Fault Detection for Binary Sensors in Smart Home Environments

Juan Ye, Graeme Stevenson, and Simon Dobson
School of Computer Science, University of St Andrews, St Andrews, Fife, UK
juan.ye@st-andrews.ac.uk

***Abstract*—Experiments in assisted living confirm that such systems can provide context-aware services that enable occupants to remain active and independent. They also demonstrate that abnormal sensor events hamper the correct identification of critical (and potentially life-threatening) situations, and that existing learning, estimation, and time-based approaches are inaccurate and inflexible when applied to multiple people sharing a living space. We propose a technique that integrates the semantics of sensor readings with statistical outlier detection. We evaluate the technique against four real-world datasets that include multiple individuals, and show consistent rates of anomaly detection across different environments.**

***Index Terms*—Wireless sensor network, fault detection, activity recognition, ontologies**

I. Introduction

Pervasive computing has the potential to revolutionise human life by using devices with sensing, intelligence, and communication capabilities to observe and respond to phenomena in an environment without human intervention. Foremost among pervasive scenarios is smart home research, where sensors are embedded and attached to all sorts of our everyday objects such as beds, mugs, appliances, and even our bodies. These sensors perceive the state of the physical environment through the interactions people have with these instrumented objects. By reasoning on these captured states, a pervasive system can infer what tasks residents are carrying out, and therefore automatically provide services to help achieve these goals without the need of direct or explicit guidance from the residents [6]. Smart home research has had a significant impact on many human beneficial applications among which ambient assisted living is one of the most exciting examples [4].

Many smart home prototypes have been developed to date, including the Aware Home [1], MavHome [31], Gator Tech Smart Home [12], and iDorm [7]. However the possibility of widespread deployment of such systems remains unclear. One reason among many (such as privacy and ethical issues) is the high likelihood of sensor anomalies. Researchers from the University of Virginia conclude their experience of deploying sensors as "Homes [...] can be hazardous for sensors, particularly when hundreds of sensors are deployed over long time durations" [14]. During their years of experimenting, they report an average of one sensor failure per day.

Pervasive sensors suffer from many types of technical limitation [21], including hardware failure, disconnection from the network, vulnerability to environmental interference, and limited battery life. However, in smart home environments, another major contributing factor is the presence of *humans*; that is, users (including both foreground users who are target subjects and background users who are active in the same environment like visitors, children, or even pets) might dislodge or move sensors accidentally [14], [16]. This often leads to *sensor anomaly*, where sensors do not fail, but continue to report values that are technically reasonable (e.g., the reported values are still within a reasonable range) but are unexpected or contradict the events occurring in the world.

Addressing sensor anomaly is more challenging than detecting broken sensors. Hnat et al. [14] discuss solutions towards addressing the latter problem. For example, they set a time interval that is reasonably long for any two consecutive data points. If no data is reported from a sensor within the interval since the last report, then this sensor can be considered broken. In contrast, sensor anomaly can be much more subtle and thus more difficult to detect. Simply setting fixed or variable time intervals cannot solve the problem, and often we need to validate collected values against either a data model that represents expected sensor values or values collected from neighbouring sensors via a correlation model.

This paper proposes *CLEAN* — a knowledge-driven technique to detect anomaly in event-driven binary sensors. We focus on two types of anomaly — *random* events that occur sporadically due to short or intermittent environmental interference; and *systematic* events where sensors consistently behave abnormally due to the sensor recalibration or dislodgement. We claim the following novelty and contributions:

- It is *knowledge-driven* in that it does not rely on any training data or annotated data and thus can be used from system initialisation and is not affected by changes in the patterns or routines of users' activities.
- It novelly combines knowledge and statistical models by using a well-defined knowledge as part of a clustering-based outlier detection technique. It is configured with a flexible and dynamic mechanism to configure and adjust thresholds at runtime, which reduces engineering effort in setting thresholds for different environments.
- It can detect multiple sensor anomalies simultaneously, and can scale up to a large number of sensors.
- It is not constrained by the number of users cohabiting the same environment.
- It is demonstrated to be widely applicable, through evaluation using four third-party real-world datasets with different sensor deployments, user profiles, and collection

periods.

The rest of the paper is organised as follows. Section II briefly discusses existing fault detection work in sensor network and identifies their limitations and difference from CLEAN. Section III introduces the proposed approach where we discuss similarity measure between sensor events and elicit a clustering-based outlier detection algorithm. The work is evaluated in Section IV and concluded in Section V along with the directions for future work.

II. Related Work

We briefly review sensor anomaly detection techniques in both wireless sensor network in general and smart home environments in particular. In the area of sensor networks, most fault detection techniques target sensors in environmental monitoring domains, usually involving temperature, humidity, light and wind speed. These sensors produce homogeneous, periodic, and real-numbered readings. Also these models often target individual sensors by tracking their historic values. However, the most common classes of sensors deployed in a smart home are infrared sensors, RFIDs, motion sensors, accelerometers, camera and microphones [28]. These sensors produce heterogeneous, event-triggered, and binary or featured readings. It is difficult to define a range limit over RFID readings, a seasonal pattern of sound, or an explicit numeric correlation between infrared sensor readings and acceleration data. We will review the fault detection techniques that deal with these two types of sensors and present how CLEAN addresses the problem and advances the state-of-the-art.

A. Failure detection in Wireless Sensor Networks

Fault detection is gaining more and more attention in wireless sensor networks, as longer-term deployment in real-world settings significantly increases [24]. Fault detection techniques can be categorised into four groups: rule-based methods, estimation methods, time series analysis, and learning-based methods [21], [24].

A rule-based method relies on expert knowledge about sensor readings to develop heuristic rules for identifying and classifying faulty sensor data. This method works best when the types of faults that can occur and the methods to detect them are known *a priori*. An early solution adopted by Mourand et al., defines ranges for valid sensor readings so as to exclude any observations falling out of reasonable domain limits [20]. This approach is mainly used to clean the data and thus to reduce the burden from the domain experts before performing any data analysis process.

An estimation method learns sensor correlations to predict normal behaviour of sensor observations. For physical phenomena like temperature or light, there often exist statistical correlations between sensor measurements. For example, correlations between measurements of sensors that monitor the same physical phenomena but are deployed at different locations, or correlations between measurements of sensors that monitor potentially related phenomena (e.g., temperature and humidity) but are deployed spatially together [9]. Gaussian processes have a long history of use to represent such spatial correlations between sensors [24]. We also assume that there exists a correlation between event-driven sensors which is not statistical between their values but semantical in the placed locations and attached objected of the sensors. We use such semantic relations to spot abnormal sensor events. For example, we could regard as abnormal a single firing of a sensor deployed in a bathroom among a collection of kitchen-hosted sensor firings.

Time series analysis builds a model for data streams collected by the same sensor to exploit their temporal correlations over a long-term period. To detect a fault, a sensor measurement is compared against its predicted value computed using time series forecasting. This method works best if the monitored physical phenomena such as temperature or light exhibits a certain periodic or seasonal pattern. Fang et al. [10] propose to use ARIMA – AutoRegressive Integrated Moving Average – to reduce errors in data collection while achieving energy efficiency. Each node learns an ARIMA model that will predict if the measurements sampled by the node are within a certain error bound. If the sampled measurement does not agree with the predicted measurement, then it will be further validated by a spatial model to determine whether the model is out of date, or the sampled data is faulty. The assumption of these techniques is that sensors report values in a fixed frequency, which does not hold for event-driven sensors that only report when a triggering condition is satisfied.

A learning-based method uses a certain amount of training data to derive a model for normal and faulty sensor readings and then, given an input sensor reading, statistically detects and identifies classes of sensor faults. It is usually integrated with the above estimation and temporal correlation based methods. Bayesian networks, Hidden Markov Models, and neural networks are the most common techniques applied. Dereszynski et al. [5] propose to use Bayesian Network for real-time fault detection and correction on temperature sensor stream collected in an ecological monitoring setting. The approach has two steps: *(i)* inferring a Bayesian network structure for each sensor deployment site, which captures spatial relationships between sensors and then *(ii)* extending the network structures to a Dynamic Bayesian network (DBN) to incorporate temporal correlations. The spatial and temporal correlations captured in different Bayesian networks can help to distinguish sensor failures from valid observations and as well as to predict the true values for the missing or corrupted readings. Similarly, Hill et al. [13] apply a DBN to analyse and diagnose anomalous wind velocity data. They build individual sensor models, on top of which a coupled DBN model is learned to represent the joint distribution of two sensors.

Paschalidis et al. [22] use Markov models to characterise the normal behaviour of sensor networks; that is, a Markov model at each sensor node is built to estimate anomaly-free probabilities from its past observation traces, and a tree-indexed Markov model is developed to capture their spatial correlations across the network. Based on derived optimal anomaly detection rules, the approach can assess whether its

most recent empirical measure is consistent with the anomaly-free probability model.

The need for training data is the main obstacle to the use of learning-based techniques for detecting abnormal event-driven sensors, which we expand on in the next section.

B. Detection Techniques in Smart Home Environments

In the area of smart homes, researchers propose top down, application-level methods to detect sensor faults [16], [23]. The principle is to look at how sensor failure affects reasoners. Such technique builds a performance profile for a set of classifier instances that are trained with all possible combinations of sensors. Detecting a sensor failure is achieved by comparing the runtime performance with these acquired profiles.

For example, Kapitanova et al. [16] train state-of-the-art classifiers (like Naive Bayes and Hidden Markov Model) to learn high-level human activities (like cooking) from a subset of sensors: excluding one sensor from the whole set of sensors at a time. Fault detection is performed by comparing the performance of these classifiers at recognising real-time activities. There are three main drawbacks to this method: *i)* its principle is to spot single sensor failure at a time, however in reality there could be multiple sensors failing simultaneously, *ii)* while the method might work on a small number of sensors (e.g., dozens) which is determined by the way they construct the classifier profile, and thus are unlikely to scale to hundreds or thousands of sensors, *iii)* sensors in the training data collection period never work the same in the long term, and neither do residents behave as expected. During the trial period, the sensors are typical in their best condition (e.g., fully charged and finely tuned) and the residents carefully (sometimes deliberately) interact with sensors. However when the sensors are not under the close watch of professional technicians, they frequently exhibit quite different behaviour. The proposed technique, CLEAN, overcomes these drawbacks.

III. Proposed Approach

We consider sensor anomaly detection as an outlier detection problem, where we assume the majority of sensor events function coherently and we try to detect the minority of sensor events that behave inconsistently from the majority. There are many different outlier detection algorithms [15], and here we choose an unsupervised solution, a clustering-based outlier detection algorithm – *FindCBLOF* [11]. This technique has been successfully applied to detecting abnormal network behaviours, and shares the above assumptions. The basic principles of FindCBLOF is as follows:

1) Cluster all the data points into groups, and sort the groups by their size in a descending order;
2) To each data point, assign a Cluster-Based Local Outlier Factor (CBLOF), which is a product of the size of the cluster that the point belongs to and the similarity between the point and the closest large cluster. The large cluster here means the cluster containing the majority of data points (say 90%). The CBLOF suggests the similarity between a data point and a cluster in a statistical way that represents the probability that the point belongs to the larger cluster. Any data point whose CBLOF is below a pre-defined threshold is considered an outlier. That is, the smaller the CBLOF, the less similar the point and the larger cluster are, and thus the point is more likely to be an outlier.

To adapt the *FindCBLOF* algorithm to detect anomaly in sensor events, we need to address the following four questions:

- What is the distance measure between two sensor events?
- How do we define a cluster as "large"?
- As sensor events are not static but streaming and continuous data, it is highly likely that abnormal sensor events occur repeatedly. How do we take into account the historic behaviour of sensors and combine it with the CBLOF?
- How do we set a threshold on the CBLOFs to decide which data points are outliers?

In the following we will propose strategies to address these questions.

A. Distance Measures between Sensor Events

Readings of event-driven sensors are binary, and the distance between these binary numbers alone has little meaning. However, each sensor can be characterised by its implicit semantics such as where the sensor is displaced, which object the sensor is attached to, and who the room or object belongs to. These semantics provide more information than the binary readings. In this section we follow the approach we proposed in [30] to characterise the semantics of a sensor event and quantify their distance measure. More technical details can be found in that paper.

We characterise a sensor event into semantic features $\{\texttt{t}, \texttt{l}, \texttt{o}, \texttt{u}\}$, describing that at the timestamp t, a sensor that is installed on an object o at a location l reports a reading about a user u. For example, a sensor event can be represented as `[2008-02-25T00:20:14Z, bedroom, door, main_user]`, indicating that the sensor installed at the door of the bedroom that belongs to the main user fires at the give timestamp. We adopt an ontological approach where we organise concepts in each feature space into a hierarchy based on their granularity level [29]. In the above example, `bedroom`, `door`, and `main_user` are concepts or instances in the Location, Object, and User feature space, and their relationships with the other peer concepts can be: $\texttt{bedroom} \sqsubseteq \texttt{sleeping\_area} \sqsubseteq \texttt{living\_environment}$, $\texttt{door} \sqsubseteq \texttt{movable\_structure}$ (from WordNet [19]), and $\texttt{main\_user} \sqsubseteq \texttt{any\_resident}$.

We can use the hierarchy to quantify the similarity measure of any two of its concepts. Wu et al. [26] propose a conceptual similarity function that works by finding the Least Common Subsumer (*LCS*) of the two input concepts and computing the path length from the LCS up to the root node. The LCS is the most specific concept that both concepts share as an ancestor. This is given by:

$$sim(c_1, c_2) = \frac{2 \times N_3}{N_1 + N_2 + 2 \times N_3},$$

where c_1 and c_2 are concepts in a feature space, N_1 (N_2) is the path length between c_1 (c_2) and the LCS node of c_1 and c_2, and N_3 is the path length between the LCS and the root.

When c_1 is equal to c_2, their LCS node is itself and the similarity is 1.0. When c_1 is semantically far from c_2, their LCS node might be close to the root in the hierarchy, which makes N_1 and N_2 large and N_3 small, so the similarity is close to 0. Therefore, the larger the similarity measure, the closer the two concepts. There exist other measures to quantify the distances between categorical values [2]. Most of them are based on frequencies of the values occurring in a certain dataset, which is not applicable in our approach.

Finally the distance between any two sensor events s_1 and s_2 can be defined as

$$1 - \sum_{i \in \{l,o,u\}} sim(s_1^i, s_2^i) \times \omega_i, (\sum_{i \in \{l,o,u\}} \omega_i = 1), \quad (1)$$

where sim is the above similarity function of the hierarchical concepts, and ω_i is the weight of each feature, which reflects the importance of each feature on capturing the similarity of two sensors. For example, if the activities of interest are location-specific, a higher weight can be placed on the location feature. To simplify, we uniformly apply across all the datasets the same weight on these three features; that is, 0.33. We use such similarity between sensors to spot the abnormal sensor events that is further distant from densely clustered events.

B. Ordering of Clusters

Once we have defined the distance measure between any two sensor events, we can cluster them. Let a sensor sequence contain a temporally-ordered list of sensor events. Clustering this sensor sequence will lead to multiple groups, some of which may correspond to different activities from different users while the others could contain the abnormal events. Therefore, it is not as likely that one cluster takes the absolute majority of data points as the original algorithm assumes. To solve the problem, we use the *shoulder-locating* method; that is, we order the clusters by their size, and an abrupt change in their sizes suggests a threshold for distinguishing large and small clusters. For example in Figure 1, we cluster a given sensor sequence into six clusters, and the percentages of their size to the whole number of data points in a descending order are 40, 38, 12, 5, 3, and 2. As we can see, the shoulder point is at the cluster whose percentage is 12%, where we observe the maximal difference between the size percentages. We then consider any cluster whose size percentage is above 12% to be large; that is, any cluster to the left of the dotted red line is considered as large and thus normal; i.e., Cluster 1 and 2. If the clustering results in one group or groups with identical sizes, the shoulder cannot be located; i.e., the percentage on the shoulder point will be zero. If so, we cannot find the minority of sparse points and thus we conclude there is no anomaly in this sensor sequence. If the gaps of sizes between groups are the same; e.g., a slope line, then the shoulder point will move downward to the smallest group. In this case, any other group except the smallest group is considered to be large.

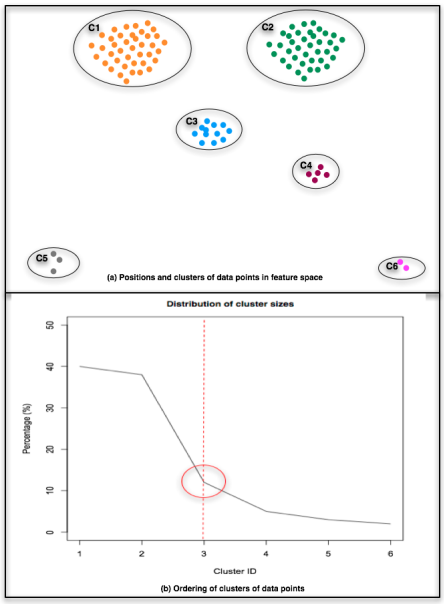

Fig. 1. Distribution of Cluster Sizes and the corresponding shoulder locating

This shoulder-locating method provides a flexible alternative to using a predefined fixed threshold (say 50% or 30%) to determine a "large" group. The reason is that the threshold always depends on the number of sensors deployed, the number of currently fired events, and activities being conducted at the moment. This also saves effort, either in terms of knowledge engineering or training, normally required to configure or adjust the best settings of the threshold. In Section IV we will demonstrate that we run CLEAN over four datasets without the need to re-configure any of these thresholds.

C. Considering Historic Sensor Behaviours

As mentioned earlier, abnormal sensor events may be persistent rather than one-off, especially for the systematic type of anomaly caused by technical degradation or dislodgement. Here we consider two extra factors: *frequency* and *temporality*. We assume that the more often and the more recent a sensor behaves abnormally, the more likely it is that a fault occurs again. We apply these two factors as a weight to *CBLOF* in an exponential function:

$$fw = e^{-f/N} \quad (2)$$

where N is the total number of sensor events being monitored and f is the number of times that abnormal events among the last N events are reported by a certain sensor. The choice of N depends on the intensity of sensing. In the following experiments, we set it universally to be 100; that is, we will look at how many times a sensor will function abnormally in

the last 100 events.

$$tw = e^{max(1,td/T)-1} \quad (3)$$

where td is the temporal distance between the current time and the reported event, and T is the range of time of interest (for example, one day is used in our experiment).

Finally for each data point, the extended CBLOF will be

$$size\_of\_cluster * sim\_to\_large\_cluster * tw * fw. \quad (4)$$

To decide the threshold below which the data point is considered as an outlier, it is unrealistic to set a fixed threshold because the number of sensor events varies with the activities being conducted by the users; for example, some activities like cooking tend to fire more sensors, while the other activities like sleeping only fires a limited number of sensors (e.g., the bedroom door) at its beginning and ending states. It also matters with the number of sensors being installed and the number of users living in the environment. Also different parameters in both temporal and frequency weight functions will make it difficult to fix the thresholds. To solve the problem, we re-use the shoulder-locating method; that is, we order all the CBLOFs in a descending order, and find the shoulder point where the maximum change is found. Then we use the CBLOF at this shoulder as the threshold. Any data point whose CBLOF is below this threshold is an outlier.

IV. Experiment and Evaluation

CLEAN is evaluated on four real-world datasets that are collected from different smart home environments with different human users and sensor configurations. These datasets capture typical activities and more importantly they represent common types of smart home datasets in terms of the number of sensors, the number of users cohabiting, and the degree of inherent noise. We believe that experimenting on these datasets gives us a comprehensive view of the effectiveness of the proposed technique.

The first two datasets are collected by the University of Amsterdam (named TVK A and TVK B respectively in the following) from two real-world, single-resident houses which were instrumented with wireless sensor networks. The sensor network in the first house is composed of 14 state-change sensors attached to household objects like doors, cupboards, and toilet flushes, while the network in the second house contains reed switches to measure whether doors and cupboards are open or closed; pressure mats to measure sitting on a couch or lying in bed; mercury contacts to detect the movement of objects (e.g., drawers); passive infrared to detect motion in a specific area; float sensors to measure the flush of toilet. All these sensors output binary readings (0 or 1), indicating whether or not a sensor fires. These two datasets are interested in the same set of activities, while the TVK B sensory data contains more noise than the TVK A data [17].

The third dataset is the PlaceLab Couple dataset [18]. To the best of our knowledge, this dataset is by far the most complicated and largest dataset collected in a real-world environment that is publicly available. The PlaceLab dataset contains over nine hundred sensor inputs, among which 707 are object sensors, including wireless infra-red motion sensors, stick-on object motion sensors, switch sensors, and RFIDs. The dataset was gathered over a period of 15 days during which a married couple (who were unaffiliated with the PlaceLab research) lived in the PlaceLab, generating 455,852 object sensor events in total. This dataset is not only composed of the highest variety of sensors but also contains many noisy events, which is due to the following three reasons: (1) the majority of the sensors (except RFID sensors) are not identity-specific, (2) they are very sensitive to environmental interference (e.g., motion detection sensors), and (3) this couple often perform interleaved activities and only one subject's activities have been annotated [18].

The fourth dataset is the interleaved activities of daily living (labelled as IAA) dataset from the CASAS smart home project [3]. This dataset was collected in a smart apartment testbed hosted at Washington State University during the 2009-2010 academic year. The apartment was instrumented with various types of sensors to detect user movements, interaction with selected items, the states of doors and lights, consumption of water and electrical energy, and temperature, resulting in $2,804,812$ sensor events. The apartment housed two people, `R1` and `R2`, who performed their normal daily activities during the collection period. This dataset will demonstrate CLEAN's performance in detecting abnormal sensor events in a multi-user environment.

A. Methodology and Measure

The overall goal of the evaluation is to assess the effectiveness of detecting *random* and *systematic* abnormal sensor events. The effectiveness is measured in *precision* – the percentage of the times of detected abnormal events actually noise being injected into the data, and *recall* – the percentage of the times of injected abnormal events being detected.

In the following experiments, we prepare the datasets by segmenting the sensor events into one-minute time slots, which is the most common segmentation technique in use [16], [25]. Algorithm 1 illustrates the random anomaly injection and evaluation process. For example, if we want to inject a P percentage of abnormal events into the dataset (i.e., the total number of abnormal events will be $P * N$, where N is the total number of sensor events in the original data). For each injection, we randomly *(i)* select a time slot, *(ii)* generate a timestamp within the interval of the time slot, and *(iii)* select a sensor id that is different from all the sensor ids originally contained in the time slot. Then we create a new sensor event with the timestamp and sensor id and inject it into the time slot. We repeat the injection process $P * N$ times, where P is chosen from 10% to 90%. For each percentage, we run I iterations (i.e., in our experiment, I = 100) and present the precision and recall over these iterations in a box plot. The box plot presents the precision and recall distribution of detection, including the minimum, maximum, and mean. This gives a more detailed and complete view than the averaged precision and recall. In the systematic detection, we randomly select a

Algorithm 1: Evaluation of Random Abnormal Events

Data: L: a list of one-minute segments of sensor events
I: the number of iterations
N: the number of sensor events in total
P: the injection rate of abnormal events
S: the number of sensors
Result: A: the detection precision and recalls
for $i \leftarrow 1$ **to** I **do**
  $IL = L$
  for $n \leftarrow 1$ **to** $P * N$ **do**
    $seg\_id = rand\_gen(1, size(L))$
    $ts = rand\_gen(L.get(seg\_id).start\_time, L.get(seg\_id).end\_time)$
    $found = False$
    while $!found$ **do**
      $sid = rand\_gen(1, S)$
      if $!L.get(seg\_id).contains(sid)$ **then**
        $found = True$
    $IL.get(seg\_id).inject(create\_event(timestamp, sid))$
  // run CLEAN over CL and put the evaluation accuracy into results
  $CL = clean(IL)$
  $A.append(eval(CL, L))$

number of sensors, and then for each randomly selected sensor we create a sensor event and inject it into each time slot. The number of sensors is chosen from 1 to half of the total number of sensors. For the PlaceLab dataset we only chose 10% of the sensors (71). For each number of selected sensors, we run 100 iterations and present the precision and recall over these iterations in a box plot. For example, if the number is 20, then the 100 iterations generates 100 combinations of sensors, each 20 in size.

In terms of the clustering algorithm, we use DBSCAN [8], which does not require pre-defined cluster sizes and is amenable to our needs – grouping events by their distance and neighbourhood density, which are set as 0.5 and 3 respectively. That is, we cluster events if their distance is close enough (i.e., within 0.5) and have enough close neighbours. This is the only place that we need to set up the thresholds for the clustering algorithm to work. As mentioned in Section III, we do not need to configure the thresholds to determine whether a cluster is large or whether an event is an outlier.

B. Random Anomaly Detection Results

Figure 2 presents the precisions of detecting random anomalies on these four datasets. The precision is consistently high across all the datasets, indicating that the detected events are indeed abnormal. Note that the assumption of the algorithm is that the majority of sensor events are normal, which suggests that the majority of sensor events are more likely to form into a cluster whose density is greater than the clusters that contain abnormal events. So, although we inject noisy events that are over 50% percent of the total number of sensor events, it is less likely that these events form a high-density group. We observe that the precision on the IAA dataset, which involves two residents, decreases more rapidly than the precision on the other datasets. The reason for this is that the injected noise might form a cluster which is not significantly different from the cluster representing a less active behaviour being conducted by a background user.

Figure 3 presents the recalls on these four datasets increase with the injection rate; that is, the more noise injected, the higher chance of detecting them. This sounds counterintuitive, and is partly due to the inherent noise in the datasets; e.g., we detect abnormal events that are not injected but already exist in the data. Our second experiment is to remove the potential noisy events from the original datasets first by running the same algorithm, and then by repeating the random noise evaluation process. Figure 4 and 5 present the precisions and recalls on the cleaned datasets, which are the ones where each segment contains non conflicting sensor events. Both the precisions and recalls have greatly improved, especially on the TVK B and PlaceLab datasets, which tend to have more noise than the other two [17], [18], [27]. Note that all these datasets are not annotated with noisy sensor data.

To characterise the potential noise in the original dataset, we present the frequency of sensors that have reported abnormal events on a daily basis from the original TVK B dataset in Figure 6. It is clear that sensor 28 consistently reports abnormal readings. We manually examine the raw data and find that this sensor is a passive infra-red sensor installed in the kitchen, which reports all the time, especially on the last day. CLEAN does not detect any abnormal events from this sensor because the user was not at home on that day and there was no other sensor firing. Without peer comparison, CLEAN always forms one cluster that contains only this sensor, thus the conclusion that there is no anomaly is drawn. This is the one drawback of CLEAN, which can be complemented by a rule based check; that is, if one sensor reports continuously over an long period (e.g., one day), then this sensor should be considered abnormal.

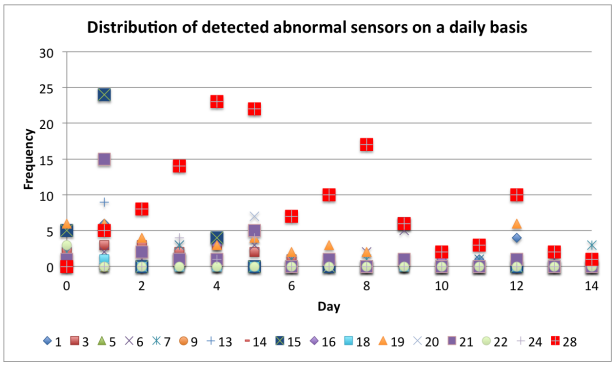

Fig. 6. Frequency of sensors that have been detected to report abnormal events on a daily basis in TVK B dataset

C. Systematic Anomaly Detection Results

Figure 7 and 8 present the precisions and recalls of detecting systematic sensor anomalies. We can see that there are fluctuations in both precision and recall, especially on the PlaceLab dataset. If a randomly generated sensor event is associated with a sensor that fires frequently throughout the dataset, and if that

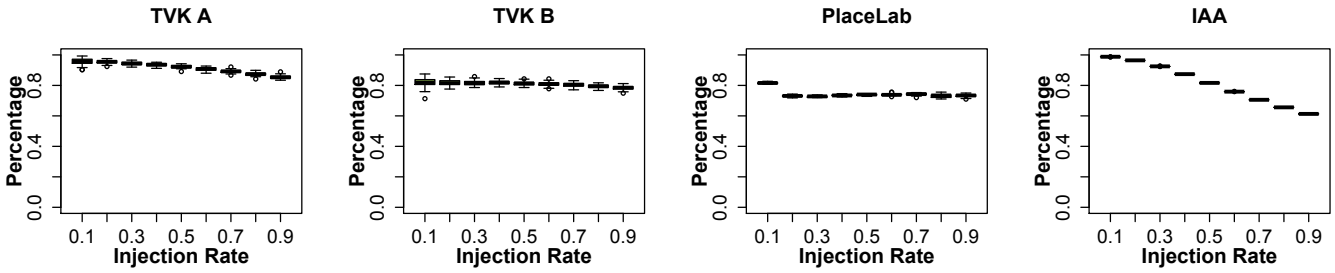

Fig. 2. Precisions of detecting random noise on the original datasets. The consistently high precision indicates that detected noise has a high chance of being the injected random noise.

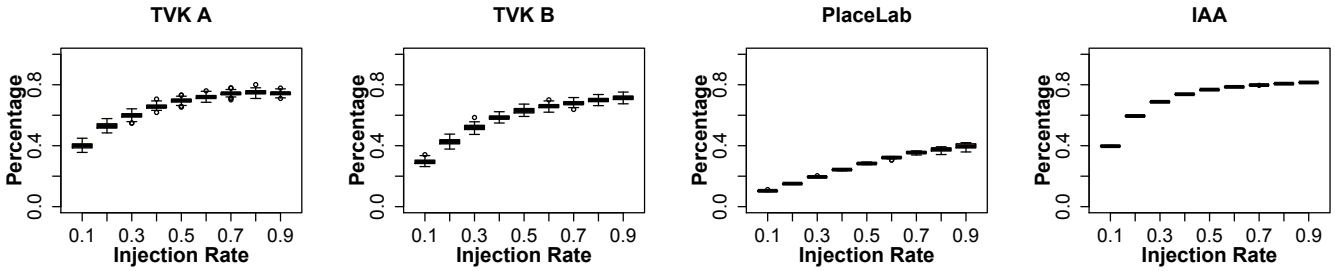

Fig. 3. Recalls of detecting random noise on the original datasets. The various recalls over different datasets are due to the potential noise in the original datasets.

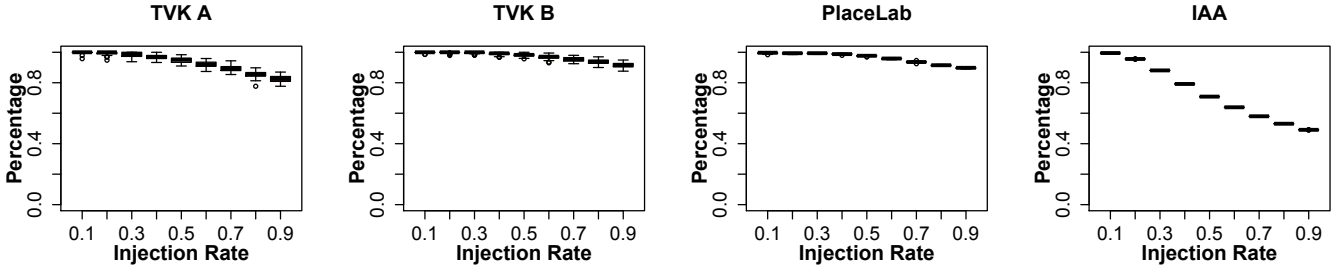

Fig. 4. Precisions of detecting random noise on the cleaned datasets. After cleaning the original dataset, the precision has increased on the first three datasets.

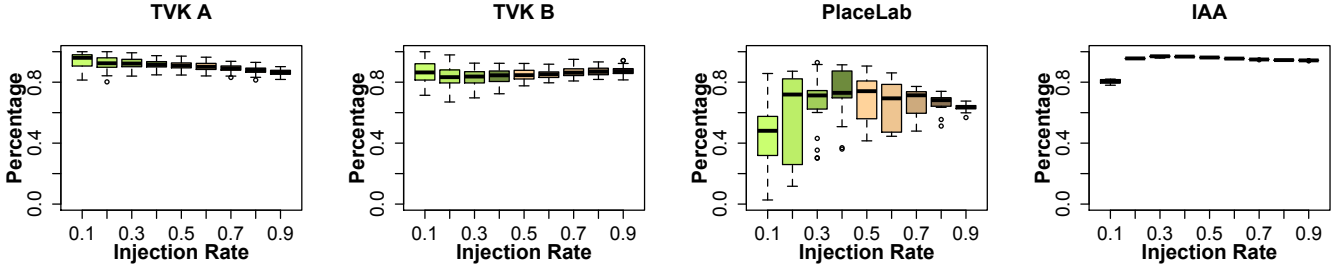

Fig. 5. Recalls of detecting random noise on the cleaned datasets. After cleaning the original dataset, the recall has increased on all the datasets.

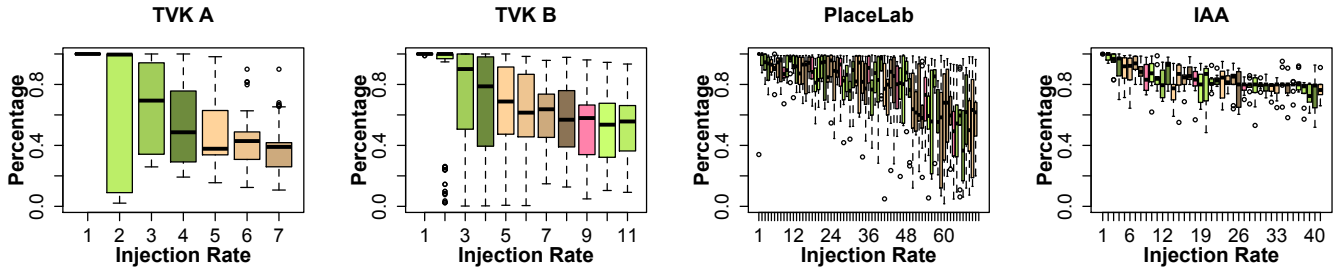

Fig. 7. Precisions of detecting systematic noise on the original datasets. Precision of detecting systematic sensor noise is highly influenced by the chosen sensors.

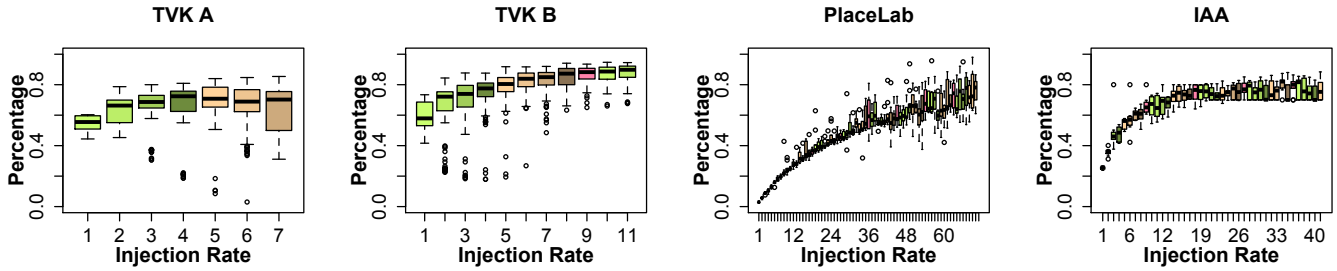

Fig. 8. Recalls of detecting systematic noise on the original datasets

sensor has similar semantics to a group of other firing sensors, then there is less chance of that event being detected as an outlier. This will affect the temporal and frequency weights as a result. On the other hand, if a generated event is associated with a sensor that is critical to a less frequent activity, then it is more likely that that this sensor event will be detected as being abnormal. For example, when we deliberately set the sensor on the front door – which only contributes to identifying the user's entry and departure from the house – in the TVK A dataset to report readings in every time slot, the detection precision and recall are 94.3% and 80% respectively. Because of these frequency and contribution factors of sensors, the precisions and recalls do not follow a more observable trend as that of detecting random anomaly. Similarly the detection precisions and recalls improve on the cleaned datasets, as shown in Figure 9 and 10. These two figures also show that the precision and recall have not significantly improved on the cleaned datasets, especially for PlaceLab and IAA which contain a large number of sensors and thus the randomly chosen sensors will lead to higher variability of the detection accuracies. This again is consistent with the earlier statement.

V. Conclusion and Future Work

This paper presents CLEAN, a technique that leverages sensor semantics in a statistics-driven outlier detection method to detect abnormal events, which does not rely on any training data nor requires ground truth annotation. It is generic in that it scales well with the number of sensors, can be deployed with single- or multi-resident environments, and can be integrated with existing activity recognition techniques. By taking the streaming sensor events gathered from various sensors as input, CLEAN detects and removes abnormal events, which can be used to filter data fed as input to any activity recognition algorithms. We demonstrate its detection performance on four real-world datasets with various environments, sensor deployments, the number of users living in an environment, and the extent of noise underlying in the dataset.

Currently, CLEAN is designed to detect "excessive" data, and cannot yet detect what "missing" data. Missing sensor data is subtle in that not reporting a value does not imply a sensor is broken, and could simply mean that the user has not interacted with the object that the sensor is attached to. For example in the activity of making coffee, we cannot conclude that the sensor on the sugar jar is not working just because it did not fire as the activity was carried out — the user could simply have chosen not to add sugar to the coffee. Our future work will look into correlations between sensors, and between sensors and activities, and integrate such knowledge with CLEAN to try to detect missing events.

References

[1] G. D. Abowd and E. D. Mynatt. *Designing for the Human Experience in Smart Environments*, pages 151–174. John Wiley & Sons, Inc., 2005.

[2] S. Boriah, V. Chandola, and V. Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the SIAM International Conference on Data Mining, SDM 2008, April 24-26, 2008, Atlanta, Georgia, USA*, pages 243–254. SIAM, 2008.

[3] D. Cook and M. Schmitter-Edgecombe. Assessing the quality of activities in a smart environment. *Methods of Information in Medicine*, 48:480–485, 2009.

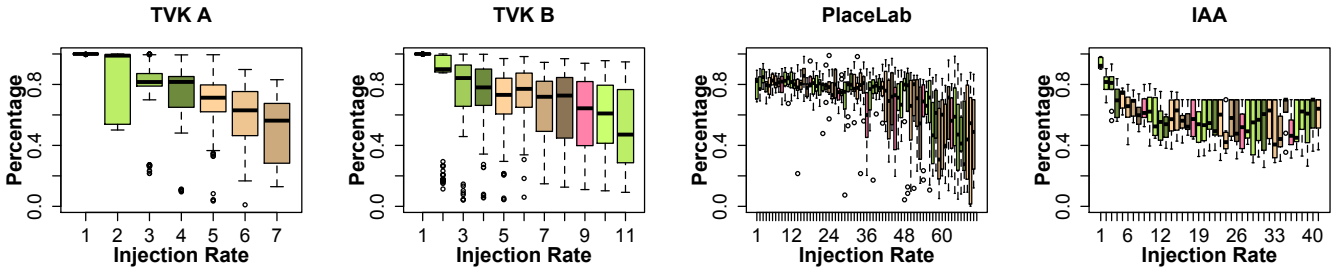

Fig. 9. Precisions of detecting systematic anomaly on the cleaned datasets. Precision has not significantly improved after cleaning.

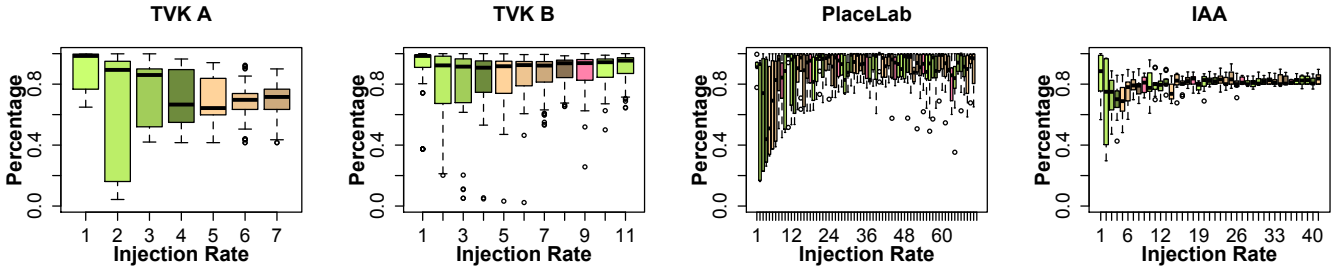

Fig. 10. Recalls of detecting systematic anomaly on the cleaned datasets

[4] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277 – 298, 2009.

[5] E. W. Dereszynski and T. G. Dietterich. Spatiotemporal models for data-anomaly detection in dynamic environmental monitoring campaigns. *ACM Trans. Sen. Netw.*, 8(1):3:1–3:36, Aug. 2011.

[6] D.J.Cook. How smart is your home? *Science*, pages 1579–1581, 2012.

[7] F. Doctor, H. Hagras, and V. Callaghan. A fuzzy embedded agent-based approach for realizing ambient intelligence in intelligent inhabited environments. *IEEE Transactions on SMC (A)*, 35(1):55–65, 2005.

[8] M. Ester, H. peter Kriegel, J. S, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD-96*, pages 226–231. AAAI Press, 1996.

[9] L. Fang and S. Dobson. In-network sensor data modelling methods for fault detection. In *Evolving Ambient Intelligence*, volume 413 of *Communications in Computer and Information Science*, pages 176–189. Springer International Publishing, 2013.

[10] L. Fang and S. Dobson. Unifying sensor fault detection with energy conservation. In *Proceedings of IWSOS'13*, 2013.

[11] J. Han, M. Kamber, and J. Pei. *Data Mining: Concepts and Techniques, 3rd Edition*. Morgan Kaufmann, 2011.

[12] S. Helal, W. Mann, H. El-Zabadani, J. King, Y. Kaddoura, and E. Jansen. The gator tech smart house: a programmable pervasive space. *Computer*, 38(3):50–60, March 2005.

[13] D. Hill, B. Minsker, and E. Amir. Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of IAHR' 07*, Marid, Spain, 2007.

[14] T. W. Hnat, V. Srinivasan, J. Lu, T. I. Sookoor, R. Dawson, J. Stankovic, and K. Whitehouse. The hitchhiker's guide to successful residential sensing deployments. In *Proceedings of SenSys '11*, pages 232–245. ACM, 2011.

[15] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22:2004, 2004.

[16] K. Kapitanova, E. Hoque, J. A. Stankovic, K. Whitehouse, and S. H. Son. Being smart about failures: Assessing repairs in smart homes. In *Proceedings of UbiComp '12*, pages 51–60, 2012.

[17] T. L. M. Kasteren, G. Englebienne, and B. J. A. Krose. Human activity recognition from wireless sensor network data: Benchmark and software. In *Activity Recognition in Pervasive Intelligent Environments*, volume 4 of *Atlantis Ambient and Pervasive Intelligence*, pages 165–186. 2011.

[18] B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Proceedings of UbiComp '07*, pages 483–500, 2007.

[19] G. A. Miller. Wordnet: a lexical database for English. *Commun. ACM*, 38(11):39–41, Nov. 1995.

[20] M. Mourad and J. Bertrand-Krajewski. A method for automatic validation of long time series of data in urban hydrology. *Water Science Technologies*, 45:263–270, 2002.

[21] K. Ni, N. Ramantahan, M. Nabil, H. Chehade, L. Balzano, S. Niar, E. Zahedi, S. Kohler, G. Pottie, M. Hansen, and M. Srivastava. Sensor network data fault types. *ACM Trans. Sensor Netw.*, 5, 2009.

[22] I. C. Paschalidis and Y. Chen. Statistical anomaly detection with sensor networks. *ACM Trans. Sensor Netw.*, 7, 2010.

[23] H. Sagha, J. Millan, and R. Chavarriaga. Detecting and rectifying anomalies in body sensor networks. In *Proceedings of the International Conference on Body Sensor Networks*, 2011.

[24] A. B. Sharma, L. Golubchik, and R. Govindan. Sensor faults: Detection methods and prevalence in real-world datasets. *ACM Trans. Sen. Netw.*, 6(3):23:1–23:39, June 2010.

[25] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *Proceedings of UbiComp '08*, pages 1–9, 2008.

[26] Z. Wu and M. Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, ACL '94, pages 133–138, Stroudsburg, PA, USA, 1994.

[27] J. Ye, L. Coyle, S. Dobson, and P. Nixon. Using situation lattices in sensor analysis. In *Proceedings of PERCOM '09*, pages 1–11, 2009.

[28] J. Ye, S. Dobson, and S. McKeever. Situation identification techniques in pervasive computing: a review. *Pervasive and mobile computing*, 8:36–66, Feb. 2012.

[29] J. Ye, G. Stevenson, and S. Dobson. A top-level ontology for smart environments. *Pervasive and Mobile Computing*, 7:359–378, June 2011.

[30] J. Ye, G. Stevenson, and S. Dobson. KCAR: A knowledge-driven approach for concurrent activity recognition. *Pervasive and Mobile Computing*, (0), 2014.

[31] G. Youngblood and D. Cook. Data mining for hierarchical model creation. *IEEE Transactions on SMC (C)*, 37(4):561–572, July 2007.