Exploiting Immunological Properties for Ubiquitous Computing Systems

Philipp H. Mohr, Nick Ryan, and Jon Timmis

Computing Laboratory, University of Kent, UK
{phm4,nsr,jt6}@kent.ac.uk

Abstract. The immune system exhibits properties such as learning, distributivity continual adaptation, context dependent response and memory during the lifetime of a host. This paper argues that such properties are essential for the creation of future context-aware and ubiquitous systems where the need for such properties is becoming increasingly clear. To that end, we present an immune inspired system, which draws heavily on the immune network metaphor to create a meta-stable context-aware memory system that could be delivered in small hand-held devices.

1 Introduction

Locating the information, tools, and other resources that we require, when we require them, is a potentially time-consuming and frustrating task. Systems that automatically make such resources available when needed are highly desirable, but to produce them presents a significant challenge. Such ideas are not new, there are various recommender systems [1,2], but these are still a long way from being portable, producing meaningful results in real time, and adapting to gradual changes in the user's behaviour.

Context-aware or, perhaps more correctly, context-sensitive systems are an important aspect of Ubiquitous computing. For a review of earlier work see Dey and Abowd [3]. Context-awareness describes the capability of a system to recognise changes in its environment and adapt its behaviour accordingly. 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.

To create such an "activity-sensitive" system, the problem arises 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.

We believe that a system which is capable of fulfilling the above task should be unsupervised, work in real-time, use online learning, and be continuous, noise tolerant, and resource friendly. Having examined the ubiquitous computing literature, there seems little evidence to suggest that traditional approaches to such a problem will deliver; see for example [3]. Having investigated the area of Artificial Immune Systems (AIS), we believe that certain immune algorithms may be a good choice to help address some of these challenges facing Ubiquitous computing. AIS have been used for data classification, clustering, and compression, in continuous and online learning systems where adaptability is paramount. They have been applied in areas such as computer security [4] and email classification [5], but not yet to the area of context-aware systems.

It is our goal to develop a system which can support context-aware applications to deliver appropriate resources to users derived from an assessment of their current activity and needs based on the context in which they find themselves.

In Section 2 the paper introduces Ubiquitous systems and relevant work in the field. In Section 3 we present work in the area of Artificial Immune Systems which is relevant to the work proposed in this paper. Section 4 outlines our proposed system, as well as reporting initial experimental work. Section 5 presents our conclusions and future aims.

2 Ubiquitous Computing

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 [6]; if there are differences of emphasis within this community, they lie in details such as the extent of invisibility. In this paper we use the term "ubiquitous computing" to include all these nuances.

An important aspect of ubiquitous computing is context-awareness; Dey *et al.* provide definitions for context and context-aware systems, which are widely accepted in the field:

> "*Context* is any information that can be used to characterize 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."

> "A system is *context-aware* if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users task."

The field of ubiquitous computing began with relatively trivial applications. Some were simple rule-based systems which had a handful of rules, e.g. that the light should be switched off when the last person leaves the house. Others dealt with the automatic presentation of information about places of interest in the user's immediate vicinity, typically using a GPS receiver to detect location and then query a database. However, it was soon realised that these simple systems are not enough to achieve the ambitious goals described above, and that far more sophisticated systems are required whose creation can only be achieved by collaborative effort between ubiquitous computing and other fields.

Ubiquitous computing has a wide range of sub-fields, we highlight the major ones and some existing work which has been carried out. A key area is capturing data from sensors, both worn on the body and distributed throughout the environment. Contextual information captured by sensors comprises attributes which describe relevant details about the environment, e.g. time, location, weather, mood of the user, activity, nearby people, etc.; intuitively, separate sensors may be required for each attribute. Bao and Intille [7], for example, try to detect physical activities by placing five biaxial accelerometers on different parts of the body. Other projects like "The Smart Floor" [8] have integrated sensors in the floor which are used to identify the people walking on it (identification is based on each persons unique walking pattern). Much emphasis has been placed on sensing location information. Determining outdoor location is fairly straightforward using GPS receivers, but indoor location is difficult to determine without significant investment in infrastructure. There are many different technologies for location detection, for a detailed overview see "Location Systems for Ubiquitous Computing" [9].

An area of particular concern in ubiquitous systems is privacy and security. In the early days not much attention was given to security and privacy issues, mainly because it is a difficult problem to resolve and working prototypes where needed first to prove the general concept of ubiquitous computing. Currently a range of privacy enhancing infrastructures are available such as the one developed by Osbakk and Ryan [10].

The storage of data about users and their environment is another important area which is split into two sub-areas, one concerned with storing as much data as possible in order to create a complete memory and develop ways of retrieving useful information based on that data, the other concerned with storing only relevant information. For an example of the former see [11], for an example of the latter see [12]. People in the first field argue that memory is cheap and more or less unlimited, but this might not be a feasible approach for all applications as only the information is stored in a sequential way and connections may be difficult to extract.

Next we focus on projects and frameworks related to classifying and predicting users' behaviour. Mozer's "Neural Network House" tries to learn the behaviour of the inhabitants of a house to save energy, e.g. the light is only switched on when the system expects someone to enter a room. The system is based on a feed-forward neural network and trained with back propagation [13].

Kröner *et al.* [14] present a mobile personal assistant called SPECTOR. The aim of the system is to assist the user with tasks or problems which occur in a daily life situation, e.g. the user should be alerted when the sum of the prices of the shopping items in his basket is greater than the amount in his bank account to avoid embarrassment at the checkout. A machine learning technique is used to create a decision tree which reflects the user's behaviour (a training set is required to jump-start the system). In order for the system to adapt to changes in user behaviour, or improve the decision making process, they developed two versions of a decision tree editor: one abstracts away from the underlying decision tree and allows the user to tick or untick boxes of attributes related to a certain task; the other allows direct editing of the decision tree.

Mayrhofer *et al.* [15] have developed an interesting framework designed to work on resource limited devices which tries to anticipate the user's behaviour and adapt to her needs in advance. Their framework consists of four major steps. The first is feature extraction which turns raw sensor data into usable context attributes, the second is classification of the input data, the third is the labelling of a set of context attributes to give them real world meaning, and the fourth is the prediction of the user's future context. They put strong emphasis on the exchangeability of individual components by providing well defined interfaces. Currently the classification is done using the Growing Neural Gas algorithm, which produces reasonable results. No quantitative evaluation has been done yet for the prediction step.

3 Immune Networks

The immune memory mechanism proposed by Jerne [16], commonly known as the Immune Network Theory, attempts to explain how the immune system maintains a memory of encounters with antigens (Ag) in the absence of antigenic stimulus. It is based on the assumption that B-cells can, in addition to being able to recognise antigens, recognise each other through interactions of idiotopes [17, 18]. This allows for the formation of a network structure of stimulating and suppressing signals which propagate through the network, boosting or decaying the concentration of a particular B-cell. The network is self-organising and self-regulating and, while not widely accepted from an immunological point of view, has been widely exploited in the area of AIS.

3.1 Artificial Immune Networks

Work by Neal [19] proposed a meta-stable immune network algorithm capable of dynamically identifying new clusters in a continual stream of data. The algorithm was based on the immune network theory (outlined above) and is a result of work in [20] and [21]. The algorithm is divided into two main phases: the first phase creates the initial network, and the second adapts it to a changing environment. The algorithm creates a reduced map of the input data space, where each data item is a vector (in the case of Neal, this was a vector of four real numbers). In the subsections below we explain how the Meta-stable Memory structure works by splitting the algorithm into three parts: network initialisation, network growth, and survival in the network.

Network Initialisation. An initial network is created by randomly selecting a number of vectors from the data set being analysed. The whole data set is referred to as the antigen pool (Ag). The selected vectors are then used to create an Artificial Recognition Ball (ARB) and added to the network. An ARB represents a region of antigen space which is covered by a particular type of B-cell, excluding the need for repetition of individuals [21]. The Ag data items are then matched against each ARB in the network. If the Euclidian distance between two ARBs is less than a pre-defined value, referred to as the Network Affinity Threshold (`NAT`), they become neighbours by the creation of a link between them. Connected ARBs stimulate each other, which allows them to survive longer. All information regarding connected neighbours and resource levels of the ARBs are stored locally within each ARB.

Network Growth. The primary response is invoked if the nearest ARB to the Ag being presented is further away than the `NAT` value; in this case the Ag is converted into an ARB and added to the network — this is the growth mechanism in the network. It should be noted that no cloning or mutation in the traditional sense of AIS is being performed here. The secondary response is invoked if the Ag falls beneath the `NAT` value of any ARB, in this case the Ag will not be added to the network. This is under the assumption that the existing ARB which is the closest already represents to a sufficient degree the region of the input space into which the Ag falls. However, the presence of the Ag will not be forgotten, as the matching between the ARB and Ag will increase the stimulation level of the ARBs in the network inversely proportional to the distance.

Survival in the Network. As mentioned previously, each ARB records a resource level which changes continuously and can increase based on the level of stimulation. The stimulation level is calculated by summing the affinity (match value) between all Ags presented during one iteration of the data, and the affinity between all connected ARBs. The more stimulated an ARB is, the more resources it can claim. The resource level can only grow to a pre-defined upper limit and shrink to a pre-defined lower limit. The shrinking is caused by a decay function, which is applied to all resource levels when a new Ag is presented to the network (in the case of Neal this was a linear decay). When the resource level of an ARB falls below this lower limit it is removed from the network.

3.2 Initial Observations

Before attempting to deploy this algorithm in our application, intensive studies of it where undertaken. We identified a major problem and redundant operations within the algorithm as published. When a new Ag is presented to the network and is within the `NAT` value of any ARB, it will not be converted into an ARB and therefore not be added to the network. Conversly, when a new Ag is outside the `NAT` value of any ARB, it will be converted to an ARB and added to the network. However, two ARBs will only become neighbours if they lie within each others `NAT` value but, as we mentioned above, this cannot be the case, because one cannot lie within the `NAT` value of another.

The neighbouring ARBs Mark Neal presents have two origins, some are introduced by the random selection of starting ARBs, and the others appear due to a fault in the code of his prototype implementation; the fault is described using the code below:

```
1   result = AIS.present(data[dn]); // Find closest ARB and
                                        record distance
2   dn++;
3   if (result < (AIS.NAT) )
4   {
5      //secondary response
6      AIS.allocate();
7      AIS.cull(); // only cull
8   }
9   else
10  {
11     // primary response
12     AIS.allocate(); // allocation of resources during
                                   repertoire expansion
13     AIS.clone(data[dn]); // repertoire expansion itself...
14     AIS.relink(); // change !
15     AIS.cull();
16  }
17  // dn++ should be positioned here
```

Where `data` represents the data array and `dn` the index number an Ag has in the data array, and `result` is the distance from `data[dn]` to the closest ARB.

In line 1 the distance from `data[dn]` to its closest ARB is stored in `result`. We will refer to line 2 as `dn+1` (this is where the error lies). Line 3 tests if `result` is less than the `NAT` value; the true branch is fine. The false branch causes the problem, because when `result` is larger than `NAT`, `data[dn]` is converted to an ARB and added to the network, but instead of adding the data item which was used to calculate `result`, `data[dn + 1]` is added; this leads to non-deterministic behaviour. To fix this problem `dn++` should be moved to line 17. However, this change introduces a new problem, as it is not possible for two ARBs to be positioned within each others `NAT` value, so no neighbours will be created. Therefore, a further fix is required: a small change needs to be made to the `ImmuneNetwork::addARB(ARB* clone)` method in the published code by Neal. An additional, slightly larger `NAT` value needs to be used to allow two ARBs to fall within a `NAT` value where they can become neighbours. The modification is shown below:

```
// original:
if (distance(clone->pattern, nodes[others]->pattern) < NAT)

// modified:
if (distance(clone->pattern, nodes[others]->pattern) < NAT * 1.3)
```

The NAT value is incremented by 30% in the second line, which produces sensible results. Furthermore, with these changes there is no need for an initialisation phase — the algorithm can start with an empty structure and the Meta-Stable memory structure gradually evolves.

4 A Context-Aware Immune System: CAIS

Now the two background areas have been reviewed, attention can be turned to our application area. As previously stated, the goal of our system is to assist the user through the provision of a user friendly context-aware system that provides an assessment derived from their current context.

Ideally, as mentioned, the system should be implemented on a resource-constrained device and must be effective even in the absence of connectivity. In practice, context-aware software running on mobile devices 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. There are some benefits to autonomy and keeping more information locally on the user's device, particularly if privacy of sensitive contextual data is an issue. We have to assume that control is lost over any information which is disclosed. If there is no need for disclosure, then privacy can be assured, however this complicated issue of privacy is beyond the scope of this paper.

We propose to use the feature extractor developed by Mayerhofer, *et al.* (see Section 2), because it supplies a feature vector which consists of individual context attributes. Currently their system only provides support for the user's current context, but this is due only to their choice of sensors. We believe an extension providing a range of possible activities can be implemented with reasonable effort.

An important property of the algorithm is adaptation to a change in the user's behaviour. The adaptation should not happen too fast, but also not too slowly. Petzold *et al.* show with the examples of a 1-State and 2-State Context Predictor that a sudden change from one state to another is not sufficient to reflect the user's behaviour [22].

We believe that the immune inspired algorithm outlined above enables the system to capture the gradual change in humans behaviour. Furthermore, the algorithm's memory structure allows for data compression and adaptability, and online learning enables continuous operation.

4.1 Framework

Work in [18] proposes a framework for Artificial Immune Systems which consists of three components. The first describes the data structure and representation used, the second talks about affinity measures, and the third describes the algorithm. We adopt this framework and explain below our system in terms of these three components.

Representation. The system's inputs consist of the user's context and possible options (e.g. different activities such as "lunch" or "meeting"). The user's context is represented by an attribute vector, $\langle a_1, a_2, ..., a_n \rangle$, which contains attributes along with their attribute identifier — note that attributes can appear in an arbitrary order. Possible options are also represented by attribute vectors, one for each option (options may comprise of an arbitrary number of attributes). An example attribute vector is given below:

```
⟨ GSM.CellID = 04x6,
  Wlan.MacAddress = 0A:40:C3:8D:00:32,
  Location.Building = Library,
  Time.Hour = 18:30 ⟩
```

Where `GSM.CellID`, `Wlan.MacAddress`, `Location.Building`, and `Time.Hour` are attribute identifiers, and `04x6`, `0A:40:C3:8D:00:32`, `Library`, and `18:30` are their respective values.

The output is a list containing a predefined number of options of probable significance to the user in respect to her current context (in descending order of relevance). The list could have the following form:

1. Cinema
2. Lunch with Tom
3. Train home in 30 minutes
4. etc.

Figure 1 shows the input of context and possible options, as well as the list of options as output. Context attributes are stored in the system as ARBs. Every ARB has its own resource level R and current stimulation level L. The same attribute can occur multiple times in the same context and/or different contexts. The notion of ARBs allows us to capture all of the different occurrences of an attribute by storing it in a single ARB.

The representation we are using is an n-dimensional hierarchical network structure, where each dimension represents a different attribute, and is itself a network structure. Figure 2 shows an example of the proposed data structure. The example consists of three dimensions and nine ARBs. Attributes from different dimensions which appear in the same context are connected by cross-dimensional-links, in our example ARB_4 and ARB_2 link D_1 and D_2, and ARB_1 and ARB_6 link D_2 and D_3. These links have a resource level L associated with them which reflects the likelihood that these two attributes occur in the same context. Furthermore, every dimension itself contains a network structure, for example dimension D_1 contains ARB_4 and ARB_5, which are connected with each other. The network structure of each dimension is a Meta-stable memory structure (our improved version) which contains at least one ARB, therefore ARBs in the same dimension can become neighbours if they are similar, as explained in Section 3.2.

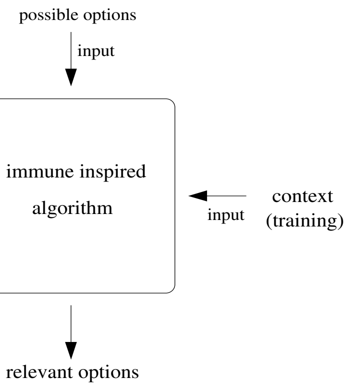

Fig. 1. CAIS

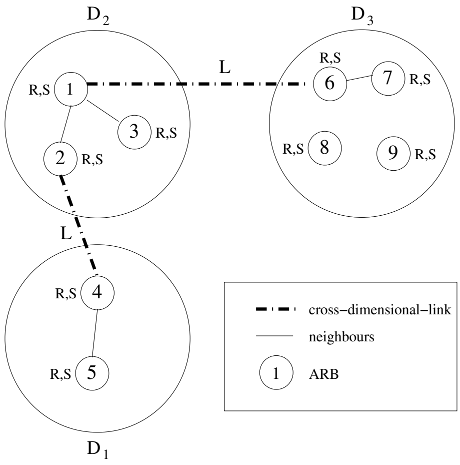

Fig. 2. Three dimensional example

Affinity measures. Affinity is the mechanism by which the distance between two elements is calculated. We use the affinity measure to determine how similar two ARBs are, if they are close enough to be neighbours, and how much one can stimulate the other. Affinity measures have to be chosen carefully in order for the algorithm to work effectively, for a discussion about misuse of affinity functions please refer to [23]. Measuring the distance between GPS co-ordinates is fairly straight-forward as standard Euclidian distance can be used, but measuring the difference between non-numeric attributes is more difficult, e.g. the difference between two mobile phone cell IDs. Mayerhofer, *et al.* have already derived affinity functions for attributes currently supported by their framework [15]; we will adapt these functions for use in our system and develop additional ones as required.

Algorithm. The main part of the algorithm is the process of learning the user's behaviour. At the beginning of its lifetime CAIS is not able to produce resource options that may be made available, as it needs to learn the user's behaviour first. The learning and adaptation of the classification mechanism is achieved by continuously feeding the local context into the system.

To help explain the learning process we define three sets: A is the set of all attributes, e.g. `Time`, `Location`, etc.; D_i is the set of ARBs in dimension i, where $i \in A$ (i.e. D_i represents a particular attribute class); and S is the set of all dimensions, $S = \bigcup_{i \in A}\{D_i\}$. In Figure 2, $A = \{1, 2, 3\}$, $D_1 = \{ARB_1, ARB_2\}$, and $S = \{D_1, D_2, D_3\}$. For an attribute i, if D_i is already an element of S, attribute i is stored in D_i. If the dimension is not an element of S, a new dimension, D_i, is created and added to S. The following pseudocode explains the learning process of CAIS:

```
LOOP
   get next input vector
   FOR EACH (attribute $i$ in vector)
      IF ($D_i \in S$)
         stimulate all existing ARBs with the attribute
         IF (distance of attribute to all ARBs > NAT value)
            convert attribute to ARB and add to dimension
         IF (distance of attribute to any ARB < NAT * p)
            make them neighbours
         ELSE
            create dimension and convert attribute to ARB and
               add to dimension
            call decay function on all resource levels
            create cross-dimensional-links between all attributes
         IF (cross-dimensional-link already exists)
            stimulate cross-dimensional-link
```

The user's behaviour is learned from the continuous input of local context. Each attribute in the context attribute vector is presented to the system. First a check is performed to see if the dimension exists to which the attribute belongs, if it does then the attribute stimulates all existing ARBs within this dimension — stimulation depends on the distance to all ARBs within this dimension, which is calculated using the appropriate affinity function. If the distance to all ARBs is greater than the `NAT` value, it is converted into an ARB and added to the dimension, furthermore if the attribute's distance is within `NAT * p` of any ARB, where `p` is the extension to the `NAT` value within which neighbours are created, it becomes their neighbour. If the dimension does not yet exist, it is created and the attribute is converted into an ARB and stored in the new dimension. After all the attributes have been considered, cross-dimensional-links between them are created or, if they already exist, are stimulated. Both levels decrease due to decay functions, details on the increase and decrease of stimulation levels will be presented in future publications after extensive experiments have been carried out.

In-order to demonstrate the capabilities of the algorithm, creation of a list of appropriate resource options is a possible application. The list of options is generated by taking all possible options as input, CAIS will rate them individually by comparing them to its memory. After all of them have been rated a list is constructed in descending order of rating. Tasks with no match may be included in the list to point out unseen options.

4.2 Prototype

In-order to achieve our goal of a ubiquitous system we have to go through an incremental process. First we identified the requirements, namely continuous operation, high data compression, noise tolerance, and forgetting of redundant patterns. Neal's Meta-stable Memory structure seemed a suitable candidate, as it fulfils the requirements listed. We developed an experimental prototype based on the improved Meta-stable Memory structure described in Section 3 in-order to test the feasibility of using the algorithm in CAIS. The prototype clusters and compresses GPS co-ordinates and displays the current state of the internal memory structure on a map. This allows us to understand and follow the algorithm while running.

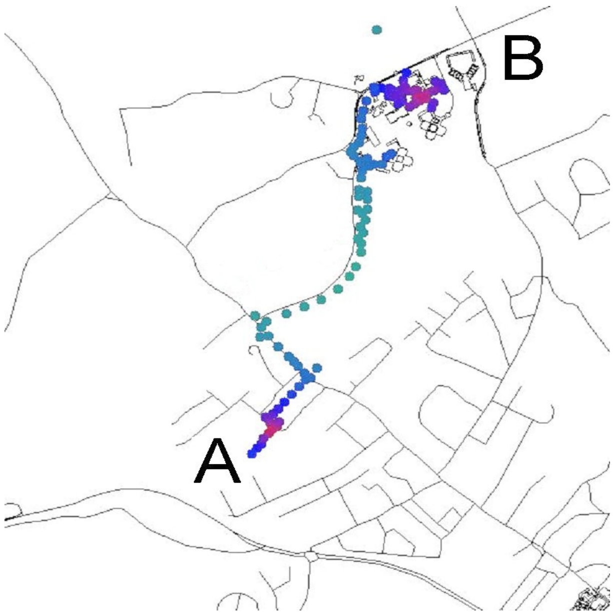

Fig. 3. Tracking GPS Data Using a Meta-Stable Immune Memory

We ran the algorithm with different data sets and different `NAT` values. Figure 3 shows the output after the algorithm iterated through 200000 points. ARBs are represented by small circles and their resource level is visualised by the darkness of the circle. These points where collected over a period of four months for the same journey from someones house (point A) to the university (point B). The map shows a high activity at point A and point B, and a low to medium activity in between — a standard averaging technique showed a very similar result. The algorithm reduced the points to about 150, and due to the use of ARBs the information about the lost points is retained by the stimulation levels of the ARB. Furthermore, noisy data which was mostly caused by occasional inaccuracies in the GPS measurements is eliminated by the decay function. The time intervals between the capture of individual points ranged from 5 seconds to 5 minutes. The `NAT` value for this particular example is 15, which corresponds to 15 meters, therefore all points within a 15 meter radius were represented by a single ARB.

The prototype produces promising, and apparently unique results, which we believe justify the usage of the improved Meta-stable Memory structure in CAIS.

5 Conclusion

In general, context-aware systems suffer from a lack of generality and tend to be tailored to a specific set of inputs. In addition, they may require large amounts of data to be of use. In an ideal world we want such systems to be able to generalise, be adaptable, and be able to compress or reduce the amount of data required in-order to function. Immune inspired algorithms appear to be a good candidate for generalisation, adaptability and compression. We believe the immune inspired algorithm presented in this paper has the potential of realising a ubiquitous system that can recognise situations which are of interest to the user, and adapt to a wide variety of user behaviour and environmental inputs. An immune inspired system has been presented that we feel will form the basis of such a ubiquitous system. The first step towards achieving this goal was presented, which was based on tracking GPS data. Clearly more work is needed, but we feel this is an encouraging first step towards a more ambitious goal.

References

1. N. Good, J. Schafer, J Konstan, A Borchers, B Sarwar, J Herlocker, J. Riedl: Combining collaborative filtering with personal agents for better recommendations, in proceedings of the sixteenth national conference on artificial intelligence (1999)
2. Xiaobin Fu, J.B., Hammond, K.J.: Mining navigation history for recommendation, in proc. 2000 conf. on intelligent user interfaces (2000)
3. Abowd, G.D., Dey, A.K.: Towards a better understanding of context and context-awareness, http://www.cc.gatech.edu/fce/contexttoolkit/chiws/dey.pdf (2000)
4. Kim, J., Bentley, P.: The human immune system and network intrusion detection (1999)

5. Secker, A., Freitas, A., Timmis, J.: AISEC: An Artificial Immune System for E-mail Classification. In Sarker, R., Reynolds, R., Abbass, H., Kay-Chen, T., McKay, R., Essam, D., Gedeon, T., eds.: Proceedings of the Congress on Evolutionary Computation, Canberra. Australia, IEEE (2003) 131–139
6. Weiser, M.: The computer for the 21st century. Scientific American (1991)
7. Ling Bao, Stephen S. Intille: Activity recognition from user-annotated acceleration data, in Proceedings of Pervasive Computing 2004,Linz/Vienna,Austria. (2004)
8. Orr, R., Abowd, G.: The smart floor: A mechanism for natural user identification and tracking (2000)
9. Jeffrey Hightower, G.B.: Location systems for ubiquitous computing. IEEE Computer **38** (2001) 57–66
10. Patrik Osbakk, Nick Ryan: A privacy enhancing infrastructure for context-awareness, position paper for the 1st UK-ubinet Workshop, Imperial College, London, UK (2003)
11. Kiyoharu Aizawa, Tetsuro Hori, Shinya Kawasaki, Takayuki Ishikawa: Capture and efficient retrieval of life log, in Proceedings of Pervasive Computing 2004 workshop on memory and sharing of experiences,Linz/Vienna,Austria. (2004)
12. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps, proceedings of ieee sixth international symposium on wearable computing (iswc02) (2002)
13. M. C. Mozer: The neural network house: An environment that adapts to its inhabitants. in Proceedings of the AAAI 1998 Spring Symposium on Intelligent Environments (1998)
14. Alexander Kroener, Stephan Baldes, Anthony Jameson and Mathias Bauer: Using an extended episodic memory within a mobile companion (2004)
15. Rene Mayerhofer, Harald Radi, Alois Ferscha: Recognizing and predicting context by learning from user behavior, in Proceedings of The International Conference On Advances in Mobile Multimedia (MoMM2003),Austrian Computer Society (OCG) (2003)
16. Jerne, N.: Towards a network theory of the immune system. Ann. Immunol (1979)
17. Farmer, J.D., Packard, N.H., Perelson, A.S.: The immune system, adaptation and machine learning. Phsica **22** (1986) 187–204
18. de Castro, L., Timmis, J.: Artificial Immune Systems: A New Computational Approach. Springer-Verlag, London. UK. (2002)
19. Neal, M.: Meta-stable Memory in an Artificial Immune Network. In Timmis, J., Bentley, P., Hart, E., eds.: Proceedings of the 2nd International Conference on Artificial Immune Systems. Volume 2787 of Lecture Notes in Computer Science., Springer (2003) 229–241
20. Timmis, J., Neal, M.: A resource limited artificial immune system for data analysis. Knowledge Based Systems **14** (2001) 121–130
21. Neal, M.: An artificial immune system for continuous analysis of time-varying data. In Timmis, J., Bentley, P.J., eds.: Proceedings of the 1st International Conference on Artificial Immune Systems (ICARIS). Volume 1., University of Kent at Canterbury, University of Kent at Canterbury Printing Unit (2002) 76–85
22. Jan Petzold, Faruk Bagci, W.T., Theo Ungerer, i.A.I.i.M.S..: Global and local state context prediction (2003)
23. Freitas, A., Timmis, J.: Revisiting the Foundations of Artificial Immune Systems: A Problem Oriented Perspective. In Timmis, J., Bentley, P., Hart, E., eds.: Proceedings of the 2nd International Conference on Artificial Immune Systems. Volume 2787 of Lecture Notes in Computer Science., Springer (2003) 229–241