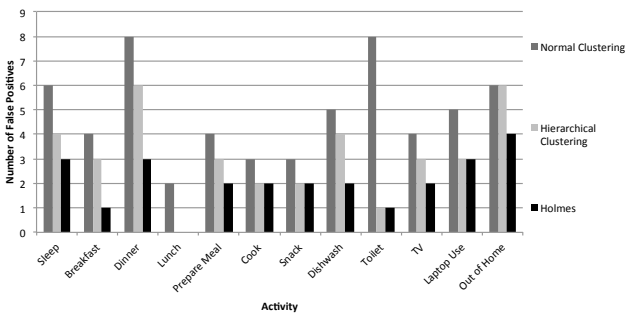


Figure 7: Effect of different components of *Holmes* in reducing false alarms

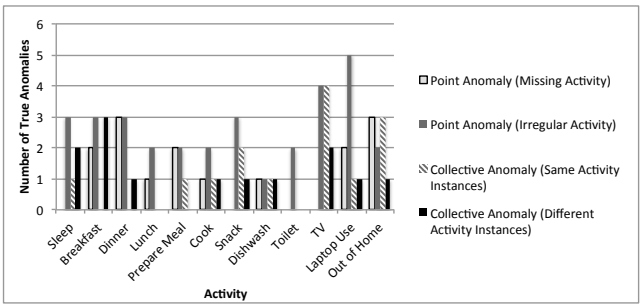


Figure 8: Different types of true anomalies detected by *Holmes* for each activity

Effect of Other Thresholds The value of *min_pts* (used in DBSCAN) is set to 4, i.e., in each cluster there has to be at least four data points (four different days). We use a relatively low value so that we do not exclude too many data points as noise. [11] also suggests that using *min_pts* more than four often produces the same sets of clusters as it produces when *min_pts* is set to four. Our experiments also confirmed this.

As $SEG\_THRESHOLD$, we use 60 minutes. If the interval between successive activities is more than an hour, we consider them in different episodes. As the value of both $FP\_THRESHOLD$ and $ITEM\_THRESHOLD$, we use 80%. We select this relatively high value to avoid too many frequent patterns and item sets. These values were selected based on similar experiments as discussed before.

5.2 Evaluation on Public Data Sets

The two public data sets do not have ground truth for anomalies. However, one of the main problems in the usability of anomaly detection systems is large numbers of false alarms. Our evaluation on the data that we collected also shows a high number of false positives. Therefore, in the absence of ground truth for anomalies, we compare the number of anomalies each system detects on the two public data sets. We conjecture that many of the detected anomalies would be false alarms that can be explained by semantic rules or some special conditions. Hence, the system generating fewer number of alarms is desirable. We do a 4-fold leave-one-out cross validation; in each fold, we use three months data for training, and the remaining months for testing. We show the average (rounded) number of anomalies generated by each system. Figures 10a and 10b show the comparison of number of anomalies in data set 1 and 2, respectively. The figures only show the activities for which at least one of the algorithms detects an anomaly.

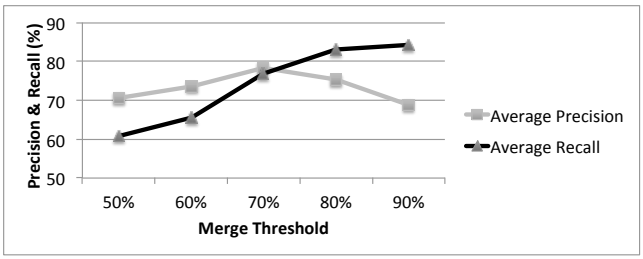
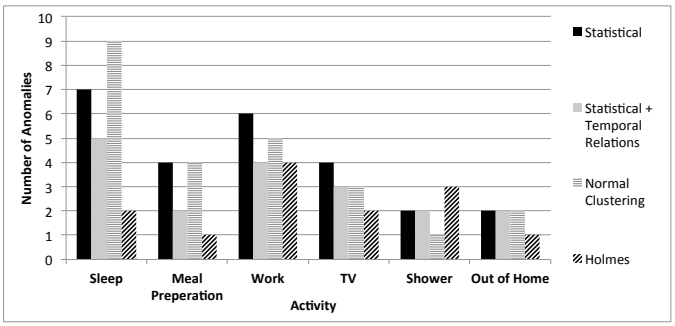
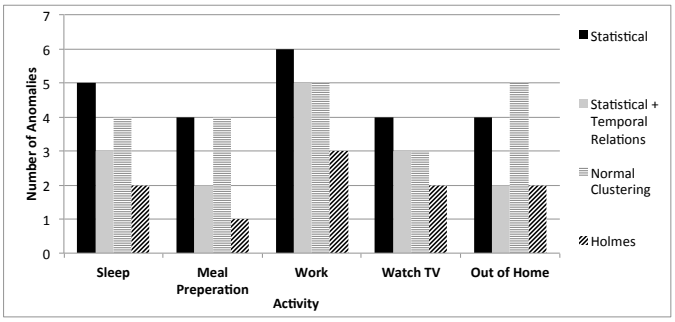


Figure 9: Effect of $MERGE\_THRESHOLD$ on precision and recall values. We used the value 80%.



(a) Average detected anomalies per month (rounded) using 4-fold cross-validation for public data set 1. *Holmes* generates the least alarms for every activity but showering.



(b) Average detected anomalies per month (rounded) using 4-fold cross-validation for public data set 2. *Holmes* generates the least number of alarms for every activity.

Figure 10

Figure 10a shows that for data set 1, *Holmes* detects more anomalies than other systems for showering which usually takes place in every two / three days. However, there are some cases where there are no instances of showering in a week. Because, other systems do not consider the interval between activity instances, they do not detect such anomalies. Either the resident did not shower in such cases or did not record shower events. On average, *Holmes* reduces the number of alarms by 41%, 26%, and 31% compared to statistical approach, statistical approach with temporal relations, and normal clustering, respectively for data set 1, and for data set 2, on average *Holmes* reduces number of alarms by 57%, 31%, and 51%, respectively. The context-aware clustering algorithm of *Holmes* helps to reduce the number of alarms, because some activities (e.g., sleeping, preparing meals and working) show different temporal characteristics on specific days of the week in both data sets. Such instances may be few in numbers if considered together with all other instances. Statistical and normal clustering approaches generate alarms as they do not isolate activity instances of such days which *Holmes* avoids.

5.3 Evaluation on *BeClose* Data

As this dataset does not have ground truth for anomalies, we compare the number of anomalies generated by different systems. Here, we do a 4-fold leave-one-out cross validation; in each fold, we use three months data for training, and the remaining month for testing. Table 3 shows the average number of anomalies generated across all four folds by different algorithms for different activities. Such as, for sleeping normal clustering produces fewer anomalies than other baselines. *Holmes* reduces the number of detected anomalies for sleeping from four to two. *Holmes* produces the minimum number of alarms for all activities in this data set. The statistical approach with temporal relations does not reduce the number of anomalies compared to the statistical approach without temporal relations mainly due to limited activities in this data set. The reduction in alarms by *Holmes* is mainly due to it's use of features based on collective activity instances and day of the week based clustering approach.

	Number of Detected Anomalies			
Activity	Stat	Stat+Temporal	Norm. Cluster	Holmes
Sleep	5	5	4	2
TV / Leisure	4	4	4	2
Out of Home	2	2	2	0

Table 3: Average detected anomalies per month (rounded) using 4-fold cross-validation for BeClose data set.

6. CONCLUSIONS

Anomaly detection systems for realistic and long-term in-home activities face many challenges including the inherent complexity in human behavior, correlations among activities, and many outside factors (e.g., weather, visitors). Current research works and actual deployed systems have too many false alarms because they do not address all of these issues. *Holmes* is the first system to address these challenges in an important, comprehensive, and novel way by using a combination of features based on both individual and collective activity instances, applying context-aware clustering and pattern mining algorithms, and learning semantic rules that explain deviation in behavior due to outside factors. Our evaluation shows that *Holmes* reduces the number of false alarms compared to existing techniques and to achieve this employs between 5 and 35 models depending on the complexity of the behavior of an individual (instead of just one model per activity used in most of the existing works).

7. ACKNOWLEDGEMENTS

This work was funded, in part, by NSF Grant CNS-1319302.

8. REFERENCES

[1] *The Elements of Statistical Learning*, chapter 14.3.12 Hierarchical clustering. Springer, 2009.
[2] Xyz. `http://www.XYZ.com/`, 2014.
[3] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *VLDB*, 1994.
[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *ICDE*, 1995.
[5] D. T. Anderson, M. Ros, J. M. Keller, M. P. Cuellar, M. Popescu, M. Delgado, and A. Vila. Similarity measure for anomaly detection and comparing human behaviors. *Int. J. Intell. Syst.*, 27(8):733–756, 2012.

[6] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. Technical report, Department of Computer Science and Engineering, University of Minnesota, 2007.

[7] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *Engineering in Medicine and Biology Society, 2005.*, pages 3551–3554. IEEE, 2006.

[8] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48(5), 2009.

[9] T. Dimitrov, J. Pauli, and E. Naroska. Unsupervised recognition of adls. In *SETN*, 2010.

[10] D. Elbert, H. Storf, M. Eisenbarth, O. Ünalan, and M. Schmitt. An approach for detecting deviations in daily routine for long-term behavior analysis. In *PervasiveHealth*, 2011.

[11] M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[12] K. Z. Haigh, L. M. Kiff, and G. Ho. The independent lifestyle assistant: Lessons learned. *Assistive Technology*, 18(1):87–106, 2006.

[13] Y. Han, M. Han, S. Lee, A. M. J. Sarkar, and Y.-K. Lee. A framework for supervising lifestyle diseases using long-term activity monitoring. *Sensors*, 12(5), 2012.

[14] M. R. Hodges, N. L. Kirsch, M. W. Newman, and M. E. Pollack. Automatic assessment of cognitive impairment through electronic observation of object usage. In *Pervasive*, 2010.

[15] E. Hoque, R. F. Dickerson, and J. A. Stankovic. Vocal-diary: A voice command based ground truth collection system for activity recognition. In *Proceedings of the Wireless Health 2014 on National Institutes of Health*, pages 1–6. ACM, 2014.

[16] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[17] V. R. Jakkula and D. J. Cook. Detecting anomalous sensor events in smart home data for enhancing the living experience. In *AAAI*, 2011.

[18] V. R. Jakkula, D. J. Cook, and A. S. Crandall. Temporal pattern discovery for anomaly detection in smart homes. In *IE*, 2007.

[19] T. Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *UbiComp*, 2008.

[20] T. Kleinberger, M. Becker, E. Ras, A. Holzinger, and P. Müller. Ambient intelligence in assisted living: enable elderly people to handle future interfaces. In *Universal access in human-computer interaction. Ambient interaction*, pages 103–112. Springer, 2007.

[21] N. D. Lane, Y. Xu, H. Lu, S. Hu, T. Choudhury, A. T. Campbell, and F. Zhao. Enabling large-scale human activity inference on smartphones using community similarity networks (csn). In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 355–364. ACM, 2011.

[22] D. Lewin, S. Adshead, B. Glennon, B. Williamson, T. Moore, L. Damodaran, and P. Hansell. Assisted living technologies for older and disabled people in 2030. *A final report to Ofcom. London. Plum Consulting*, 2010.

[23] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *UbiComp*, 2007.

[24] A. Lotfi, C. Langensiepen, S. M. Mahmoud, and M. J. Akhlaghinia. Smart homes for the elderly dementia sufferers: identification and prediction of abnormal behavior. *Journal of Ambient Intelligence and Humanized Computing*, 3(3), 2012.

[25] D. N. Monekosso and P. Remagnino. Anomalous behavior detection: Supporting independent living. In *Intelligent Environments*, Advanced Information and Knowledge Processing. SpringerLink, 2009.

[26] T. Mori, A. Fujii, M. Shimosaka, H. Noguchi, and T. Sato. Typical behavior patterns extraction and anomaly detection algorithm based on accumulated home sensor data. In *FGCN*, 2007.

[27] N. Noury, A. Fleury, P. Rumeau, A. Bourke, G. Laighin, V. Rialle, and J. Lundy. Fall detection-principles and methods. In *Engineering in Medicine and Biology Society, 2007. EMBS 2007.*, pages 1663–1666. IEEE, 2007.

[28] M. Novak, M. Binas, and F. Jakab. Unobtrusive anomaly detection in presence of elderly in a smart-home environment. In *ELEKTRO*, 2012.

[29] M. Novak, J. F., and L. L. Anomaly detection in user daily patterns in smart-home environment. *JSHI*, 3(6), 2013.

[30] J. Pei, J. Han, B. Mortazavi-Asl, and H. Pinto. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *ICDE*, 2001.

[31] P. Sukanya and K. S. Gayathri. An unsupervised pattern clustering approach for identifying abnormal user behaviors in smart home. *International Journal of Computer Science and Network*, 2(3), 2013.

[32] G. Virone. Assessing everyday life behavioral rhythms for the older generation. *Pervasive and Mobile Computing*, 5(1), 2009.

[33] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, R. Stoleru, S. Lin, and J. Stankovic. An assisted living oriented information system based on a residential wireless sensor network. In *Distributed Diagnosis and Home Healthcare, 2006. D2H2.*, pages 95–100. IEEE, 2006.

[34] J. Wu, A. Osuntogun, T. Choudhury, M. Philipose, and J. M. Rehg. A scalable approach to activity recognition based on object use. In *ICCV*, 2007.

[35] P. Wu, H. Peng, J. Zhu, and Y. Zhang. Senscare: Semi-automatic activity summarization system for elderly care. In *MobiCASE*, 2011.