A Hybrid Preference Learning and Context Refinement Architecture

Korbinian Frank[1], **Patrick Robertson**[1], **Sarah McBurney**[2], **Nikos Kalatzis**[3], **Ioanna Roussaki**[3] and **Marco Marengo**[4]

Abstract. Pervasive computing envisions a world where people are surrounded by numerous communication and computing interconnected devices that are invisible and assist users in their everyday tasks in a seamless unobtrusive manner. Most pervasive computing research initiatives aim towards the realization of smart spaces, i.e. fixed spaces that provide pervasive features in a static and geographically limited environment. To bridge these isolated pervasive spaces, the EU project Persist has introduced the concept of self-improving Personal Smart Spaces (PSSs) that follow their owners wherever they go. This paper provides an overview of the context and preference learning facilities that have been designed to support the realization of PSSs and enhance their proactivity and self-improvement features.

1 Introduction

Pervasive computing [22] [21] is a next generation system paradigm that aims to assist users in their everyday tasks in a seamless unobtrusive manner. It assumes that users are surrounded by numerous communication and computing devices of various features, which interoperate and are capable of capturing and processing information regarding users, their behaviour and their environments. This information is used to establish context-awareness features [16] and to enable the provision of personalized context-aware services [12]. In this framework, there have been various research initiatives aiming towards the design and realization of smart spaces [23] in homes, offices, universities, schools, hospitals, hotels, museums, and other private or public places, where various automation facilities support the users. Nevertheless, these are fixed spaces that provide pervasive features in a static and geographically limited environment. They are alike independent "islands" of pervasiveness in a sea of legacy service provisioning systems. When the users exit these "islands" no pervasive computing features are offered. To bridge this gap, the notion of self-improving Personal Smart Spaces has been introduced by the Persist project.

Persist is a European research project (http://www.ict-persist.eu/) funded under the Seventh Framework programme that aims to couple the facilities offered by next generation mobile communications with the features provided by the static smart spaces to support a more ubiquitous and personalised smart space that is able to follow the user wherever he/she goes. A Personal Smart Space (PSS) will provide to its owner multimodal intelligent interfaces, via which he/she will be able to access and configure the various services and resources that are available locally and remotely, even when limited or even no network connectivity is available. PSSs will be able to discover other PSSs and interact with them in order to create a richer and more flexible environment for their owners. Each PSS consists of multiple devices, both mobile and fixed, owned by a single user. As the owner of the PSS moves to different locations and places, his/her PSS interact with other mobile or fixed PSSs located in the owner's surrounding environment, aiming for a unique support for pervasive service provisioning. PSSs constantly monitor their owner's behaviour environment and they exploit learning techniques to further optimise the pervasive experience perceived by their owner's. In a nutshell, a Personal Smart Space can be defined as a set of services within a dynamic space of connectable devices, where the set of services are owned, controlled, or administered by a single user. It facilitates interactions with other PSSs, it is self-improving and is capable of pro-active behaviour. Thus, a PSS is user centric and controlled by a single user, it is mobile (at least from user perspective), it allows interactions with other PSSs and is capable of self-improvement and proactivity.

In order to establish the proactivity and the self-improvement features of PSSs, quite sophisticated learning facilities are required. These facilities need to support both the learning of the user preferences, in order to enable the provision of personalized services, as well as the inference of future or currently unavailable context information. This paper elaborates on the mechanisms that have been designed in order to support context and preference learning and automated extraction.

The rest of this paper is structured as follows. In Section 2, a library-based hybrid architecture for preference learning and context refinement is presented, which is designed so as to demonstrate properties such as: transparency, modularity and pluggability. Section 3 elaborates on the context refinement algorithms. In this respect, the following are described: Bayesian Filters that refine the location accuracy, a clustering technique used to discover recurring locations, a history-based context prediction mechanism, Bayesian high-level context inference and a proximity estimation method that is based on a diffusion model. Subsequently, in Section 4, three different preference learning algorithms are described: the if-then-else rule learning, the online incremental preference learning and the Bayesian learning of user behaviour. Finally, in Section 5 conclusions are drawn and an overview of further research challenges regarding learning in personal smart spaces is provided.

[1] German Aerospace Center (DLR), Germany, email: {korbinian.frank | patrick.robertson}@dlr.de
[2] Heriot-Watt University Edinburgh, UK, email: S.M.McBurney@hw.ac.uk
[3] National Technical University of Athens, Greece, email: {nikosk | nanario}@telecom.ntua.gr
[4] Telecom Italia s.p.a., Italy, email: marco.marengo@telecomitalia.it

2 A Library based Architecture Model

This section will present our hybrid architecture for preference learning and context refinement. It is part of the software architecture of the EU project Persist. It only sketches the relevant part of the overall architecture that can be found in [6].

The architecture section presented in this paper is shown in Figure 1. It represents a crucial part for realising self-improvement in this system. In particular it interacts with the Context Management subsystem and the Preference Management system, the access points to external usage. while itself is only called from within the Persist architecture. It is designed by the following guidelines:

- *Transparency*: The system user does not have to have any knowledge about the existence of this system and should not be aware either when it runs in the background. Response times have to be fast and computational extensive processes should run at times with low processor occupancy. All queries have to be issued to a single access point.
- *Modularity*: Deployment on mobile devices has to deal with limited resources. Therefore some processes will have to be performed in the back end of the Persist system, while other modules are deployed on all mobile devices. Therefore our architecture has to allow for a splitting of functionality and has to be able to run even when some parts are missing locally or are not reachable due to a lack of network connectivity.
- *Pluggability*: There are many different approaches for machine learning and refinement of context information. This architecture shall not reduce the performance capabilities of the Persist system by limiting to a certain number of algorithms for self improvement. Even during runtime it has to be able to include new algorithms that can immediately be used further on. This is allowed by a common interface for all methods within one library.

As you can see in Figure 1, the presented part consists of two interrelated library systems. On the left, the *Preference Learning subsystem* is strongly related with the Preference Management system as it learns and updates preference rules for this system. It is triggered by the Persist Eventing system upon user actions that are stored in the context database. These actions are retrieved by the Preference Learning subsystem and passed on to the methods in the learning library which use it either directly or after a certain time or amount of gathered actions. The resulting preference rules are stored again in the context database, but may also serve as input to the context refinement subsystem that are specialised on evaluating them.

Hence the right branch of the picture is dedicated to evaluation algorithms. Context refinement can be any method that accesses available context information from the Context Management system and refines/enriches it. Therefore a number of different algorithms are inside the context refinement library. These can work either on demand or run continuously, subscribing themselves for changes of their input context information. While on-demand algorithms are triggered by the Context Management system, if requested information is not available, continuous refinement can be caused either also by context consumers that have to be notified upon any change or by the nature of the approach.Continuous refinement would be in theory desirable for any client, but practice shows that it is much more resource consuming and often needless. So a suitable trade off has to be found.

3 Context Refinement algorithms

Context information gathered by sensors is stored and managed in the Context Database, see 2. In many cases however this information is not easily processable for context consumers, as it is too fine grained, not reliable enough, not meaningful or also just not the information that is searched for. That is why a context aware architecture must provide a number of approaches to refine the raw context information and provide the consumers with the right, reliable and precise information.

3.1 Bayesian Filters to refine Location Accuracy

Location estimation of a moving entity such as a person can be viewed as a static or dynamic estimation process, depending on how often a location estimate is needed, or can be efficiently generated. It can be shown that the dynamic Bayesian estimator is the most accurate form of estimation for this problem and very often algorithms from the family of particle filters are applied, [9]. The reason for the superiority of dynamic filtering over repeated one-shot estimation is that the former makes use of the temporal correlations of the location in the form of a process or ”motion” model. When location updates are only needed rarely then the cost of continual estimation - both in terms of battery power and any other resources such as communications bandwidth - must be weighed against the improved accuracy. Also, dynamic estimation is far moire suited to estimate the full pose of a person, i.e. also the direction the person is facing.

In sequential Bayesian filtering, sensors provide some information related to location or motion. Such sensors can encompass GPS receivers, RFID readers, WLAN sensors, UWB systems, barometric altimeters, magnetometers, inertial sensors and many more. It is important to recognize that these sensors do not necessarily need to give location information directly; a magnetic compass gives information about the orientation of a person which still allows positioning itself to be improved in a dynamic scenario [25]. Furthermore, map information can be used to improve positioning, especially when using inertial sensors. Hence ”refinement of location accuracy” is an informal but frequently used term for the process of combining any sensor information with the goal of accurate location information.

Dynamic localisation is especially important in the indoor environment since many sensors that are usually very accurate no longer perform very well, such as GPS receivers or mobile radio positioning. dedicated infrastructure such as UWB systems may be available but suffer from coverage problems. In these cases a motion model in a dynamic estimator that takes maps into account can greatly improve performance.

Finally, an important advantage of any Bayesian filter is that it inherently provides uncertainty information about the location and pose, which can be passed to other processing layers, such as probabilistic context inference.

3.2 Clustering to discover recurring locations

Current terminal devices can readily access location related information whereas other sources of contextual information are harder to gather and process, and many techniques may be applied to enhance the accuracy of such data. Nevertheless, coordinates identify a single point in space, while people are most accustomed to reason in terms of ”places”, regions of space. Location tagging is a method for inferring *”places”* from a set of positions. ”Places” are regions of space that carry some meaning to the user and to which the user can potentially attach some (meaningful) semantics. Examples of places include home, work, pub and airport.

The location tagging component is part of the reasoning framework and offers both places detection functionalities and a user inter-

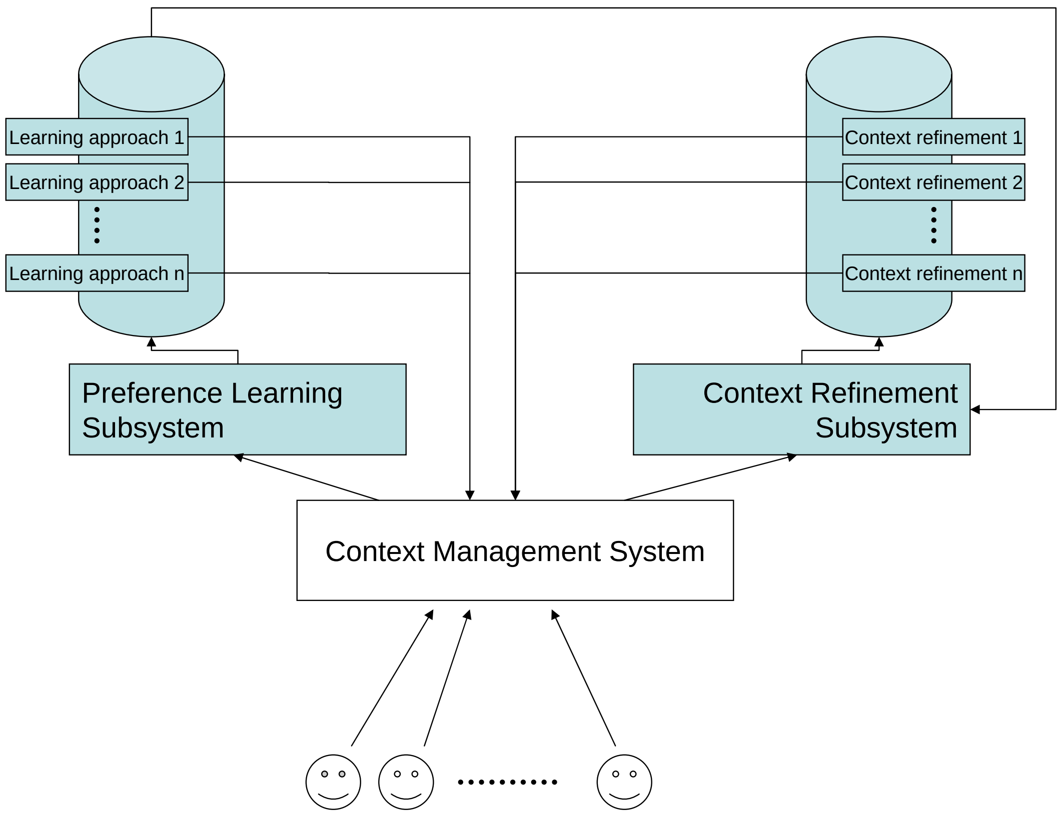


Figure 1. Simplified Architecture model for Preference Learning and Context Refinement libraries

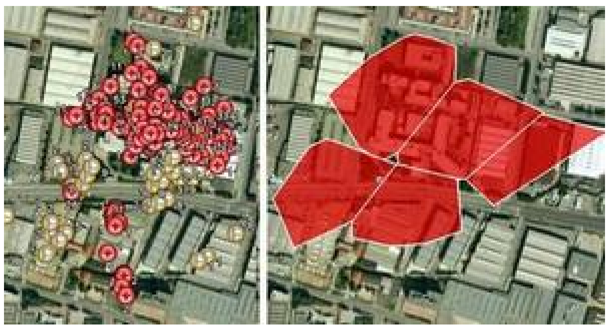


Figure 2. From *positions* to *places*: how a set of coordinates (left) is transformed into places (right)

face for managing the user location model. Once discovered, places are defined by a region of space, characterized by a non null area, a description and a place-type which is chosen from a simple place-types taxonomy [10].

The learning process involves the following steps:

- *clustering*: user position history is partitioned in several clusters of data, which represent the most recurring locations. Our implementation uses the Shared Nearest Neighbour (SNN) [15] algorithm to discover clusters with different densities.
- *place definition*: once discovered, clusters are just sets of points, making it difficult to store and to use them. A convex hull algorithm [3] is therefore applied to each cluster in order to discover the smallest polygon that includes all the cluster's points. Only the vertices of each cluster are stored.
- *user validation*: before being useful, places need to be validated by the user. Users are able to manipulate their places and eventually add new ones or delete unwanted ones. Assigning place-types (e.g. *"office"*, *"restaurant"*, …) to a user's places enables exciting scenarios, as other reasoning components could make use of such information to offer place-aware services or advertisements.

The clustering step is time-consuming and should therefore be carefully scheduled. It is also recommended to follow an incremental approach, in order to reduce the amount of processed data.

3.3 History based context prediction

The efficient collection, maintenance, and processing of *History of Context (HoC)* [13] is critical for Personal Smart Spaces, as it allows for inference of future context and current context information that is no longer available. The HoC management framework in Persist, also caters for elements of the user behaviour, thus providing input to support the user preferences' learning process, as well as the inference of user intent regarding the usage of pervasive services. The designed HoC management facilities are also able to support the extraction of periodic patterns regarding context data combinations and use these patterns in order to deliver successful context predictions.

The proposed approach regarding the HoC lifecycle consist of 4 generic phases which are schematically depicted in figure 3. Initially, context sources feed the measured context data in the main context DB synchronously (Step 0) or asynchronously (Step 1). Subsequently, this information is cached in the HoC DB maintained by a resource-rich system (e.g. a user's home PC) where it is processed and scored(Step 2). Based on this input and enforcing time-dependent attenuation to all past context values, context prediction

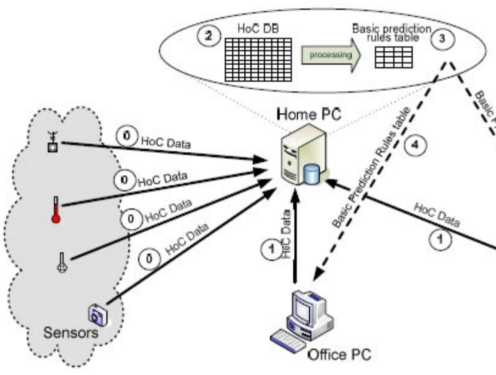
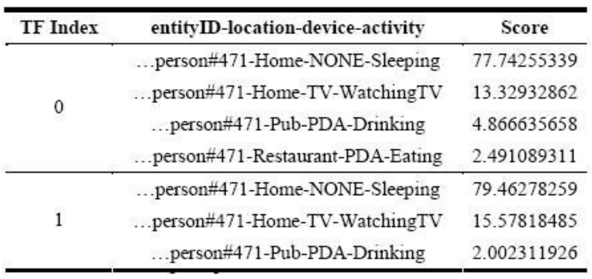


Figure 3. History of Context lifecycle

rules (called Basic Prediction Rules) are generated (Step 3) and disseminated to PSS systems supporting HoC based context prediction or estimation (Step 4).

In more details the PSS user or administrator may identify the context data types that need to be monitored. The values of HoC data recorded can be quantifiable, such as the GPS coordinates expressing location, or can be represented by abstract tags, as is the case for activity (e.g. busy, free, sleeping), device used (PDA, laptop, desktop), semantic location (home, work, restaurant), etc. These data along with the respective timestamps are initially cached at the devices connected with the context sources and when possible forwarded to a resource rich system that maintains the HoC database. The database is structured in 48 half hour time frames. Context data combinations are assigned to time frames depending on the start and end time these were observed. Subsequently, the entries assigned to each time frame are "scored". Thus, instead of maintaining all occurrences of each context data combination, as well as their start/end time and date, each such combination is assigned to a single entry in the corresponding time frame in the HoC database, along with a single score that is decreased as time passes by and increased every time the exact same context situation is observed.An example segment of the main HoC DB table is presented in figure 4.



TF Index	entityID-location-device-activity	Score
0	…person#471-Home-NONE-Sleeping	77.74255339
	…person#471-Home-TV-WatchingTV	13.32932862
	…person#471-Pub-PDA-Drinking	4.866635658
	…person#471-Restaurant-PDA-Eating	2.491089311
1	…person#471-Home-NONE-Sleeping	79.46278259
	…person#471-Home-TV-WatchingTV	15.57818485
	…person#471-Pub-PDA-Drinking	2.002311926

Figure 4. Example of part of the main table of the HoC database

This approach aims to reduce the storage resources required and achieve significant context summarization. The score values are calculated on a daily basis. Depending on the score values and the selected rule generation model, context prediction rules are generated. As the size of the prediction rules is minimal, it is possible to forward them to all the remote devices enabled for context prediction or inference, irrespective of the storage and processing resources they have available.

3.4 Bayesian High-level Context Inference

The last sections 3.1 and 3.2 showed how to add meaning and precision to context information by refining sensor readings, respectively deducing missing context information from past patterns. For many applications however context information is necessary that cannot come from sensors – as there *is* no appropriate sensor: for so called high-level context information. Most prominent among those are *situation*, *activity*, *mood*, *manoeuvre*, *comfort*, *danger*, *stress* and so on.

If an application or a preference depends on such high-level information ("IF user is going home..." or "IF user is not stressed...") this information has to be inferred from available context source information by appropriate rules.

A very promising approach for this inference is the use of *Bayesian Belief Networks (BN)* as the rule representation. BNs consist of random variables (represented as nodes) with defined states which are interconnected by directed edges representing causal dependencies and contain a conditional probability distribution (CPD) [19, 11]. The random variables can represent context attributes, their states model the possible values of the context attribute, as can be seen in Fig. 5. If sensor readings are available, the respective at-

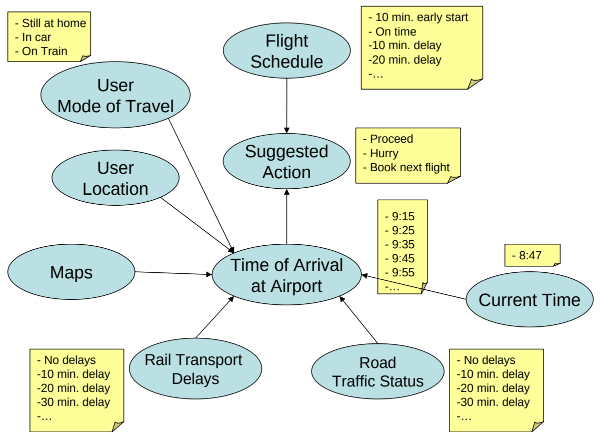


Figure 5. An example Bayesian Network

tributes are assigned *evidence*, which influences the probabilities of related attributes – calculating the most probable state of a target context attribute given all available evidence can be used as context inference. This approach combines several advantages [2]:

- A Bayesian network readily handles situations where some data entries are missing or uncertain.
- As it has both a causal and probabilistic semantic, it is an ideal representation for combining prior knowledge and data.
- Bayesian statistical methods in conjunction with Bayesian networks avoid the overfitting of data and do not need to separate data into training and testing sets. They are also able to incorporate smoothly the addition of new data as it becomes available.

One drawback of this kind of inference is its time complexity. Cooper [4] identified the problem as NP hard in the number of random variables, which holds even for singly connected BNs (see [26]). With a large number of interrelated context attributes, context inference could therefore become unfeasible in practice on mobile, resource limited devices. To overcome or reduce this problem, either the number of nodes or the number of states per node or the number of edges have to be limited. Our approach [7] is using *Bayeslets* to reduce computation time by reducing all three parameters.

Furthermore it offers us the chance to personalise inference rules, which is of particular importance for high-level context, as it depends very much on the user's individual perception. Inference rules based on Bayeslets can even adapt during run-time to changed behaviour of its owner. If the Context Management System triggers inference on a particular context attribute, it chooses the Bayeslet of the target user that has the desired context attribute as output node, connects it to the related Bayeslets to build a complete, relevant inference rule, evaluates it and returns the inferred state of the requested context attribute.

3.5 Proximity Estimation with a Diffusion Model

Consider the following short use-case: A person is visiting a company for a meeting and would like to print a document. She is not very familiar with the layout of the premises and there are certain restrictions as to which printers are allowed to accept and process requests from visitors. Assuming that the visitor is associated with the local network of the premises a context dependent search for suitable printer services can take place. This search can encompass not only administrative constraints, and those based on the service usage (e.g. for a colour printer), but also on location and proximity metrics. In particular, our visitor should usually be guided to the nearest suitable printer, where nearest refers to reachability in pedestrian terms.

Simply applying a straightforward Euclidean metric in 3D space will usually fail, since non-linearities, such as walls, in the true practical proximity are not considered. A true proximity metric should be proportional to something like the time taken to reach the destination or the distance travelled.

In [1] a mobility model pedestrians was introduced that applies a gas diffusion algorithm to estimate a path between two points in a given floor plan layout. The algorithm works by emitting a virtual "gas" source from the destination that propagates through the open portions of a building layout and becomes absorbed by walls. Any point in the building experiences a certain concentration of the "gas" that is proportional to the distance to the emitter (destination, e.g. a particular printer). So, physical facilities like printers can be easily rated by their diffusion matrix by comparing their relative "gas" densities at the starting point (e.g. where the user is located). Computation of the gas source is straightforward ([14]) and the results may be cached in a database.

Once a user has selected a suitable destination, using the diffusion model and other criteria such as suitability, the model can also be used in a very simple way to generate routing information to guide the user to the destination. All that needs to be done is to follow the steepest gradient of "gas" concentration to the destination.

In order to impose restrictions in motion for certain people (e.g. visitors may not be allowed access to certain areas) then these restrictions can be represented in their own layout. People with disabilities may also impose restrictions on certain features of a building, such as stairs.

4 Preference learning algorithms

Creating and maintaining a set of user preferences is a continuous and important task as the quality of the personalization provided depends on the quality and completeness of the user's preference set. When a new user enters a pervasive environment or an existing user starts to use a new service, network or device existing preferences may need to be refined or new preferences added. Performing this task in an entirely manual fashion is undesirable for two reasons. Firstly, the possible number of personalizable attributes (e.g. service customization parameters, service selection criteria, etc.) is potentially very large with each possibly requiring a preference. Manually creating and maintaining such a large set requires time and effort, detracting from the benefits that personalization aims to provide. Secondly, some user behaviours may be so implicit that users may not identify them as explicit preferences. For example, a user may not be aware that he/she always selects a service within a certain price range at a certain time of day. Even if the user was aware of such a behaviour it may be difficult for them to represent it in the appropriate format. Therefore it is essential that mechanisms, such as monitoring and learning are in place to aid the user with preference creation and maintenance, providing them with a more accurate and complete preference set and hence better overall personalization.

4.1 IF-THEN-ELSE Rule Learning

User preferences can be represented in many ways however, it is beneficial to provide a human-understandable format to allow the user some control over their preference set and hence personalisation. One adopted approach is to express user preferences as IF-THEN-ELSE rules to allow for human-readability. One way to achieve this is to first learn a decision tree, then translate the tree into an equivalent set of rules. The IF-THEN-ELSE rule learning algorithm [17] implements this strategy. It is an adaptation of Quinlan's C4.5 batch learning algorithm [20] which aims to split outputs into distinctive groups using a minimal number of attributes from the input dataset. As with C4.5 Gain Ratio is used to combat problems arising from attributes with multiple values.

Input to the algorithm is a list of `userAction` objects each containing some action the user has performed (e.g. `video = paused`) and a context snapshot of the user's context situation when they performed the action (e.g. `location = home, call = incoming`). Each userAction also contains further meta-data such as the service type and service instance from which the action originated. The list of userActions is stored in the Context Management System in the User History Database. Learned output is a set of decision trees (one for each learned preference) with context attributes as condition nodes and actions as leaf nodes. Each tree is then mapped to IF-THEN-ELSE rules for human-readability.

The dynamic nature of pervasive environments often challenges the batch approach to preference learning. For example, if the user changes his behaviour this cannot be captured until the next batch learning process. Even then, new behaviour may be over-written by more established (but perhaps distant) behaviours. Several projects [24] implement 'quick response' mechanisms that immediately accommodate new behaviours into the user's preference set however this approach can suffer from catastrophic forgetting which is undesirable if the behaviour change was only temporary.

To overcome these challenges a novel dual store approach is adopted for the User History Database. This acts as a *short-term store (STS)* and *long-term store (LTS)*. Figure 6 below illustrates the content of each partition.

The STS only holds userActions which have happened since the most recent batch learning process at time t_i (i.e. this is where new userActions are initially stored). The data held in the STS is the input to each batch learning process. After the next batch learning process occurs at time t_j the STS will be retrieved, learned upon, copied to the LTS and cleared. The LTS stores all user actions which have happened since the user started to ever use the system at time t_0

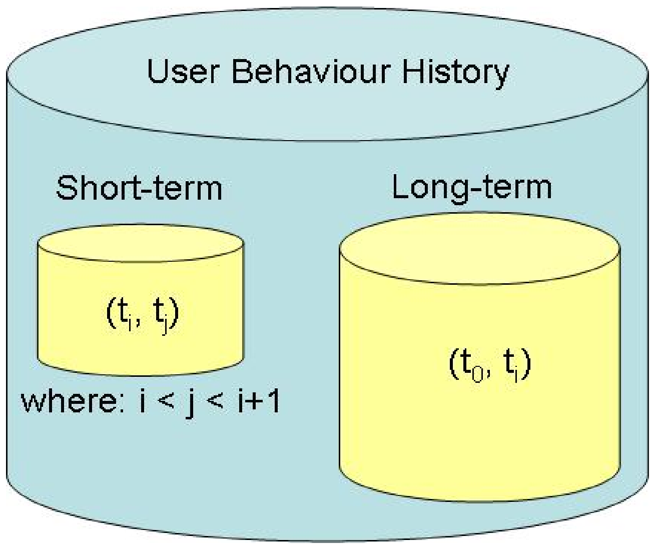


Figure 6. Dual User History

until the most recent batch learning process at time t_i. It acts as a backup containing the entire back catalogue of `userActions`. The LTS is used to resolve any conflicts that may arise when merging new learned information with existing preferences. The dual store approach allows recent changes in user behaviour to be identified quickly while also providing support where behaviour changes are temporary or where conflicts occur between new learned information and existing preferences.

4.2 Online, Incremental Preference Learning

As a compliment to the batch IF-THEN-ELSE learning algorithm, Persist will also employ on-line, incremental preference learning. This will be implemented as a hetero-associative neural network that will associate current context to user actions. The topology of the neural network allows for internal network knowledge to be translated into IF-THEN-ELSE rules for user understanding. This approach will be described in detail in a forth-coming paper.

4.3 Bayesian Learning of Behaviour and Inference Rules

As described in section 3.4 Bayesian Networks (BN) can be used to represent the probabilistic relationships between random variables that represent various aspects of context information and user actions, from sensor readings to the high-level context attributes such as a person's current activity. Such a network structure is defined completely by the set of random variables (nodes) and directed arcs between random variables that can often be interpreted as causality (in the direction of the arc) [19]. The conditional probability distributions of each node - conditioned on each possible instantiation of all the parent nodes - defines the network quantitatively (parameters of the network). As described earlier, given the network structure and parameters inference is the process of computing the probability of a set of hidden random variables given observations of other random variables. This section will briefly address the approaches used to obtain both network structure and parameters using a combination of expert knowledge and previous observations of the random variables by *learning*.

Applied to learning of behaviour rules, some RVs (preference outcome nodes) can either be interpreted as active behavioural choices by the user, or the BN can be extended to a utility network with decision and utility nodes, where the decision nodes are the outcomes. Applying such learnt rules during the use of the system is thus the process of evaluating known context information to infer the user's most probable choices or actions based on learning the BN (which itself can be interpreted as a probabilistic behavioural rule).

4.3.1 Determination of the scope of the network

Context information pertaining to a user of Smart Electronic Spaces may in general encompass a very large range of human activities, sensor readings, information such as calendars, as well as such data relating to other people; all represented as random variables (RVs) in a very large BN. Obviously it is impossible to include all sets of such RVs in a representation used in inference, as the resulting BN would be too large. Take, for instance, the case where we are attempting to infer the current activity of a person in an office environment. Her activity will be part of a BN that includes not just her location and pose but also information such as the devices she is using, her calendar schedule, her heart rate and other bio-sensor readings, her past activity, the time of day, proximity to other people, the status of projects she is working on, as well as much more; additionally, such a BN will encompass similar aspects of other people's context, such as that of her colleagues. For practical relevance it will be necessary to impose boundaries on which RVs will be used to represent the information ”around” her activity. For this purpose we have previously introduced the concept of ”Bayeslets” that are small BNs that may be linked to form larger BNs [7]. It is fair to assume that human expert knowledge will be used to determine the RVs that make up such a Bayeslet, and which smaller Bayeslets can be assembled to represent a problem domain such as activity.

4.3.2 Learning static BNs

We have thus far restricted our approach to encompass only static BNs, rather than dynamic BNs [18], and to RVs with discrete variables. To incorporate at least some temporal aspects, we do allow RVs that encode previous outcomes of RVs, such as a user's activity or location a certain time ago or in the future. Assuming that a human expert has determined the scope of RVs in a learning process we can resort to established BN learning techniques for:

1. Learning with complete data.
2. Learning with missing data.
3. Learning with missing data and also hidden variables.

Learning with complete data sets of all involved RVs is easiest and follows the approach described in [5]. Usually some form of greedy hill climbing is used to rate candidate networks and we take the best scoring NW ignoring network structure priors. Naturally, we will impose knowledge of:

- Causality between RVs (such as sensors readings being caused by a physical process); i.e. the presence and direction of arcs.
- Arcs that we know or assume are missing between RVs, thus imposing more independence.
- Complete sub-units of the network that are assumed to be known.

For missing data we will employ algorithms from the class of (structural) expectation maximization (EM) introduced in [8].

5 Conclusion and Outlook

This paper reflects the work being undertaken in task 4.4 of the EU project Persist that is investigating learning and reasoning algorithms for personal, self-improving smart spaces. Learning preferences and reasoning rules is a core part for the self-improvement targeted in Persist, inference and in general all context refinement procedures are the methods necessary to give expressivity to preferences and to evaluate them.

We decided to adopt a library based approach with a single access point and common interchange formats, that gives access to a collection of different algorithms for learning and context refinement. This library is not limited however to the presented methods, but open and extendible to new approaches allowing for the same interface.

This paper presents a number of context refinement approaches that apply on different levels. Fusing raw sensor data for more precise information in section 3.1, then the enrichment of this precise absolute location with semantics like in section 3.2. Such meaningful information is basis for the approaches for context prediction in section 3.3 and high-level context inference in section 3.4. A particular high-level context information, proximity, can be evaluated with the approach shown in section 3.5.

We have presented in this work furthermore three different algorithms for learning, each offering some variants. Differences between them are the execution procedure (batch vs. incremental learning), weighting of long past and recent input information and also the purpose. The approaches in sections 4.1 and 4.2 are identifying behavioural patterns for preferences whose conditions are permanently monitored further on, while the probabilistic approach in 4.3 can provide Bayesian context inference rules (suitable for the approach in section 3.4) as well. These approaches are implemented in the project Persist that will publish its source code at an Open Source portal as a basis and guideline for fellow researchers. Still there are a number of questions to be answered.

- A particular challenging one will be the orchestration of these approaches. A component that delegates learning or context refinement tasks to the most appropriate approaches is not considered so far.
- Neither has there been undertaken detailed work about conflict resolution mechanisms yet. In case of contradicting results from different methods this will be necessary. This functionality could be based on confidence of the methods' outcomes that is evaluated by a probabilistic component, e.g. in a static BN.
- Resource and computation time issues of the overall system have not been dealt with so far.
- Privacy issues are not treated in this paper, but are part of the Persist architecture. The outcomes of this research is published in a separate paper.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under grant agreement no. 215098 of the *Persist* (PERsonal Self-Improving SmarT spaces) Collaborative Project

REFERENCES

[1] A. Angermann, J. Kammann, and B. Lami, 'A new mobilty model based on maps', in *The 58th IEEE Semiannual Vehicular Technology Conference (VTC Fall 2003)*, (2003). Date=2003-10-06 - 2003-10-09;.

[2] M. Angermann, P. Robertson, and T. Strang, 'Issues and requirements for bayesian approaches in context aware systems', in *LoCA 2005*, ed., C. Strang, T.; Linnhoff-Popien, (05 2005). event_dates=2005-05-12;.

[3] H. Huhdanpaa C. Barber, D. Dobkin, 'The quickhull algorithm for convex hulls', *ACM Transaction on Mathematical Software*, **22**(4), 469–483, (1996).

[4] G. F. Cooper, 'Probabilistic inference using belief networks is NP-hard', Technical Report KSL-87-27, Medical Computer Science Group, Knowledge Systems Laboratory, Stanford University, Stanford, CA, (May 1990).

[5] G. F. Cooper and E. Herskovits, 'A bayesian method for the induction of probabilistic networks from data', *Machine Learning*, **09**(4), 309–347, (October 1992).

[6] I. Roussaki et al., 'Initial architecture design', Technical report, PERSIST, FP7-ICT-2007-1, (November 2008).

[7] K. Frank, M. Röckl, and P. Robertson, 'The bayeslet concept for modular context inference', in *The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies (UBICOMM08)*, eds., J.L. Mauri, N. Cardona, K. Chen, M. Popescu, and A. Doci, pp. 96 – 101. IEEE Computer Society Conference Publishing Services (CPS), (05 2008). event_dates=[2008-09-29 - 2008-10-04];.

[8] Nir Friedman, 'The bayesian structural em algorithm', in *In UAI*, pp. 129–138. Morgan Kaufmann, (1998).

[9] F. Gustafsson, F. Gunnarsson, N. Bergman, and Forssell U. et al., 'Particle filters for positioning, navigation and tracking', *IEEE Transactions on Signal Processing*, (2002). vol. 50, No. 2, 2002.

[10] H. Tschofenig H. Schulzrinne. Rfc 4589: Location types registry. IETF.

[11] D. Heckerman, 'A tutorial on learning with bayesian networks', Technical report, Learning in Graphical Models, (1995).

[12] K. Henricksen and J. Indulska. Personalising contextaware applications, 2005. K. Henricksen and J. Indulska, Personalising ContextAware Applications, in OTM Workshop on ContextAware Mobile Systems, vol. 3762, Lecture Notes in Computer Science: Springer-Verlag, 2005, pp. 122-131.

[13] Nikos Kalatzis, Ioanna Roussaki, Nicolas Liampotis, Maria Strimpakou, and Carsten Pils, 'User-centric inference based on history of context data in pervasive environments', in *SIPE '08: Proceedings of the 3rd international workshop on Services integration in pervasive environments*, pp. 25–30, New York, NY, USA, (2008). ACM.

[14] Mohammed Khider, Susanna Kaiser, Patrick Robertson, and Michael Angermann, 'A novel movement model for pedestrians suitable for personal navigation', in *ION NTM 2008*, pp. 819 – 827. The Institute of Navigation, (01 2008). Date=2008-01-28 - 2008-01-30;.

[15] V. Kumar L. Ertöz, M. Steinbach, 'Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data', *SIAM International Conference on Data Mining (SDM '03)*, (2003).

[16] Seng Loke, *Context-Aware Pervasive Systems: Architectures for a New Breed of Applications*, Auerbach Publications, 2006.

[17] S. McBurney, E. Papadopoulou, N. Taylor, and H. Williams, 'Adapting pervasive environments through machine learning and dynamic personalization', in *Proc. of the 2008 Conference on Intelligent Pervasive Computing*, pp. 395–402, (2008).

[18] Kevin P. Murphy, *Dynamic bayesian networks : representation, inference and learning*, Ph.D. dissertation, 2002.

[19] Judea Pearl, *Causality: Models, Reasoning, and Inference*, Cambridge University Press, 2000.

[20] John Ross Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kauffman, 1993.

[21] D. Saha and A. Mukherjee, 'Pervasive computing: a paradigm for the 21st century', *Computer*, **36**(3), 25–31, (2003).

[22] M. Satyanarayanan, 'Pervasive computing: Vision and challenges', *IEEE Personal Communications*, **8**, 10–17, (2001).

[23] R. Singh, P. Bhargava, and S. Kain, 'State of the art smart spaces: application models and software infrastructure', *Ubiquity*, **7**(37), 2–9, (2006).

[24] SPICE. Service platform for innovative communication environment. http://www.ist-spice.org/, 2008.

[25] K. Wendlandt, M. Khider, M. Angermann, and P. Robertson, 'Continuous location and direction estimation with multiple sensors using particle filtering', in *MFI 2006*, ed., IEEE. IEEE Verlag, (09 2006). Date=2006-09-04 - 2006-09-06;.

[26] D. Wu and C. Butz, 'On the complexity of probabilistic inference in singly connected bayesian networks', in *RSFDGrC(1)*, pp. 581–590, (2005).