Capturing Regular Human Activity through a Learning Context Memory

Philipp H. Mohr and **Nick Ryan**
Computing Laboratory
University of Kent, Canterbury, UK
{phm4,nsr}@kent.ac.uk

Jon Timmis
Department of Computer Science and
Department of Electronics
University of York, York, UK
jtimmis@cs.york.ac.uk

Abstract

A learning context memory consisting of two main parts is presented. The first part performs lossy data compression, keeping the amount of stored data at a minimum by combining similar context attributes — the compression rate for the presented GPS data is 150:1 on average. The resulting data is stored in an appropriate data structure highlighting the level of compression. Elements with a high level of compression are used in the second part to form the start and end points of episodes capturing common activity consisting of consecutive events. The context memory is used to investigate how little context data can be stored containing still enough information to capture regular human activity.

Introduction

Context-aware or, perhaps more correctly, context-sensitive systems are an important aspect of Ubiquitous computing, where Ubiquitous computing, Pervasive computing, and Ambient Intelligence all refer in some way to addressing similar goals based on Mark Weiser's vision that computers should be perfectly integrated in all parts of our lives. Weiser believed that devices should remain largely invisible and the user would interact with them often without realising (Weiser 1991); if there are differences of emphasis within this community, they mostly lie in details such as the extent of invisibility. The definition of context we use is by Abowd and Dey:

> *Context* is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (Dey & Abowd 1999).

By context-awareness we understand the capability of a system to recognise changes in its environment and adapt its behaviour accordingly, for example movement sensors which turn the light on when someone enters the room. However, there is a very large step between collecting values that describe easily measured aspects of the environment, such as location or ambient temperature, and determining a user's current activity and resource needs. In order to perform this step one needs to solve the problem of how to capture information about the environment and interpret it in terms that accurately reflect human perception of tasks and needs. Additionally, environmental data is potentially of very high dimensionality, raising another challenge in terms of complexity and data storage, especially as such systems need to be made available on small, portable, resource-constrained devices, e.g. mobile phones or PDAs.

This paper presents an online learning context memory, named Context-Aware Memory Structure (CAMS). Off the shelf sensors, e.g. Wifi, Bluetooth, GPS, and Time, are used by CAMS to acquire low level context of its user, which is processed to construct a context memory reflecting that user's regular activity. Context memory is a mechanism for retaining and recalling interesting and relevant past experience, consult (Mohr, Timmis, & Ryan 2005) for a more detailed description. CAMS can provide context-aware systems with high level context information of current activities and historic context, which helps to minimise partial loss of sensor data.

Context-Aware Memory Structure

CAMS can be seen as middleware sitting between sensors and context-aware applications, requiring it to bridge the gap between collecting values that describe easily measured aspects of the environment provided by sensors, such as location or ambient temperature, and determining a user's current activity. CAMS bridges this gap by learning its user's regular activity enabling it to recognise familiar situations; as human activity is inherently complex the learning process is non-trivial. In order to keep the complexity at a manageable level, the following design decisions have been made:

- *Online learning:* Learning rate can dynamically be changed to cope with context shifts. No temporary storage for training purposes.
- *Lossy data compression:* Storing all past context is intractable, therefore the amount of stored context data needs to be kept to a minimum; by removing duplicate or very closely related context data and 'forgetting' potentially obsolete context data.
- *Data representation:* The context-attributes are stored more or less in their original format, allowing historic context to be transparently used alongside current context

and as a user specific memory is often created over an extensive period of time it is valuable and therefore needs to be readable by new or improved algorithms — all these are difficult to achieve by Neural Networks where the information is embedded in the weights.

- *Layered design:* The system is designed in a layered fashion, keeping the complexity of each layer at a manageable level. In addition it allows the interface of the top level to be easily adaptable to the requirements of a wide range of context-aware applications.
- *Episodic memory:* Relationships between consecutive context attributes need to be highlighted, as they play an important part in capturing human activity made up of consecutive events.
- *Ubiquitous environment:* The system needs to be made available on small, portable, resource-constrained devices and it needs to work in a range of networking environments with the real possibility that it must spend a proportion of time working with no connectivity.
- *Every day environments:* In order for the system to diffuse into every day environments, the user should very rarely be required to be an active part of the system. This is achieved by making use of unsupervised learning.

The input to CAMS is sensor data, which is provided in the form of an attribute vector,

```
< Location.GPS = (67.2983, 4.06965),
  Wlan = 0A:40:C3:B1:01:43 "WlanHome",
  Time = 19:30 , DayOfWeek = 2 >
```

which contains attributes (which can appear in an arbitrary order) along with the class they belong to.

The main components of CAMS are the "Snapshot Memory" and "Episodic Memory". These are where the context processing and context storage takes place. A detailed description of both components is given below.

Snapshot memory

The snapshot memory processes and stores context attributes from context input vectors. Attributes are stored in Artificial Recognition Balls [1] (ARBs), which describe a certain region around the context attribute —in the case of 2D GPS co-ordinates they are a circle— and enable CAMS to perform data compression by eliminating the need for repetition. For example, a place of interest can be represented by a single ARB instead of all individual GPS co-ordinates which fall within this region; every ARB has a resource level R associated with it, being an indicator for how frequently it recognises context attributes. The algorithm used in CAMS is based on the principles of unsupervised and reinforcement learning. Unsupervised learning allows us to construct a system which can cluster input data without any prior knowledge about the structure of every class. Reinforcement learning requires feedback from a trainer. However, an explicit trainer is not desirable in most context-aware systems, therefore an ARB receives positive feedback (stimulation) when context attributes fall within a certain distance from the centre, resulting in an increase in its resource level. Negative feedback is introduced by the notion of 'forgetting', which gradually decays all resource levels. For example, locations a user visits often have their resource level reduced by a decay, but every visit stimulates it again, which enables these locations to remain in memory. Obsolete data does not undergo any new stimulation and so is eventually removed from the system.

For context classes with continues attribute values, for example GPS co-ordinates, the resource level affects the recognition distance of an ARB, the distance is initially large but decreases when the resource level rises, allowing areas frequently visited by the user to be represented in CAMS with a finer granularity. If these areas become less frequently visited the decay decreases the granularity again. For context classes with discrete attribute values, for example a Mac-address of a Wlan access point, the recognition distance is not affected by the resource level — it is either 0 or 1. In both cases the resource level reflects how frequently an ARB is recognising context attributes. The stimulation and decay mechanisms enable the smoothing of context data and allow for a gradual context shift — a sudden shift is not desirable, as it puts too much emphasis on one off or irregular events. In addition, the decay mechanism controls the minimum amount of time an attribute remains in the system.

All ARBs are stored in an n-dimensional network structure, where each dimension represents a different class of attributes, for example `Time` indicates that `19:30` belongs to the dimension 'time'. ARBs from different dimensions which appear in the same context are connected by cross-dimensional-links, for the example attribute vector a cross-dimensional-link would be created between the $Location.GPS$ and the $Wlan$ attribute, as well as between all the other attributes. These links exploit the information present in the relationship between context attributes. Furthermore, they have a resource level L associated with them which reflects the likelihood that these attributes occur in the same context. The resource level L undergoes stimulation and decay mechanisms similar to those of the resource level R.

We now provide a more formal description of the snapshot memory, using the formalism of metric spaces and Hoare triples. All attributes in an incoming attribute vector are stored in their corresponding $Class$ set within the snapshot memory. Every class has a metric space associated, a set of ARBs, a stimulation function, and a decay function. A metric space is a 2-tuple (X, d), where X is a set containing all possible elements of the space and d is a metric defined over X. An ARB combines multiple context attributes into one, by performing lossy data compression. This is done by representing all elements which fall within a certain region around the centre of the ARB, eliminating the need for repetition. An ARB is represented by an n-tuple, and all ARBs belonging to a certain class are stored in the corresponding set $Class(class)$. An ARB is an ordered 4-tuple containing a centre a, a radius δ, a resource level R, and a counter $count$. The centre a and the radius δ define an open ball $B(a, \delta)$. The radius δ depends on the resource level R and

[1] Concept derived from Artificial Immune Systems (Neal 2003)

their relation can be described as: $\delta = \delta_{max} - k * R$, where δ_{max} is the maximum radius and k is a scaling constant. As the resource level R increases the radius δ decreases. The variable $count$ keeps track of how many observations have been combined in the ARB. The specific implementations of the stimulation and the decay functions depend on the class they belong to. All ARBs behave according to the following pre and post conditions, where all non dynamic ARBs omit δ:

$pre: \{ARB(a, \delta, R, count)\}$
stimulation
$post: \{ARB'(a', \delta', R', count') \wedge a' = a \wedge \delta' < \delta \wedge R' > R \wedge count' = count + 1\}$

$pre: \{ARB(a, \delta, R, count)\}$
decay
$post: \{ARB'(a', \delta', R', count') \wedge a' = a \wedge \delta' > \delta \wedge R' < R \wedge count' = count\}$

The snapshot memory can only provide context-aware systems with a time slice (snapshot) of context, it cannot provide consecutive connections between them. These connections are captured by the Episodic Memory described next.

Episodic Memory

To capture a significant part of human activity the connections between consecutive events are essential, for example 'going to work' involves leaving home and moving along some streets in order to get to work. The snapshot memory is able to capture every individual part of the journey, but not the whole journey. As the user is most likely to spend more time at home and at work than at every other point of the journey, the points "home" and "work" have a higher resource level R. Once R reaches a predefined level, the points "home" and "work" are passed from the Snapshot Memory to the Episodic Memory, which captures all individual attribute values between them. The division of the memory mechanism into Snapshot and Episodic Memory is essential for keeping the complexity of the search space at a manageable level. Without this division all attributes and connections between them would have to be stored in a directed graph in order to detect and capture meaningful consecutive events — which would result in an NP complete search problem. Instead, only the attribute vectors between ARBs with a high resource level need to be stored; after the validation of an episode (explained below) this is reduced to storing only references to ARBs recognising the attributes in these vectors. The ARBs with a high resource level R passed on from the Snapshot Memory are stored in a list. Once an ARB from this list is encountered all attribute vectors are recorded until another or the same ARB from the list is encountered. The first ARB forms the start point and the second one forms the end point of the episode; the attributes encountered in between form the $trace$. If the number of attributes encountered rises above a predefined limit before the second ARB from the list is encountered, the creation of this particular episode is stopped to avoid the creation of infinite episodes and then the episode is deleted.

Figure 1: Episode validation

The user is required to name and validate a preliminary episode only once it has been created successfully, bridging the gap between the data representation within CAMS and the real world meaning. Ideally, in order for the system to diffuse into every day environments, the user should not be required to be an active part of the system, but automatic naming, for example through a priori entered domain knowledge, is error prone. We feel the amount of work required by the user is acceptable, as only those episodes which occur frequently are presented. Figure 1 shows the screen which is presented to the user to validate an episode. In the upper text field the start and end point of an episode are presented and by clicking on the "Trace" button a map showing the complete trace is opened. In order to accept an episode a name needs to be supplied.

An episode is an ordered 3-tuple containing a $start$ ARB, an end ARB and an ordered list of all context vectors encountered between $start$ and end named $trace$. As part of our ongoing research we are investigating how to substitute the $trace$, consisting of GPS co-ordinates, between the $start$ and end ARB by a trace consisting of ARBs achieving uniformity in the data representation. Picking the ARBs recognising the GPS co-ordinates in the $trace$ is not sufficient, as the difference in stimulation between these ARBs and the $start$ and end ARBs is too large at the time the episodes are created. The ARBs in the new trace would have detection radii which are too large and therefore not capable of representing the $trace$ with enough granularity. To overcome this problem we are running experiments where the

ARBs detected by the GPS co-ordinates in the $trace$ undergo a higher stimulation than the other ARBs in CAMS.

Related Work

A range of systems exist for storing users' context and trying to predict their regular activity based on that data. Three of them are briefly described and some differences to CAMS are highlighted. In the work undertaken in (Patterson *et al.* 2003), as part of the "Activity Compass", a system able to track and predict user movement, either consisting of walking, driving by car or taking a bus is developed. This is done by combining historic context data with general commonsense knowledge of real-world constraints in order to infer high-level activity. Graph-constrained particle filters are used to integrate information from street maps, including where bus stops or car parks are located. A lot of prior knowledge is required about the area the system is being used in, as it relies on a detailed graph representation of the area; not being able to learn about new bus stops or car parks automatically, but speeding up the learning process of the user's movements and not requiring the user to provide feedback. The graph has a set of vertices and a set of directed edges, where the edges correspond to straight sections of roads and foot paths and the vertices are placed in the graph to represent either an intersection, or to accurately model a curved road. This model is not suitable to represent areas with a lot of curvy roads. The context input to the system is currently restricted to GPS signals and therefore cannot detect indoor movements. Despite sharing a number of properties, it differs significantly from CAMS, for example in CAMS prior knowledge about the area is not required and a division between frequently and less frequently visited locations is made in terms of data representation.

Mayrhofer *et al.* (Rene Mayerhofer, Harald Radi, & Alois Ferscha 2003) have developed a framework trying to predict its user's activity in order to be able to adapt to her needs in advance. It consists of four major steps, the first performs feature extraction turning raw sensor data into usable context attributes, the second classifies the input data, the third labels the set of context attributes to give them a real world meaning, and the fourth predicts the user's future context. A strong emphasis is put on the exchangeability of individual components by providing well defined interfaces. The prediction can be performed by a range of algorithms, including the Growing Neural Gas algorithm. The user is required to give the context attributes a real world meaning, but cannot provide a meaning to the predicted situations. In CAMS, on the other hand, labelling gives episodes a real-world meaning.

The work presented above cannot be regarded as context memory structures, as it is not possible to retrieve context attributes in their original format once processed. In the work by Mayrhofer *et al.* they are for instance embedded in weights of a neural network. An example of a true context memory structure is the Context Cube (Lonnie D. Harvel *et al.* 2004), it is an n-dimensional data abstraction which is an extension to SQL databases. Context attributes are stored in the appropriate class, as well as the relationships

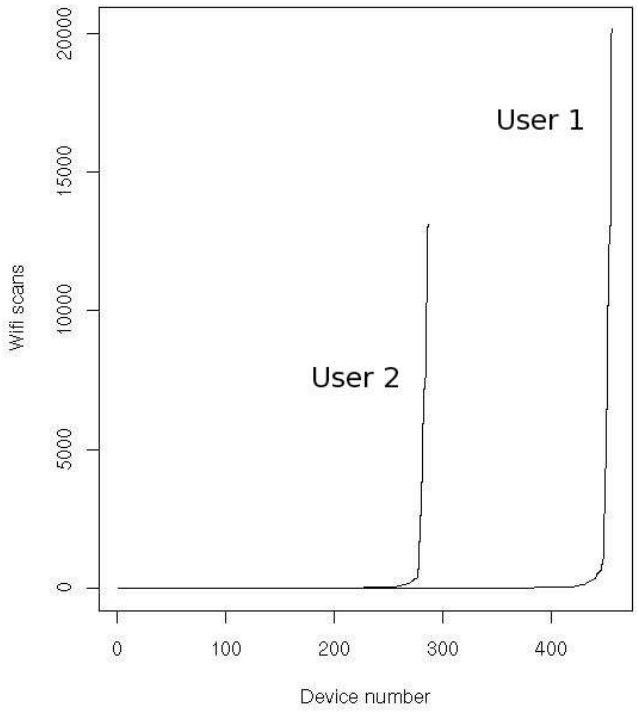

Figure 2: Wlan scans

Figure 3: All GPS points

	days	scans	Wlan	BT	GPS
User1 total	32	70711	183978 254 diff.	201486 430 diff.	23985
User1 CAMS			254	430	105
User2 total	34	112827	105800 435 diff.	23447 919 diff.	2794
User2 CAMS			435	919	37

Table 1: Scans and CAMS compression for two users

between the attributes. This is similar to the Snapshot Memory in CAMS, but no mechanisms are present for assigning resource levels to context attributes or dynamically changing the granularity of stored context attributes.

Results

Two CAMS prototypes running on HP h2210 Linux handhelds are deployed, continuously scanning their user's context. The gathered context data consists of GPS, Wlan, Bluetooth, week day, and time readings. Table 1 shows the distribution of the scanning results of two users over a period of 32 and 34 days respectively. User 1 performed 70711 scans during which 183978 Wlan, 201486 Bluetooth, and 23985 GPS readings where taken. The Wlan readings where taken from 254 different devices and the Bluetooth readings where taken from 430 different ones — these values are equivalent to the values for CAMS, as Wlan and Bluetooth are represented as discrete attributes. User 2 performed 112827 scans during which 105800 Wlan, 23447 Bluetooth, and 2794 GPS readings where taken. The Wlan readings where taken from 435 different devices and the Bluetooth readings where taken from 919 different ones. Figure 2 shows a diagram presenting the number of times a particular Wlan device was scanned. From the diagram we can see that a Wlan device was either scanned very rarely or very often, backing up the claim we made in the "Episodic Memory" subsection that certain points, such as home and work, can be distinguished from other points based on the frequency they are encountered. The diagram for the Bluetooth devices shows very similar curves and is therefore omitted.
Figures 4 and 5 show the size of the ARBs for the GPS class after 6 days and after 34 days respectively from the data recorded by user 2, for comparison figure 3 shows all GPS points recored by the same user. After 6 days 13 ARBs have been created and after 34 days 37 ARBs have been created. Out off the created ARBs after 6 days 4 ARBs and after 34 days 12 ARBs represent frequently visited areas at and around user 2's home and office with a fine granularity.

Conclusions

Context memory and user profiling systems are essential for recognising familiar situations and activities. But their creation is non trivial as context information is very numerous and human activity is inherently complex. In order to

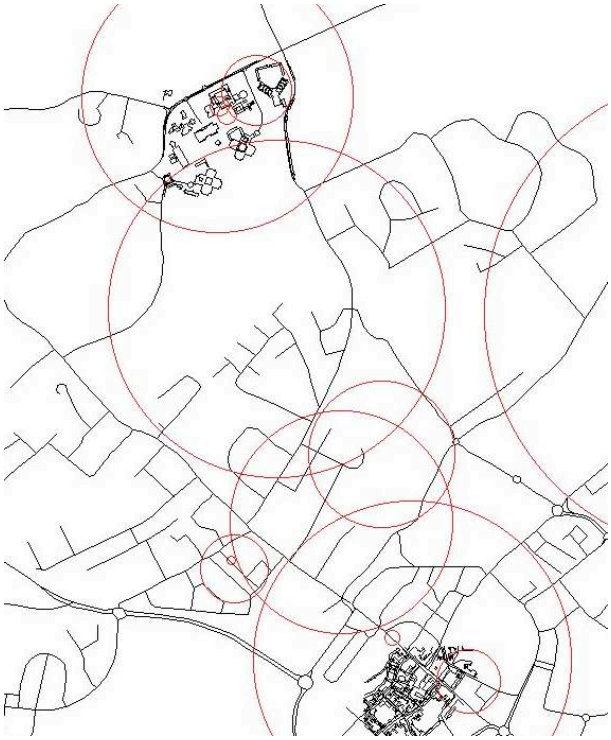

Figure 4: CAMS after 6 days

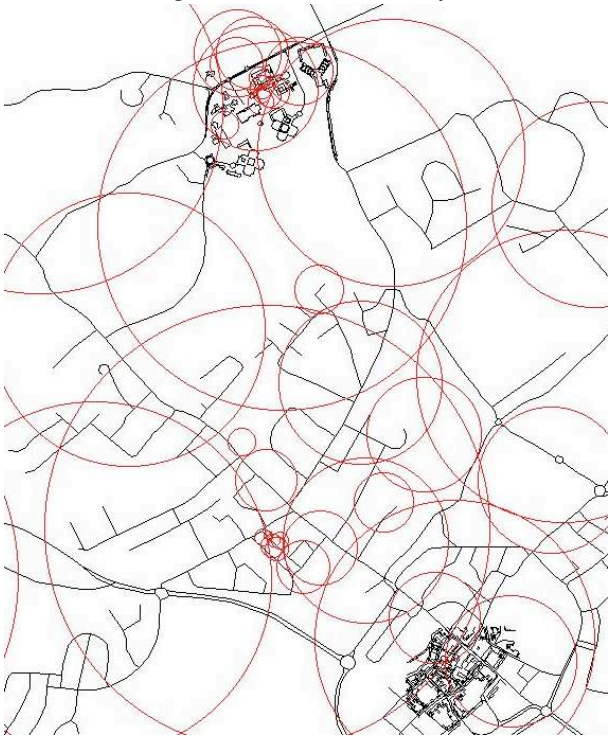

Figure 5: CAMS after 34 days

deal with these problems a Context-aware Memory Structure (CAMS) is proposed which is capable of performing lossy context data compression, determining relevant context attributes, and creating Episodes capturing human activity consisting of consecutive events. The division of CAMS into Snapshot Memory and Episodic Memory is essential for keeping the complexity of the system at an acceptable level. Context prediction is partly already present in CAMS in particular in the Episodic Memory, therefore a possible application using CAMS is a context predictor using resource levels included in the snapshot and episodic memory to calculate probabilities of future context, enabling applications to be pro-active. Promising results have been presented, but more experiments involving a larger number of users and additional context classes, such as accurate indoor location, are needed to evaluate CAMS further.

References

Dey, A. K., and Abowd, G. D. 1999. Towards a better understanding of context and contextawareness. Technical Report GIT-GVU-99-22, Georgia Institute of Technology, College of Computing.

EPOCH. The european framework 6 network of excellence in processing open cultural heritage. http://www.epoch-net.org(accessed May. 2006).

Lonnie D. Harvel; Ling Liu; Gregory D. Abowd; Yu-Xi Lim; Chris Scheibe; and Chris Chatham. 2004. Context cube: Flexible and effective manipulation of sensed context data. volume 3001 of *Lecture Notes in Computer Science*, 51–68. Springer.

Mohr, P.; Timmis, J.; and Ryan, N. 2005. Immune inspired context memory. In *1st International Workshop on Exploiting Context Histories in Smart Environments*, 4.

Neal, M. 2003. Meta-stable Memory in an Artificial Immune Network. In Timmis, J.; Bentley, P.; and Hart, E., eds., *Proceedings of the 2nd International Conference on Artificial Immune Systems*, volume 2787 of *Lecture Notes in Computer Science*, 229–241. Springer.

Patterson, D.; Liao, L.; Fox, D.; and Kautz, H. 2003. Inferring high-level behavior from low-level sensors. In *Proceedings of the Fifth International Conference on Ubiquitous Computing, Seattle, USA*.

Rene Mayerhofer; Harald Radi; and Alois Ferscha. 2003. Recognizing and predicting context by learning from user behavior. 25–35.

Weiser, M. 1991. The computer for the 21st century. *Scientific American*.