Learning Accurate Temporal Relations from User Actions in Intelligent Environments

Asier Aztiria[1] Juan C. Augusto[2] Alberto Izaguirre[1] and Diane Cook[3]

[1] University of Mondragon, Mondragon, Spain
{aaztiria;aizaguirre}@eps.mondragon.edu
[2] University of Ulster, Jordanstown, United Kigndom jc.augusto@ulster.ac.uk
[3] Washington State University, Pullman, Washington, U.S.A cook@eecs.wsu.edu

Summary. Ambient Intelligence environments depend on their capability to learn user's preferences and typical behavior. In this paper we present an algorithm that taking as starting point information collected by sensors finds out accurate temporal relations among actions carried out by the user.

1 Introduction

Ambient Intelligence (AmI) [2] [8] [14] can be understood as 'a digital environment that proactively, but sensibly, supports people in their daily lives' [3]. Such systems can improve the life of users in many ways: for example, by making an environment safer, more comfortable and more energy efficient. These environments should achieve such goals without creating any extra burden to the users so as to maximize users' acceptance. Let us consider for example aims like adjusting the temperature automatically, turning off the lights if the user has gone out of the house, or issuing an alarm when it detects an unsafe situation. In order to accomplish these aims, the environment will need information about user preferences and habits (i.e., their 'normal' behaviour).

This paper makes a contribution to the problem of discovering patterns of activity. The problem is inherent to any intelligent environment. Here we will focus on a Smart Home scenario but the results can be easily extrapolated to similar environments. The next section summarizes previous related work. Section 3 explains the nature of the collected data and how patterns are represented. Then in Section 4 we explain how those patterns are obtained using the algorithm $\mathcal{A}_{PUBS}$ which is part of PUBS (Patterns of User Behaviour System). Section 5 shows the results of the validation experiments. Section 6 explains our plans for current and future work in this system and finally we provide our conclusions in Section 7.

2 Related Work

Learning is an essential feature in any AmI system. However, given the diversity of elements in AmI systems, learning has not been devoted as much attention in the literature as it may require. Even so, some notable exceptions have been. Artificial Neural Networks were the first technique used to infer rules for smart homes, and a survey of those works can be found in [7]. A first attempt was made in the MavHome project to predict the next smart home inhabitant action using pattern discovery and Markov model techniques [5]. Jakkula and Cook [11] extend this work to predict actions using temporal relations, defined by means of Allen's temporal logic relations [1].

Other techniques, such as Fuzzy-Logic in iDorm [10], Case-Based Reasoning in MyCampus [13] or Decision Trees in SmartOffice [9] have been used. Taking into account the characteristics of each problem we can state that each problem favours the use of a certain technique, but as Muller pointed out [12] 'the overall dilemma remains: there does not seem to be a system that learns quickly, is highly accurate, is nearly domain independent, does this from few examples with literally no bias, and delivers a user model that is understandable and contains breaking news about the user's characteristics'.

3 Data Collected, Required patterns and their Formalization

Our approach aims at obtaining user patterns from sensor data. Different sensors provide different types of information; therefore the learning process has to consider each accordingly. Three main different groups of sensors are considered (we are aware there are more types of sensors, e.g., alarm pendants, RFID, etc. but these are the relevant ones for the examples listed in this paper).

- (type O) Sensors installed in objects (devices, furniture, domestic appliances, etc). They provide direct information about user actions. For example, a sensor installed in the bedroom lamp may indicate when that lamp was switched on and off.
- (type C) Context sensors. These sensors provide information about context and not about user actions directly. Temperature, light and smoke sensors are examples of type C sensors.
- (type M) Motion sensors. These sensors can be used to infer where the user is (in the bedroom, outside the house, etc.).

Let us consider the event sequence illustrated in Figure 1. As the figure shows, sensors installed in the objects provide clues about users' actions that can be used to define patterns such as the one defined in (1):

'Bedroom Lamp is turned on 5 seconds after Motion Bedroom is turned on If and When bLight is <10' (1)

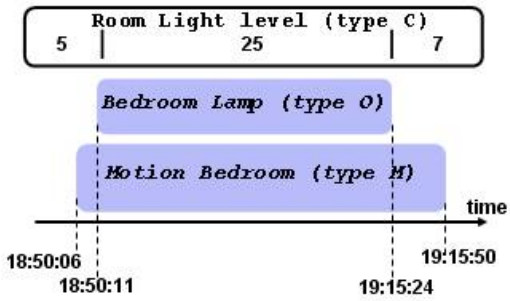

Fig. 1. Event sequence and context information

In order to use a clear and non ambiguous representation which can facilitate future automation, patterns will be described as ECA (Event-Condition-Action) rules [4]. They will capture situations where an Action that is detected through an activated sensor, called main sensor triggering (*mainSeT*) (e.g., Bedroom Lamp), is related (in terms of time, e.g., 5 seconds) to an Event involving another activated sensor, called the associated sensor triggering (*associatedSeT*) (e.g., MotionBedroom), if some Conditions (e.g., Bedroom light <10) are true. For more details see [6].

4 Learning temporal patterns with $\mathcal{A}_{PUBS}$

The process to identify patterns in data collected by sensors is summarized in the following algorithm:

$\mathcal{A}_{PUBS}$ Algorithm (for learning patterns)

---

for each sensor of type O (consider it as *mainSeT*)
  Identify the *associatedSeT* of type O or M (See Section 4.2.1)
  for each *associatedSeT*
    Identify possible time relations (See Section 4.2.2)
    if there exists a time relation **then** make it more accurate using context information, i.e., by using C-type (See Section 4.2.3)

Notice the emphasis on sensors type O as *mainSeT* given they are those more closely linked with user's explicit and intended actions.

4.1 Identifying associated sensor triggering

The aim of this first step is to get a list of possible related sensors (associated) in order to minimize the complexity of the learning process. For the purpose of discovering possible *associatedSeT*, we search for previous events of other sensors that happened before each event related to the *mainSeT*. If

$\mathcal{A}_{PUBS}$ discovers that before an event instance of *mainSeT* there are frequent occurrences of an event of another sensor triggering, then the later will be considered as *associatedSeT*. Finding an *associatedSeT* does not mean definitively there will be a pattern that describes a relation *associatedSeT* - *mainSeT*, but indicates there could potentially be one.

A list of possible *associatedSeT* is obtained with a similar approach to the Apriori method for mining association rules with only two differences.

- Limit possible associations to the object we are analyzing (*mainSeT*).
- The result does not consider a pair (*mainSeT*, *associatedSeT*) as a sequence, but only as sensors that can be potentially related in a meaningful way.

A modified Apriori algorithm (adding the aforementioned constraints) has been used in this step. As in every association mining process, minimum coverage, support and window size values must be provided. Let us consider the *mainSeT* 'Bedroom Lamp' again; the result of this first step will be a list of *associatedSeT*s.

$$BedroomLamp = >[MotionReception,\ MotionBedroom,\ LuxoLamp]$$
$$where\ n < TotalSensorNumber$$

4.2 Identifying time relations

Once we know what other actions triggering sensors could be related to actions triggering *mainSeT*, the next step is to discover if there are possible meaningful relations. Those relations could be either quantitative (2) or qualitative (3).

'BedroomLamp is turned on 5 sec. after MotionBedroom is turned on' (2)

'BedroomLamp is turned on after MotionBedroom is turned on' (3)

Quantitative relations include a specific measurement (e.g., 5 seconds) whereas qualitative relations emphasize the relative occurrence of the actions/events (e.g., after). Numerical relations carry extra valuable information with regards to qualitative ones which are useful for automation. For example, knowing the user likes to listen to the radio news at a particular time or to have the temperature of the house within 20°C-23°C at a particular month of the year. Qualitative relations like 'after dinner and before going to bed' are not constrained enough to schedule a favorite TV program. Our work has focused on numerical relations, as we know other reports in the literature had covered qualitative relations.

Grouping instances

The starting point for discovering numerical time relations will be to generate a table listing time distances between occurrences of *mainSeT* events and previous appearances of *associatedSeT*. Consider the scenario in Figure 2 where the relations between Bedroom Lamp and *associatedSeT*s are depicted.

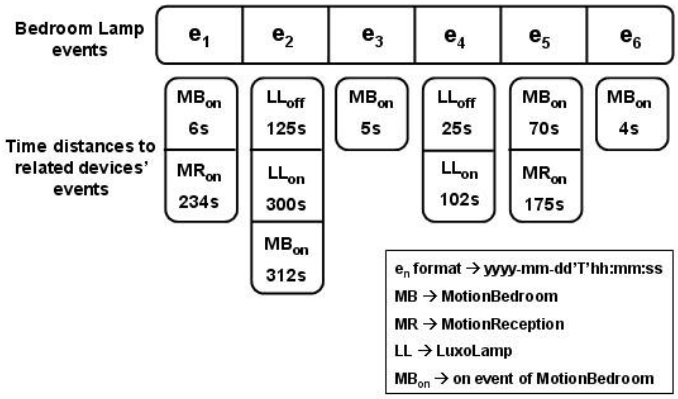

Fig. 2. Time distances between *mainSeT* and *associatedSeT*

Once the temporal relations are collected, the next step is to find out if there is any interesting time distance that is repeated frequently enough (see subsection 'extracting patterns' for more details). In order to discover these patterns, groups are made taking into account similarities among them. In our case, considering the *associatedSeT* MBon (MotionBedroom's 'on'), the distances would be ({e1,6s} {e2,312s} {e3,5s} {e4,-} {e5,70s} {e6,4s}) and groups among (6,312,5,70,4) must be done. The technique to make groups could be as complex as we can imagine and different techniques could be suggested to accomplish this task. In this case the technique used is based on joining values that are within a range established by:

$$[min, max] = \overline{x} \pm (\overline{x} * tolerance) \qquad where \qquad \overline{x} = \frac{\sum_{i=1}^{n} a_i}{n} \qquad (1)$$

with: tolerance = tolerated deviation from $\overline{x}$ (%); a_i = time distance of a element; and n = number of elements

If a value does not fulfill the requirements to join any group, a new group is created using that value as the group mean. Every time a new value is added to a group the mean value of that group is recalculated. Considering the possible relation between Bedroom Lamp and MBon, the process of making groups will be (considering 50% as tolerance percentage in all cases).

(e_1,6s); There is no group, create(group0, $\overline{x}$ (6s), [3,9])
Tolerance = 6 ± (6 * 0.5)

(e_2,312s)≠[3,9], create(group1, $\overline{x}$ (312s), [156,468])
Tolerance = 312 ± (312 * 0.5)

(e_3,5s)=[3,9], join(group0, $\overline{x}$ (5.5s), [2.75,8.25])
Tolerance = 5.5 ± (5.5 * 0.5)

(e_4,70s)≠[3,9] and 70≠[156,468] create(group2, $\overline{x}$ (70s), [35,105])
Tolerance = 70 ± (70 * 0.5)

(e_6,4s)=[2.75,8.25], join(group0, $\overline{x}$ (5s), [2.5,7.5])
Tolerance = 5 ± (5 * 0.5)

Extracting patterns

Once groups are made, the following step is to evaluate what information groups do and do not reveal about a pattern. As in the first step (identify *associatedSeT*) we consider interesting patterns to be those that cover more instances than established by the minimum confidence level. If the minimum confidence is, say, 25% the only group considered as interesting in our case would be group 0, which covers 3 instances out of 6, creating a pattern:

'BedroomLamp is turned on 5s after MotionBedroom is turned on' (4)

4.3 Identifying appropriate conditions of pattern occurrence

Every pattern has a confidence level that indicates how many *mainSeT* instances are covered by the pattern. The pattern (4) has a confidence level of 50% because it covers the instances e_1, e_3 and e_6 out of a total of 6. It means that only in 3, out of 6 times, the Bedroom Lamp was switched on when there was a motion detection at bedroom 5 seconds before. Having such a confidence level (rarely a confidence in this point will be close to 100%) does not mean that a pattern is not useful, but it means that it is not as accurate as it would be desirable. Finding out (if possible) under what conditions a pattern appears or not will be the last step to increase accuracy in patterns detection.

The possible conditions are given by: (a) Time specifications (e.g., such as: time of day, day of week, season, etc.) and (b) Context sensors (type C sensors such as those to measure temperature, light, humidity, etc.).

Adding conditions

In order to discover the conditions, two tables, *covered* and *non-covered* tables, are generated. In the *covered* table there will be instances classified well by the pattern together with the context information collected when they happened. The *non-covered* table will contain instances where the pattern fails. Dividing both tables, using the information they contain helps to provide more accurate patterns. Some example of these conditions are 'when time is between 18hs and 20hs', 'when day of the week is Saturday', 'when temperature is less than 25°C' or 'when light level is less than 10'.

Let us consider the tables in Figure 3 show the context information collected when instances of our example happened. Separating both tables will allow us to know when our pattern defines properly the relation between Bedroom Lamp and Motion Bedroom. In this case the easiest way to separate *covered* and *non-covered* tables seems to be by using the sensor bLight that

indicates the light level in the bedroom when action happened. Adding that condition, our pattern will be:

'BedroomLamp is turned on 5s after MotionBedroom is turned on When BedroomLight level is <10' (5)

Adding these conditions do not increase the instances the pattern can classify well, (still classifyies the same instances, 3/6), however we make sure that it does not include instances that do not have that pattern.

non-covered

	e_2	e_4	e_5
time of day	17:30:28	16:05:37	16:17:22
day of week	monday	tuesday	wednesday
bTemp Sensor	26	27	24
bLight Sensor	12	15	13

covered

	e_1	e_3	e_6
time of day	18:50:12	17:15:30	19:05:10
day of week	monday	tuesday	thursday
bTemp Sensor	25	26	22
bLight Sensor	5	6	5

Fig. 3. *covered* and *non-covered* tables

Patterns achieved up to now explain when and under what conditions instances of *mainSeT* happen. That will allow us to understand the user's behavior, but this is insufficient if the objective is to automate a house. For example, we may know that every time the bedroom lamp was turned on and the bedroom light level was less than 10 units, there was a motion sensor triggering in the bedroom 5s before. But this does not inform us about the converse relation, i.e. we cannot make sure that every time there was a motion sensor triggering in the bedroom and the bedroom light level was less than 10 units, 5s after that there was a bedroom lamp sensor activation. Therefore, it is necessary to analyze the pattern from the opposite point of view and find out the conditions to be able to automate. The way of getting these conditions is the same as before. The only difference is that to generate the tables, *associatedSeT* instances are considered instead of *mainSeT* instances and an *associatedSeT* instance is considered as covered if there is an instance of *mainSeT* just as the pattern indicates.

Classification technique

In order to separate the tables and get the conditions, we have used a modified JRip algorithm (JRip is a rule learning algorithm provided by Weka) [15]. The unique modification done in the algorithm has been made to get always conditions related to the *covered* table, i.e., JRip algorithm provides

rules without taking into account if they are related to one class or another, its unique aim is to separate classes. Because in our case separation gives us the conditions under which the pattern is useful, it is desirable that the classification technique always provides rules about the *covered* class.

5 Results

$\mathcal{A}_{PUBS}$ has been validated by applying it to artificial data generated by the authors and then to a real dataset collected from MavPad, a smart apartment that was created to test algorithms as part of the MavHome project [16]. The dataset we used was collected in three different time periods: Trial1 (spanning 15 days), Trial2 (spanning almost 2 months) and Trial3 (spanning 3 months). The types of sensors installed are: 26 sensors on objects such as lamps, lights or outlets, 53 context sensors such as light, temperature or humidity, and 37 motion sensors distributed in all the rooms.

Figure 4 shows the number of patterns discovered by $\mathcal{A}_{PUBS}$ in different trials, considering different minimum confidence levels (25%, 50%, 75% and 100%). A 25% confidence level means that, at least, a quarter of the instances must be covered by the pattern to consider it as interesting. On the other hand, the number of accurate patterns indicates the number of patterns out of total where it has been possible to define conditions to know when they can be used.

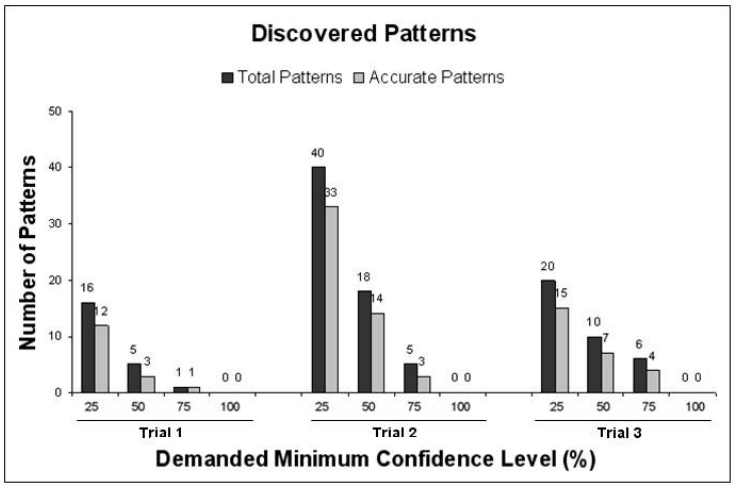

Fig. 4. Number of patterns and accurate patterns obtained in different trials considering different confidence levels.

Some conclusions can be extracted from these results. First of all, it seems clear that it is almost impossible to define patterns associated to a specific

object based on only one relation (in fact, there is no pattern with 100% confidence level) so the importance of defining conditions is clear. Moreover figures show in most of the patterns (92 out of 121) it has been possible to define conditions of occurrence in order to obtain patterns with relatively high levels of accuracy. The system can be run successively with different requests of accuracy to find the optimal balance in between a high accuracy in the detection and achieving patterns with minimum level of confidence. Table 1 shows the runtime needed by $\mathcal{A}_{PUBS}$ in each trial.

Table 1. Experiments' runtime considering different confidence levels

Confidence level (%)	**Trial 1**	**Trial 2**	**Trial 3**
25	45.7s	336.2s	137.5s
50	30.1s	227.4s	101.7s
75	26.8s	194.2s	88.3s

6 Future Work

Once a methodology to learn patterns is clear, the way of getting the objectives of each part can vary and different techniques or approaches can be suggested. Our short-term efforts will be aimed to discover patterns with qualitative relations, getting complex patterns where more than two activated sensors would be related and experimenting with some other datasets. Further future work will also include the possibility of incorporating user preferences and user feedback which will provide a useful heuristic to achieve much more personalized and user-adapted patterns.

7 Conclusions

Learning in AmI systems is a task that must be carried out as unobtrusively as possible, for example, by considering the information collected from sensors as the sole (or at least primary) source of information for user's actions. Discovering patterns on how objects are used, the places visited and the time associated with those events is informative on user's behaviour and preferences. This in turns helps an environment to accomplish the task of providing a service tailored to the inhabitants of the environment.

A three step algorithm, $\mathcal{A}_{PUBS}$, to find out temporal relations has been described, where the first two steps are focused on finding out relations among activated sensors and defining what type of relation these are, and the third step attempts to define when or in what conditions they must be used.

Results obtained after validating $\mathcal{A}_{PUBS}$ on artificially generated dataset and datasets collected by colleagues in a Smart Home indicate the importance

of including conditions as part of a pattern specification and also showed the efficacy of $\mathcal{A}_{PUBS}$ to find such patterns in substantially large datasets with acceptable performance.

References

1. J. Allen. Towards a general theory of action and time. In *Artificial Intelligence*, volume 23, pages 123–154, 1984.
2. J. C. Augusto. *Ambient Intelligence: the Confluence of Ubiquitous/Pervasive Computing and Artificial Intelligence*, pages 213–234. Intelligent Computing Everywhere. Springer London, 2007.
3. J. C. Augusto and D. J. Cook. Ambient intelligence: applications in society and opportunities for ai. In *20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, 2007.
4. J. C. Augusto and C. D. Nugent. The use of temporal reasoning and management of complex events in smart homes, 2004.
5. J. C. Augusto and C. D. Nugent. Smart homes can be smarter. In *Designing Smart Homes. The Role of Artificial Intelligence*, pages 1–15, 2006.
6. A. Aztiria, J. C. Augusto, and A. Izaguirre. Spatial and temporal aspects for pattern representation and discovery in intelligent environments. In *Workshop on Spatial and Temporal Reasoning at 18th European Conference on Artificial Intelligence (ECAI 2008) (to be published)*, 2008.
7. R. Begg and R. Hassan. *Artificial neural networks in smart homes*, pages 146–164. Designing Smart Homes. The Role of Artificial Intelligence. Springer-Verlag, 2006.
8. D. J. Cook and S. K. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley-Interscience, 2005.
9. C. Le Gal, J. Martin, A. Lux, and J. L. Crowley. Smartoffice: Design of an intelligent environment. *IEEE Intelligent Systems*, 16(4):60–66, 2001.
10. H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an ambient-intelligence environment using embedded agents. *IEEE Intelligent Systems*, 19(6):12–20, 2004.
11. V. R. Jakkula and D. J. Cook. Using temporal relations in smart environment data for activity prediction. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.
12. M. E. Muller. Can user models be learned at all? inherent problems in machine learning for user modelling. In *Knowledge Engineering Review*, volume 19, pages 61–88. Cambridge University Press, 2004.
13. N. M. Sadeh, F. L. Gandom, and O. B. Kwon. Ambient intelligence: The mycampus experience. Technical Report CMU-ISRI-05-123, ISRI, 2005.
14. M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, 1991.
15. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, 2nd ed.* Elsevier, 2005.
16. G. M. Youngblood, D. J. Cook, and L. B. Holder. Managing adaptive versatile environments. In *IEEE International Conference on Pervasive Computing and Communications*, 2005.