User Activity Recognition for Energy Saving in Smart Homes

P. Cottone, S. Gaglio, G. Lo Re, M. Ortolani

In Proceedings of the third IFIP Conference on Sustainable Internet and ICT for Sustainability

User Activity Recognition for Energy Saving in Smart Homes

Pietro Cottone, Salvatore Gaglio, Giuseppe Lo Re, and Marco Ortolani

DICGIM – University of Palermo
Viale delle Scienze, ed. 6 - 90128 Palermo, Italy
{pietro.cottone, salvatore.gaglio, giuseppe.lore, marco.ortolani}@unipa.it

***Abstract*—Current energy demand for appliances in smart homes is nowadays becoming a severe challenge, due to economic and environmental reasons; effective automated approaches must take into account basic information about users, such as the prediction of their course of actions.**

The present proposal consists in recognizing user daily life activities by simply relying on the analysis of environmental sensory data in order to minimize energy consumption by guaranteeing that peak demands do not exceed a given threshold. Our approach is based on information theory in order to convert raw data into high-level events, used to represent recursively structured activities. Experiments based on publicly available datasets and consumption models are provided to show the effectiveness of our proposal.

I. Introduction

Recent years have witnessed a steady increase in energy demand, both from industries and households; sustainable energy consumption for commercial and private buildings is consequently becoming a relevant issue, also due to economic and environmental motivations [1]. Concurrently, the design of Building Energy and Comfort Management (BECM) systems has grown to become a self-standing research area within the greater field of Ambient Intelligence (AmI) [2]. While the general scope of AmI is to apply techniques from artificial intelligence to transparently support users in their everyday activities, a BECM system can be defined more specifically as a control system that uses computers and a distributed sensor network for monitoring a building in order to ensure efficient usage of the available energy sources.

Current literature about building automation, however, shows that building control is still mainly performed manually, as in the case of artificial lighting setting, powering appliances on and off, or seasonal control of heating systems; additionally, automation in buildings has historically focused on narrow-scope tasks, such as lighting control with simple motion detection and a fixed timeout, or indoor climate control based on temperature and CO_2 level. On the other hand, user activities and behavior have considerable impact on the amount of consumed energy in all kinds of buildings (i.e., residential, office, and retail sectors). A significant amount of the energy dissipated in these areas can be saved by fine-tuning deployed devices and appliances according to actual user needs; for instance, many research efforts have been focused on proposing "smart thermostats" based on occupancy prediction, or on maximizing user comfort by providing appropriate artificial lighting, based on the activity carried on at a given moment.

In order to implement such approaches, predicting the users' course of actions is regarded as the most important input for building automation systems, as it is essential to cope with the issue of reducing energy consumption without negatively affecting the user experience. A typical problem, for instance, is due to the presence of peaks in energy demand, caused by the concurrent use of a set of appliances, whose combined power demand is higher than the maximum allowed consumption. This affects many energy providers, whose supply offers typically impose a limit on the maximum available instant power or apply additional costs when power consumption exceeds some prefixed quantity. As pointed out in [3], being dynamic should be an essential property of any BECM system, and no static assumptions should be made; also, such systems should be as unobtrusive as possible in order to improve user acceptance, coherently with AmI requirements.

The present work aims at recognizing daily life activities performed by users in a smart home in order to minimize energy consumption by guaranteeing that peak demands do not exceed a given threshold. We require the system to be able to work without an explicit human intervention, so a specific challenge is related to the *language* used to obtain a high-level, generalizable description of human behaviour. Our approach assumes that the only available data comes from the measurements of a typical environmental sensor network; moreover, we do not require specific knowledge of the particular application scenario. Modeling human activities from this perspective is a compelling open issue, as each of them may be loosely defined as a sequence of actions, which in turn may be expressed as movements or interactions with objects, captured by a specific series of sensory readings. If frequent patterns were extracted from such sequences, they might be used to characterize typical activities; however, such kind of data is often noisy and unreliable, so the mapping

of data onto high-level activities is not straightforward; furthermore, typical data mining approaches are affected by combinatorial explosion when confronted with the huge amount of sensory data to be processed.

Our proposal is to preliminarily convert sensory readings into meaningful *events*, by applying a lossy compression algorithm based on minimum description length; *patterns* representing activities are then extracted from the event sequence, and clustered in order to train a Hidden Markov Model, which implements an automated activity recognizer. Each activity is associated to its own typical consumption model, based on the involved home appliances, so that, once we reliably identify the current user activity, we can infer its impact on the energy consumption and we can act in order to prevent the arising of unwanted peaks. Our approach then consists in delaying the activation of those appliances whose operations are not strictly required by the user, until the combined demand for energy falls below some predefined threshold.

The remainder of the paper is organized as follows. Section II summarizes some of the approaches presented in literature, with regards to energy management, and activity recognition in smart homes. Section III presents our approach to energy saving by way of automated user activities recognition, and Section IV provides experimental assessment of our system. Finally, some concluding considerations are provided in Section V.

II. Related Work

The presence of peaks in energy demand is often symptom of a suboptimal scheduling of the use of electric appliances. As current electricity demand follows an ever-increasing trend, it is vital to find solutions for bounding the maximum amount of energy needed by the users at any given time, through approaches to consumption management aimed to stabilize the overall demand. Many works in literature [4]–[6] debate this problem, and it has been claimed that simply shifting or turning off unnecessary devices can bring significant advantages in terms of energy saving, without a relevant impact on users' comfort [7]. A drawback of most proposals, however, is their intrusiveness: such systems typically involve users in achieving an acceptable level of energy consumption, which is often too tiresome, time-demanding and ultimately unacceptable for users. For this reason, our system tries to adhere to the AmI philosophy which requires minimizing direct user intervention; the basic idea is that turning off devices or delaying their usage may be safely performed if the system can reliably assume that they are not required for on-going user activities.

This perspective requires that activity recognition should be automated, and common proposals include (i) methods based on the use of logic, (ii) probabilistic methods, or (iii) methods based on common sense reasoning. In the context of logic-based methods, activity recognition consists in reconstructing the plan an agent is following, based on the observation of its actions, and the main difficulty lies in the hypothesis of rationality, which often does not hold when the agent is human, especially in the presence of illness or disability. The authors of [8] use reticular theory and a logic language for describing actions in order to detect non-standard behavior; their system can generate new plans and provide explanation for unusual actions. In [9], event calculus is used to recognize activities and support users in performing the correct action at the right place and time. The significant advantage of using a logic language, such as event calculus, is the possibility to embed a-priori knowledge about the application domain, which reduces the need for annotations and allows for easy interpretation of the produced rules; the drawback, however, is the inability to deal with ambiguity, which arises when the system fails at detecting the on-going activity and cannot even estimate the most likely one.

Probabilistic methods regard the sequence of events as a time series, and the goal is to determine the chain of hidden states which generated the observations. The probabilistic approach requires computing the sequence which maximizes the probability of the hidden states, given a particular set of observations. Several methods, such as Semi-Hidden Markov Models, Skip Chain Conditional Random Fields, and many others, have been applied to address the issue of activity recognition, as reported in [10]–[12]. Probabilistic methods require the availability of a large amount of labeled data in order to show acceptable performance; the need for annotation may be partially mitigated by hard-coding knowledge about how activities are typically carried on, e.g. by extracting it from the Web. In [13], for instance, a system whose purpose is to create a database of bits of common-sense knowledge is proposed; such data may be integrated in automated systems in order to augment their ability of interacting with the real world. Translation of sensory data into high-level abstractions is made by merging knowledge with information from the Web, and transforming the obtained data into clauses; the system then performs statistical inference reasoning.

From a data mining perspective, activity discovery is often seen as the problem of detecting recurring patterns within a sequence of events; however, there are substantial differences between frequent itemsets detection, and discovery of patterns corresponding to activities. First of all, itemsets do not account for the ordering of the elements, which on the other hand is quite relevant during activity discovery; secondly, each itemset must not contain repetitions, whereas a pattern might do. In order to overcome such limitations, most proposals rely on the so-called *T-patterns* [14], and candidate itemsets are chosen according to criteria defining their meaningfulness within the event sequence. The authors of [15] use a variant of the *Apriori* algorithm [16] to discover sequences of events repeating with regular periodicity, besides patterns related to frequent activities. The system starts from elementary sequences and expands them to obtain longer

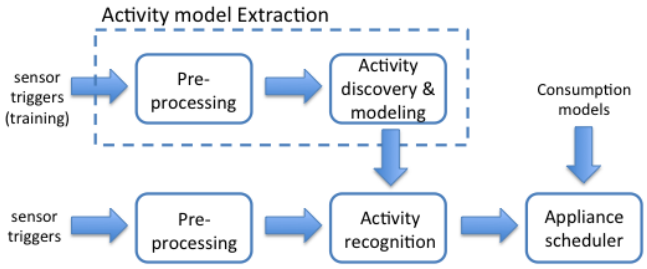


Figure 1: An overview of proposed system.

ones, up to a maximum predefined size. Another approach, proposed in [17], relies on standard Apriori and considers the event sequence as a stream of incoming data; after identifying all sequences of predefined size and support, a transform function maps them into models for activities. A hierarchical description is proposed for such models, and activities are divided into tasks and sub-tasks; the bottom of the hierarchy is represented by activities that cannot be further decomposed; activity recognition, as well as description, is carried on in a bottom-up fashion. A similar approach is described in [18], where the authors address the issue of broken or concurrent activities by considering *emerging patterns*, i.e. those patterns able to capture meaningful differences between two classes of data. Finally, an approach worth mentioning is proposed in [19], where activities of daily living (ADL) are discovered by means of evolutionary techniques; the purpose is the creation of an Evolving ADL Library containing models for activities; the library evolves by learning additional models from new sequences.

III. User Activity Recognition

The aim of our work is to level out peaks of energy consumption by identifying the appliances whose service is effectively needed by users, and postponing the use of the others until the combined demand for energy falls below some predefined threshold. To this end, reliable estimates of current activities of daily living must be available; however, recognizing them in a fully automated way is quite challenging.

It has been mentioned that solutions borrowed from data mining regard activity discovery as the issue of finding recurring patterns within the stream of sensory readings, or alternatively of extracting the ones corresponding to the activities of interest. Typically, a supervised method is used for creating predefined models for the most common activities, beginning by labelling training samples, and then looking for occurrences in actual data. A few assumptions are implicit with this strategy: all involved users are assumed to carry on the same set of known activities with similar regularity and demeanor; a considerable amount of data needs to be collected and consistently labelled, and finally incomplete or discontinuous patterns within original data are not well tolerated. In general, working directly on raw sensory data is likely to result in overfitting, i.e. the obtained representations are so overly specific for the employed dataset that they are not generalizable to different contexts or users. In fact, a current open issue regards transferring activity-related knowledge across different scenarios [20]–[22], as in most cases learned models cannot be easily exported to contexts with different characteristics (e.g., type or location of sensors, prior knowledge, and so on); the use of advanced simulators has been proposed to overcome the issue [23].

The idea behind our approach is to relieve the designer from the task of creating a detailed model for each activity to track, so, unlike previous proposals, we address this problem from an *algorithmic* perspective, rather than a *learning* one. We make use of data collected from an environmental sensor network with no detailed prior knowledge about the specific application scenario; careful programming of the network can ensure both robustness and efficiency of data gathering [24]. A general high-level description of what may be regarded as an *activity* is all is required, thus bypassing the difficulty of creating a reliable model of an activity in terms of sensory triggers or supposed interactions between users and their home appliances. Our system then attempts to infer models for activities defined as *recursive structures* and starts by identifying relevant *events*, which, in this context, may be thought of as short and recurrent sequences of raw sensor readings. Hence, the designer is not forced to embed knowledge into the system and may rather choose a description of events in terms of simple basic concepts, such as time duration or type of sensor measurement. We implicitly assume that "ground" events are characterized by a short duration, and will directly correspond to readings; most of them will likely be not very meaningful for characterizing user activities, and their information content will not be apparent unless they are considered in a combination with other ground events, thus having the hidden structure of the activity progressively emerge.

An overview of the proposed approach is shown in Figure 1: the outcome of the upper half block is the set of activity models that can be used by the activity recognizer, together with the events generated by the preprocessor, to feed the appliance scheduler; an energy-optimized usage plan for the appliances will eventually be produces, also taking into account standard energy consumption models. The remainder of the section will describe each module.

A. Preprocessing

We consider sensors of diverse nature, whose readings cannot be fed into our system without adequate preprocessing. Assuming a full-fledged environment, the performance of an activity will sparkle a great number of sensor readings; for instance, breakfast preparation may involve proximity sensors (to the cupboard, to the oven,

etc.), item sensors (toaster, coffeemaker, taps), and environmental sensors (temperature, water flow), whose state may be represented by a *binary*, *discrete* or *continuous* variable, respectively. We define a *trigger* as the (sensor$_{ID}$ sensor$_{state}$) pair. Representative information must be extracted from a series of raw sensor triggers; to this end, our approach was to devise a specific language, where an event is defined in terms of triggers according to the following syntax:

Ev$_{ID}$ ⟨dur$_{min}$ dur$_{max}$⟩ trig$_{ID}$ [, ⟨gap$_{min}$ gap$_{max}$⟩ trig$_{ID}$]

According to this definition, each event is identified by the minimum and maximum expected duration of the whole sequence, an initial trigger followed by an optional sequence of triggers with intervening gaps of duration in the range [gap$_{min}$, gap$_{max}$].

Our aim is to discover *templates* for events present in the series of triggers, and for this purpose we use a greedy probabilistic method based on a frequentist approach; given enough samples, it is possible to obtain a reliable estimate of the probability of each event template.

Initially, we just select the most frequent pairs of trigger occurrences via a sliding window algorithm that filters out pairs whose duration would not satisfy our search criteria; moreover, in order to select meaningful items, we impose additional constraints by applying a lower bound on the acceptable frequency:

$$\theta_{freq} = mean_{freq} + 2 \cdot dev_{freq}$$

where mean and standard deviation are computed over the frequencies of all pairs.

Pairs of triggers may already be considered as elementary event templates, and may be expanded by iteratively adding more triggers to them. In order to discover the most frequent triggers comprised within each pair (if any), we exploit the conditional probability that a trigger falls within a given pair. Upon adding a new trigger to a sequence, the algorithm looks for the next possible value maximizing the updated conditional probability; addition of a trigger may reduce, but never increase the number of occurrences of a sequence in the overall trigger sequence, so the iterative procedure will terminate when such number falls below a preset value.

As a final step, all basic events made up of a single trigger are added to the newly found templates, thus producing a complete list of templates sorted by their relative frequency in the sequence.

Discovering the possible list of event templates enables us to scan previously unseen trigger sequences in order to identify the actual occurrences of events contained therein.

This step is accomplished by *SAIL* (*S*tring m*A*tching with w*I*ldcards and *L*ength constraints) [25], an on-line algorithm able to locate patterns as soon as they appear in the sequence, which we modified to account for our representation for events and triggers.

The entire preprocessing algorithm is shown in Figure 2.

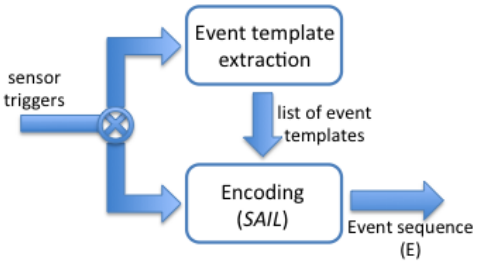


Figure 2: The preprocessor module.

Input: string E; int n_{min}, n_{max}
Output: alphabet **a**

1: **nlist** ⇐ extract_ngrams(E, n_{min}, n_{max})
2: **a** ⇐ ∅
3: **while nlist** ≠ ∅ **do**
4: **nlist** ⇐ sort(**nlist**)
5: **ngram** ⇐ getfirst(**nlist**)
6: **if** get_obtainable_compression(**ngram**) < θ_{comp} **then**
7: **return a**
8: **else**
9: **a** ⇐ **a** $\bigcup$ {**ngram**}
10: **nlist** ⇐ **nlist** − {**ngram**}
11: E ⇐ delete(E, **ngram**)
12: **nlist** ⇐ update(**nlist**, E, **ngram**)
13: **end if**
14: **end while**

Figure 3: Finding a better alphabet for event encoding.

B. Activity Discovery and Modeling

The use of *SAIL* transforms the *trigger* sequence into an *event* sequence, ready to be scanned to find frequent and relevant patterns, representing our high-level activities. In order to keep this problem manageable, and to cope with the complexity of exploring the search space, we rely on information theory. We aim at compressing the event sequence by lossy optimal coding so that events with low information content will be discarded; in other words, the most relevant patterns will be those that better describe the whole sequence, according to the Minimum Description Length (MDL) principle [26]; additionally, the compression of the event sequence allows for a decrease in the computational cost of later processing, thus coping with the exponential complexity of frequent event pattern mining.

Our algorithm for activity discovery is inspired to arithmetic coding and entropy-based compression. In order to find an optimal encoding for the event sequence E produced by *SAIL*, we regard it as a string of symbols over the alphabet of event IDs. Figure 3 shows the pseudocode for our algorithm.

Borrowing the terminology from information theory, an n-gram is a subsequence of n contiguous items from a given string, so our aim is to translate the original sequence using a new alphabet whose symbols are the most significant n-grams in E. Line 1 of the algorithm extracts the list of

n-grams of size between $n_{\min}$ and $n_{\max}$ together with their frequencies.

The algorithm then proceeds iteratively (lines 3-14). The n-grams are sorted according to the *MDL* principle: basically, each of them is viewed as a potential new symbol of the alphabet, and the length of string E is re-computed accordingly, using a binary encoding; the n-grams are sorted according to the degree of compression the can produce, and the n-gram producing the best compression is chosen (lines 4-5). In the following instructions, the frequencies of the remaining n-grams are updated, avoiding overlapping; the iteration stops when no n-gram is able to produce a compression rate above the chosen θ_{comp} threshold. Convergence is ensured since addition of an n-gram to the alphabet may only cause the frequencies of the remaining n-grams (hence, their potential compression rate) to decrease. The algorithm then returns the n-gram alphabet resulting in better encoding.

Once a shorter version E_R of the event sequence is obtained thanks to the new encoding, the most frequent patterns have to be discovered. Our approach is similar to *DVSM* (*D*iscontinuous *V*aried-order *S*equential *M*iner) [27], which is an Apriori-based iterative algorithm, relying on five main components: a candidate generation function, a pruning function, a candidate set, and a frequent pattern set. Initially, a candidate set is generated by considering the pruned set of all pairs of consecutive events in E_R. The idea of the algorithm is that each pattern in the candidate set is expanded at each iteration, according to a generation function. New patterns are checked against a pruning function, and only the ones surviving pruning are added to the new candidate set. Only those patterns whose expansions are all discarded (i.e. they are not "covered" by their expansions) will be part of the frequent pattern set. The algorithm stops when the candidate set is empty. The candidate generation function expands a pattern by adding the previous and the subsequent event in E_R, in order to create two new patterns. The pruning function is based on the MDL principle, and discards those sets of patterns unable to produce a sufficient compression rate for E, according to a predefined threshold.

In order to compute the compression rate, *DVSM* iteratively creates a hierarchical structure: at each step, variations of similar patterns in terms of the Levenshtein distance [28] are grouped together into general patterns. The compression rate of variations and general patterns are checked against two threshold values, C and C_v respectively:

$$\frac{1}{1+e^{A_v}} < C_v \qquad A_v = \frac{DL(D|a_i)*\Gamma_v(a_i)}{DL(D|a)*(1-\Gamma_g(a))} \tag{1}$$

$$\frac{1}{1+e^{A}} < C \quad A = \frac{DL(D)}{DL(a)+DL(D|a)*(1-\Gamma_g(a))} \tag{2}$$

where a is a general pattern, a_i one of its variations, $DL(\cdot)$ a measure of the description length and Γ is a continuity measure of the pattern, as in [27].

The final frequent pattern set returned by *DVSM* contains the most relevant patterns, which will be clustered into meaningful classes to obtain the discovered activities, by integrating temporal information with other features of interest, such as composition similarity, with an approach similar to [27]. This step is accomplished by k-medoids, a variant of the well-known k-means clustering, where representative points are bound to belong to the initial dataset. k-medoids uses a dissimilarity measure computed over all the possible pairs of points, giving it more robustness than traditional k-means measures (see [29]). As with k-means, the number of partitions is a parameter chosen by the user.

The chosen dissimilarity measure reflects our definition of pattern dissimilarity, according to the T-pattern model, which considers three main features: *causality*, *critical intervals* and *missing components*. Causality is expressed by the order of the events in the pattern: earlier part of the pattern can explain the presence of the later one; therefore, the more two patterns differ for their event sequences, the more it is probable they are instances of different activities. This function is implemented via the Levenshtein distance. Critical intervals deal with the relations between the distributions of components of a pattern; that is, this measure consider the time distances separating consecutive components. The corresponding function measures temporal information about the pattern element (time of the day, duration, etc) and, clearly, the distance between two different components. The last function computes the so-called missing components, that is the differences between the events present in two patterns; this function finds the best pair of corresponding events between two patterns, if any. In order to choose the best partitioning of the original pattern set, the algorithm is run multiple times with different initial random representative points. In the end, we choose the partition that achieves the best overall dissimilarity measure among the obtained clusters. Such clusters constitute the so-called *discovered* activities, i.e. activities emerging from collected data.

In the last phase, we encode the features of the obtained clusters into models representing the discovered activities. We adopt an approach based on boosting; we use hidden Markov models (HMM) [30] to describe activities, and we train an HMM for each activity we discovered, using the correspondent cluster set as training set. In the recognizing phase, a window of fixed size is slid over the input events, and an activity label is assigned to the last event in the window, according to the HMM that achieves the higher posterior probability in correspondence to that event.

The software modules involved in activity discovery and modeling are represented in Figure 4.

C. Recognizing Activities for Energy Saving

Once models for activities are available, our system may process an incoming stream of sensor triggers, convert them into event sequences, and use a sliding window on

Table I: The datasets used for testing the system.

Dataset	Features	Activities	Sensors
adlnormal	20 users (one at a time), about 6,000 sensor readings, 100 activity instances	5 activities (*Telephone use*, *Hand Washing*, *Meal Preparation*, *Eating and Medication Use*, *Cleaning*)	motion sensors, analog sensors for monitoring water and stove burner use, as well as software sensors (VOIP), and contact switch sensors on phone book, cooking pot and medicine container
aruba	1 user, about 6,000 sensor readings (out of 1,600,000 total), 120 activity instances (6,471 total)	11 activities: *Meal Preparation*, *Relax*, *Eating*, *Work*, *Sleeping*, *Wash Dishes*, *Bed to Toilet*, *Enter Home*, *Leave Home*, *Housekeeping*, *Resperate*	binary sensors: motion sensors and door closure sensors (temperature sensors were also present, but they were not used by the proposed system)
kast	1 user, 2,120 sensor readings and 245 activity instances spanning 28 days	7 activities (characterized by different time duration and different frequency): *Leave house*, *Toileting*, *Showering*, *Sleeping*, *Preparing breakfast*, *Preparing dinner* and *Preparing a beverage*	14 binary sensors deployed in the house, placed on doors, cupboards, refrigerator and a toilet flush.

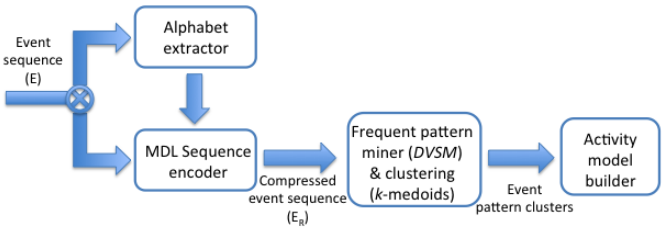


Figure 4: Activity discovery and modelling.

them in order to recognize the current activity; the label assigned to the last element of the window is that of the activity corresponding to the HMM that maximizes the posterior probability.

The proposed system uses the knowledge coming from recognizing user activities to cleverly manage typical appliances of a house (e.g. household appliances, refrigerator, air conditioning). Typical energy consumption can be divided into two principal components: the so-called *baseline* consumption, composed by the normal schedule of a typical home, and an *activity-driven* consumption, that is the energy spent to accomplish a specific task. The baseline consumption can be predicted by a typical schedule of the principal devices in a house, according to a prefixed user preference model. The activity-driven one is not predictable due to the extremely variable habits of people. Therefore, even if an accurate planning of the use was organized, it would be impossible to be sure that there will be no peaks in the total energy demand, due to the activity-driven consumption. Our system estimates the current energy usage and its evolution in the near future and checks that it is compatible with the activity the user is performing. When the total estimated energy use exceeds the prefixed threshold, the system temporarily turns off the minimum necessary amount of devices to satisfy the energy use constraint. Once the activity is over, the system attempts to turn back on a device; if that is not possible due to on-going excessive energy demand, the system checks for another device to turn off in order to trade for the reactivation of the old one. The time during when a device is off is thus minimized, causing as little impact on user comfort as possible.

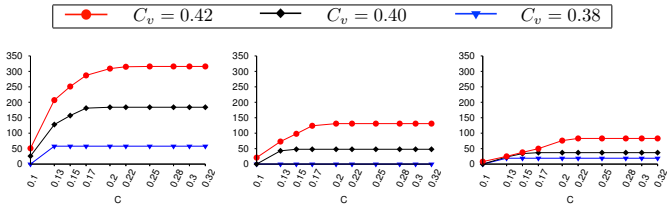
IV. Experimental Evaluation

In order to assess the performance of our system, we devised two sets of experiments specifically aimed at its main components and functionalities, i.e. activity discovery and recognition, and energy saving. We considered two reference scenarios; in the former case, we analyzed events generated by sensors deployed in a smart home environment, where each sequence of triggers was labeled according to the activities performed by the user, whereas for the latter we assumed that the system was able to control a predefined set of appliances, and we simulated an energy consumption scenario, according to a realistic energy use profile.

A. Assessment of Activity Discovery and Recognition

Three public datasets were used to measure the accuracy of the system: *adlnormal* [31], and *aruba* [32] (both from the CASAS project), and the one we named *kast* [10] from the Context Awareness in Residence for Elders (CARE) project. All datasets are annotated, i.e. their sensor trigger sequences are labeled with the activity the user was performing in correspondence to that portion of data: the so-called *actual* activities; however, the three datasets are very different with respect to the set of employed sensors and to the way the data was collected; their descriptions are reported in Table I.

In order to assess the ability of the system to correctly identify patterns of events, we checked its performance against the 3 datasets, with varying compression thresholds for the *DVSM* module (i.e. C influencing general patterns, and C_v for variations, see Eq. (1) and (2) on p. 5). Figure 5 shows that the algorithm performs similarly in all cases, but the resulting number of patterns is very threshold-dependent. Higher values for both thresholds increase the number of discovered patterns, up to a saturation point; the best performance is obtained with *adlnormal*, arguably due to the fact that test users



(a) *adlnormal* (b) *aruba* (c) *kast*

Figure 5: No. of extracted patterns as a function of the compression threshold C, parameterized on C_v.

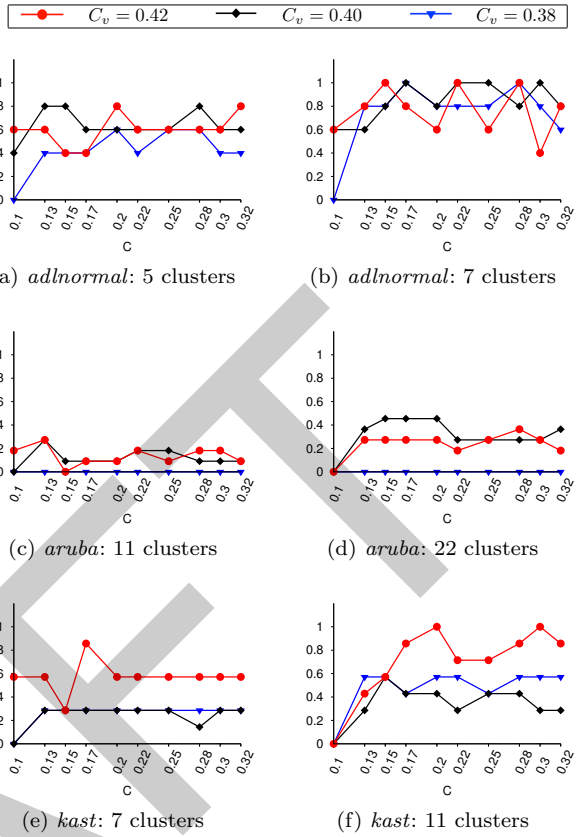
were instructed to simulate daily actions by following a preset script. A bad choice of thresholds may result in failing to discover any patterns at all, as is the case with $C_v = 0.38$ for *aruba*. The results show that appropriate values of C and C_v allow *DVSM* to prune most of the less meaningful patterns, also in combination with the preprocessing and encoding steps, that purge the input trigger sequence from non-significant data.

We also tested the performance of our k-medoids algorithm in producing meaningful classes of activities, in terms of the goodness of its clustering. To this end, we used the same metrics as in [27], namely:

- q_1: the ability to identify activities, computed as the ratio between the number of actual labels assigned to the *discovered* cluster representatives, and the total number of *actual* activities;
- q_2: the ability to assign correct labels to the extracted patterns with respect to actual activities, computed as the fraction of patterns actually belonging to the activity assigned to the cluster medoid, per each cluster.

The obtained results are shown in Figures 6 and 7 for different values of C, C_v; in order to assess the influence of the chosen number of clusters (k) on our metrics, we initially set this parameters equal to the number of actual activities for each dataset, and then increased it. The results show that q_1 is more sensitive to k than to the thresholds C and C_v, and higher values of k cause an increase in q_1, as is particularly evident in *adlnormal*. The worst performance is obtained on *aruba*, due to the presence of many unlabelled triggers, reflecting the fact that actual activities poorly correspond to the user's normal life; this is also highlighted by the results for q_2 on the same dataset, which show that when the cluster does represent an actual activity, its patterns are labeled in the correct way. For the other datasets, q_2 confirms the results from q_1, and shows good performance on accuracy in classification. The number of patterns does not influence this metric as much as it does for q_1, suggesting that increasing the number of clusters improves the "coverage" of our approach, but not the quality of produced clusters.

Finally, we assessed the accuracy of the HMM-based activity recognizer, with respect to discovered and actual activities. We aimed at computing the best values for



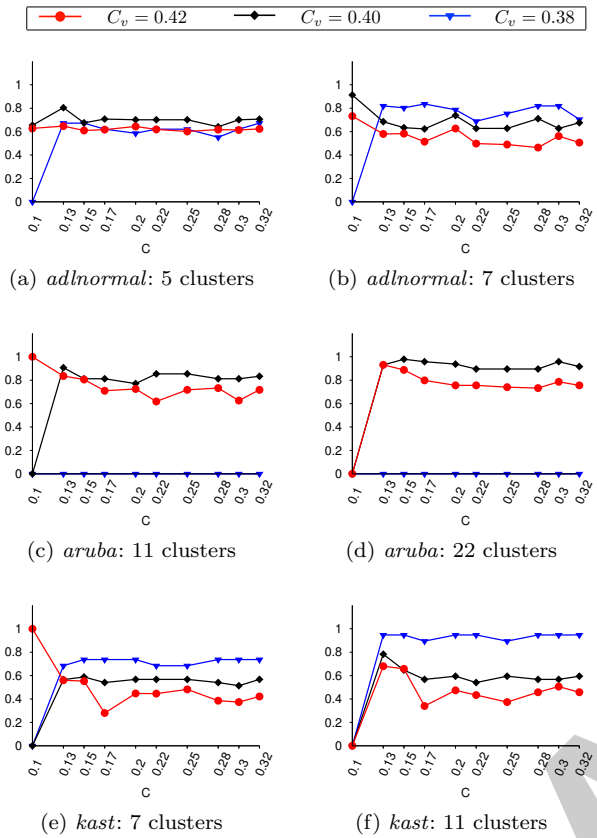
(a) *adlnormal*: 5 clusters (b) *adlnormal*: 7 clusters

(c) *aruba*: 11 clusters (d) *aruba*: 22 clusters

(e) *kast*: 7 clusters (f) *kast*: 11 clusters

Figure 6: Metric q_1 for the 3 different dataset as a function of the compression threshold C e C_v and number of clusters.

setting the HMM parameters, i.e. the number of hidden states (N), and the size of the sliding window (w); to this end, we used a grid search, with $N \in [3;15]$ and $w \in [3;15]$, and computed the accuracy of the system at each point in the grid. We conducted two separate tests, aimed at the recognition accuracy of actual and discovered activities, respectively. Results for the best configuration of parameters with respect to actual activities are shown in Table II, where the corresponding value for discovered activities is also shown. As expected, better results are achieved for actual activities in *adlnormal*, due to better correspondence between actual and discovered activities. The achieved accuracy is very high, confirming the capacity of our method of building reliable models. The results obtained for the *aruba* and *kast* show that our recognition system is able to create models of discovered activities with no assumption regarding the particular scenario. On the other hand, results on actual activities in these dataset suffer from the poor correspondence between discovered activities and actual activities. The setting for parameters N and w is also dependent on the specific dataset; such values need to be carefully chosen with respect the data at hand, as they basically represent how different activity definitions are mirrored into the corresponding datasets.



(a) *adlnormal*: 5 clusters (b) *adlnormal*: 7 clusters

(c) *aruba*: 11 clusters (d) *aruba*: 22 clusters

(e) *kast*: 7 clusters (f) *kast*: 11 clusters

Figure 7: Metric q_2 for the 3 different dataset as a function of the compression threshold C e C_v and the number of clusters.

B. Energy saving

In this section we prove the efficiency of our system in terms of energy saving, by showing its usefulness to prevent peaks of energy consumption. Our experiments were done on simulated data; an interesting study about the characteristics of this typical home appliances can be found in [33]. In our test scenario, we estimate the load curve from data based on average consumption of the most common appliances and on the statistics of their daily use (hour of activation, duration of the use, etc). For example, we are interested in the typical use of an appliance, that is the moment of the day when it is usually turned on and how long its typical utilization cycle takes; according to this data, a typical load curve is generated. The energy consumption deriving from the activity sequence performed by the user can then be summed up to the baseline load.

Table II: Best results in recognition accuracy.

	C	C_v	k	N	w	**Actual**	**Discovered**
adlnormal	0.17	0.38	7	4	12	0.95	0.98
aruba	0.30	0.42	11	6	3	0.66	0.92
kast	0.13	0.40	11	6	3	0.55	0.97

Table III: Appliances used for each activity.

Activity	**Appliances**
Cooking	Hobs, oven
Cleaning	Dishwasher
Eating	Coffeemaker
Phone call	Lamp

In our experiments, we assume that an average energy consumer is present at home and makes use of some known appliances to accomplish a set of predefined activities, which we extrapolated from the *adlnormal* dataset. Our system, based on the activity recognition results, is able to infer the current user activity, guiding the appliance scheduler so as to prevent peaks in energy demand. In this context, a peak is caused by instantaneous energy consumption exceeding a prefixed threshold. The correspondence between a single appliance or set of appliances and each activity is shown in Table III; the other appliances that we assume in use, even though not directly influenced by user activities, are: *Electric Heating and Circulation pump*, *Washing Machine*, *Tumble dryer*, *Refrigerator* and *Freezer*. In our experiments, the threshold to identify peaks was set at 3kW, and the proposed approach was able to reduce the number of the peaks of about 30%. For the final user this arguably results into a considerable saving in the energy bill and a more efficient curve of energy demand. Moreover, this result validates our approach and demonstrates that activity recognition can be very useful for energy saving. Figure 8 shows a slice of the original energy demand curve, and the one obtained when our system is in use. The effect of our system, in terms of avoiding peaks in energy demand, is evident in the new schedule of the baseline consumption. The rightmost part of the picture shows a more efficient distribution of energy demand as compared to the original one; this is obtained by postponing the use of appliances not involved in the activity the user is performing at a given moment.

V. Conclusion

This paper presented an information theory-based approach to recognizing daily life activities performed by users in a smart home, aimed at minimizing energy consumption by guaranteeing that peak demands do not exceed a given threshold. Our system is intended to be

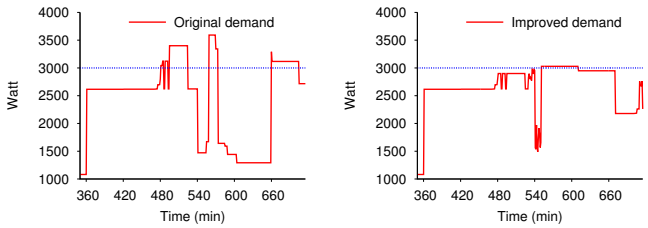


Figure 8: Comparison of original energy demand, and the one obtained after applying our approach.

completely automated in order to improve acceptance by the potential users. This poses a challenge as it is typically very difficult to discover high-level activities starting from raw sensory data, unless a very application-specific approach is used. We provided experimental results based on real data from public datasets, showing that we were able to obtain satisfying precision in modeling predefined activities via an unsupervised approach, and significant accuracy recognizing on-going activities. We finally showed how those results can be used to improve energy efficiency in a simulated smart home scenario, based on reliable energy consumption models.

References

[1] B. Bose, "Global warming: Energy, environmental pollution, and the impact of power electronics," *Industrial Electronics Magazine, IEEE*, vol. 4, no. 1, pp. 6–17, 2010.

[2] A. De Paola, S. Gaglio, G. Lo Re, and M. Ortolani, "Sensor9k: A testbed for designing and experimenting with WSN-based ambient intelligence applications," *Pervasive and Mobile Computing*, vol. 8, no. 3, pp. 448–466, 2011.

[3] T. A. Nguyen and M. Aiello, "Energy intelligent buildings based on user activity: A survey," *Energy and Buildings*, vol. 56, no. 0, pp. 244 – 257, 2013.

[4] M. Erol-Kantarci and H. Mouftah, "Wireless sensor networks for domestic energy management in smart grids," in *25th Biennial Symposium on Communications (QBSC)*,, 2010, pp. 63–66.

[5] M. Milenkovic and O. Amft, "Recognizing energy-related activities using sensors commonly installed in office buildings," in *SEIT 2013: Proceedings of the 3rd International Conference on Sustainable Energy Information Technology*, ser. Procedia Computer Science, Elsevier. Elsevier, In Press.

[6] O. Sianaki, O. Hussain, and A. Tabesh, "A knapsack problem approach for achieving efficient energy consumption in smart grid for endusers' life style," in *Innovative Technologies for an Efficient and Reliable Electricity Supply (CITRES), 2010 IEEE Conference on*, 2010, pp. 159–164.

[7] J. V. Paatero and P. D. Lund, "A model for generating household electricity load profiles," *International Journal of Energy Research*, vol. 30, no. 5, pp. 273–290, 2006.

[8] A. B. B. Bouchard and S. Giroux, "A logical approach to adl recognition for alzheimer's patients," in *Proceedings of the 4th International Conference on Smart homes and health Telematic (ICOST'06)*. IOS press, 2006, pp. 1–8.

[9] L. Chen, C. Nugent, M. Mulvenna, D. Finlay, X. Hong, and M. Poland, "Using event calculus for behaviour reasoning and assistance in a smart home," in *Proceedings of the 6th international conference on Smart Homes and Health Telematics*, ser. ICOST '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 81–89.

[10] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," in *Proceedings of the 10th international conference on Ubiquitous computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 1–9.

[11] S. P. Rao and D. J. Cook, "Predicting inhabitant action using action and task models with application to smart homes," *International Journal on Artificial Intelligence Tools*, vol. 13, pp. 81–100, 2004.

[12] D. Hao Hu, S. J. Pan, V. W. Zheng, N. N. Liu, and Q. Yang, "Real world activity recognition with multiple goals," in *Proceedings of the 10th international conference on Ubiquitous computing*, ser. UbiComp '08. New York, NY, USA: ACM, 2008, pp. 30–39.

[13] W. Pentney, A.-M. Popescu, S. Wang, H. Kautz, and M. Philipose, "Sensor-based understanding of daily life via large-scale use of common sense," in *Proceedings of the 21st national conference on Artificial intelligence - Volume 1*, ser. AAAI'06. AAAI Press, 2006, pp. 906–912.

[14] M. S. Magnusson, "Discovering hidden time patterns in behavior: T-patterns and their detection." *Behav Res Methods Instrum Comput*, vol. 32, no. 1, pp. 93–110, Feb. 2000.

[15] P. Rashidi and D. Cook, "Keeping the intelligent environment resident in the loop," in *Intelligent Environments, 2008 IET 4th International Conference on*, july 2008, pp. 1 –9.

[16] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, pp. 207–216, June 1993.

[17] B. Chikhaoui, S. Wang, and H. Pigot, "A frequent pattern mining approach for adls recognition in smart environments," *Advanced Information Networking and Applications, International Conference on*, vol. 0, pp. 248–255, 2011.

[18] H. K. Pung, "epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition," *2009 IEEE International Conference on Pervasive Computing and Communications*, pp. 1–9, 2009.

[19] J. A. Iglesias, P. Angelov, A. Ledezma, and A. Sanchis, "Human activity recognition in intelligent home environments: An evolving approach," in *Proceedings of the 2010 conference on ECAI 2010: 19th European Conference on Artificial Intelligence*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2010, pp. 1047–1048.

[20] P. Rashidi and D. J. Cook, "Activity knowledge transfer in smart environments," *Pervasive Mob. Comput.*, vol. 7, no. 3, pp. 331–343, Jun. 2011.

[21] T. L. M. van Kasteren, G. Englebienne, and B. J. A. Kröse, "Transferring knowledge of activity recognition across sensor networks," in *Proceedings of the 8th international conference on Pervasive Computing*, ser. Pervasive'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 283–300.

[22] D. H. Hu and Q. Yang, "Transfer learning for activity recognition via sensor mapping," in *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Three*, ser. IJCAI'11. AAAI Press, 2011, pp. 1962–1967.

[23] A. Lalomia, G. Lo Re, and M. Ortolani, "A hybrid framework for soft real-time wsn simulation," in *Proc. of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications*. IEEE, 2009, pp. 201–207.

[24] L. Gatani, G. Lo Re, and M. Ortolani, "Robust and efficient data gathering for wireless sensor networks," in *Proceedings of the 39th Annual Hawaii International Conference on System Sciences. HICSS'06.*, vol. 9. IEEE, 2006.

[25] Chen, Gong, Wu, Xindong, Zhu, Xingquan, Arslan, Abdullah, He, and Yu, "Efficient string matching with wildcards and length constraints," *Knowledge and Information Systems*, vol. 10, no. 4, pp. 399–419, Nov. 2006.

[26] J. Rissanen, "Minimum description length principle," in *Encyclopedia of Machine Learning*, C. Sammut and G. I. Webb, Eds. Springer, 2010, pp. 666–668.

[27] P. Rashidi, D. J. Cook, L. B. Holder, and M. S. Edgecombe, "Discovering Activities to Recognize and Track in a Smart Environment," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, pp. 527–539, 2011.

[28] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," Tech. Rep. 8, 1966.

[29] S. Theodoridis and K. Koutroumbas, *Pattern Recognition, Third Edition*. Orlando, FL, USA: Academic Press, Inc., 2006.

[30] L. Rabiner and B. Juang, "An introduction to hidden markov models," *ASSP Magazine, IEEE*, vol. 3, no. 1, pp. 4–16, 1986.

[31] D. Cook and M. Schmitter-Edgecombe, "Assessing the quality of activities in a smart environment," *Methods of Information in Medicine*, vol. 48, no. 5, pp. 480–485, 209.

[32] D. J. Cook, "Learning setting-generalized activity models for smart spaces," *IEEE Intelligent Systems*, vol. 27, no. 1, pp. 32–38, 2012.

[33] R. Stamminger and U. B. L. Fakultät, *Synergy Potential of Smart Domestic Appliances in Renewable Energy Systems*, ser. Schriftenreihe der Haushaltstechnik Bonn. Shaker, 2009.