Supplementing Case-based Recommenders with Context Data*

Lorcan Coyle, Evelyn Balfe, Graeme Stevenson, Steve Neely, Simon Dobson, Paddy Nixon, and Barry Smyth

Adaptive Information Cluster
School of Computer Science & Informatics
UCD Dublin
Belfield, Dublin 4
Ireland
firstname.secondname@ucd.ie

Abstract. We propose that traditional case-based recommender systems can be improved by informing them with context data describing the user's environment. We outline existing applications with similar objectives and describe an application of our own — Ticketyboo — which uses music listening preferences and context information from users' calendars to recommend tickets for music concerts. This data is gathered by virtual sensors that monitor each user's music player and calendar applications. The novelty of this approach is that context data is provided to Ticketyboo via a dedicated context infrastructure. This results in a clear separation between the providers and consumers of context data. By utilising context data in this way, minimal user input/feedback is required to guide the system since the need for explicit user feedback is negated.

1 Introduction

The goal of traditional recommender systems is to help individuals to more effectively identify items of interest from a potentially overwhelming set of choices [1]. As such, these systems aim to provide answers (recommendations) to users in response to their query/goal. The context that has been used in existing recommender systems has mostly been in the form of user preferences. We propose a novel application — Ticketyboo — which infers implicit context from virtual sensors in the pervasive computing domain. These sensors monitor a user's interaction with their media player and their calendar applications and use the context inferred from these applications to make concert ticket recommendations.

Traditional recommenders rely on the user to initiate the recommendation process by entering a query or expressing a goal, the novelty of Ticketyboo is

* The authors would like to acknowledge the support of Science Foundation Ireland (under the grant "Secure and Predictable Pervasive Computing") and the Irish Research Council for Science, Engineering and Technology.

that it infers context from external sources and uses this context information to automatically generate recommendations for the user without the need for a goal to be explicitly expressed. This novelty addresses the argument that was first introduced by Aha et al. [2] where they drew attention to the fact that users often don't have a definite problem description nor domain expertise. They remarked that conversational Case-Based Reasoning (CBR) recommender systems improved upon traditional CBR recommender systems because the system interacts with the user in a conversation mode to construct the full problem specification, rather than the user having to explicitly define it. Thus the user does not need to have extensive domain expertise in order to receive their recommendations. Ticketyboo users require no domain expertise and do not have to express a problem that the system is required to solve. Any knowledge that Ticketyboo requires in order to make recommendations is inferred implicitly from two desktop applications, a media player and a calendar (in this instance, iTunes and iCal).

Ticketyboo receives its context data from a context infrastructure called Construct (described in more detail in Section 3.2). However, Ticketyboo is just one example of how access to context data via context-infrastructures can provide advantages to recommender systems. In Section 4.3 we describe how other recommender systems could be enhanced from access to context data in a similar way.

The rest of this paper is laid out as follows: Section 2 describes how context is obtained in the pervasive computing domain and how other types of context have been applied to existing recommendation systems; Section 3 describes the Ticketyboo application and shows how context data may be used to improve recommendations; Section 4 describes an evaluation framework for Ticketyboo, proposed improvements to the recommendation process, additional context that could be applied to existing recommender systems and the possibility of using other, less structured, contexts. Finally, Section 5 concludes the paper and proposes some further work.

2 Context

Before we continue, we should first clarify what we mean by context. Webster defines context as "the interrelated conditions in which something exists or occurs". In the pervasive computing domain Dobson et al. defines it as such: "The context of a system captures the environment in which it operates, including all additional or non-functional aspects that, while not being core to the system's behaviour, nevertheless affect the way in which that behaviour should be optimised" [3]. This definition also applies to the CBR domain where context data can be used to optimise recommender systems. In this paper we combine the two domains of pervasive computing and CBR to implicitly infer context from pervasive computing sensors and use this context data to optimise the concert recommendations made to users. In this section we look at context in both the pervasive computing domain and the CBR domain.

2.1 Context in Pervasive Computing

The concept of context is well understood in the pervasive computing domain. Initial research was centred on exploiting available location, time, and identity information, and focused on demonstrating the utility of context awareness as an application feature. The Active Badge call forwarding system [4] used identity and location information, collected using an IR-based tags worn by users, to route telephone calls to the phone closest to the intended recipient. The CyberGuide [5] used a location-aware handheld device and map display to provide tourists with a computerised tour-guide. Using the handheld, users could find directions, and get background information on items of interest. CyberGuide also used the user's location information and travel history to suggest places to visit.

Recent work in the pervasive computing domain has focused on developing infrastructures to support the development of general purpose context-aware applications. Such infrastructures aim to move as much work as possible of the work as possible onto network-accessible middleware infrastructures. This makes it easier to deploy new sensors, devices, and services, and makes it easier to share sensor and context data - placing the burden of acquisition, processing, and interoperability on the infrastructure instead of individual devices and applications. One such system is Context Fabric [6], a distributed infrastructure that uses structures called *infospaces* to collect information on people, places, and things. Sources of information, such as sensors, can publish their data to infospaces, allowing interested applications to query for required context on demand [7]. Another example infrastructure is the Aura Contextual Information Service [8], which aims to provide applications with a single interface for accessing context information. The Aura CIS assumes that the information that is most relevant to mobile applications can be provided by supplying information about four classes of entities. These are: People, Devices, Physical Spaces, and Networks. A Network class is included to provide communication information to applications that require it (Nearby location with high bandwidth etc.). Generic physical objects, such as desks and chairs are treated as dumb devices, and vehicles as Physical Spaces with no fixed location, in order to keep the number of classes in the model small. The model also explicitly treats power supplies, such as batteries and wall sockets, as attributes of Devices and Physical Spaces respectively. Services represent information using this model, and applications are provided with a database like abstraction of these services, with which they may access context data of interest.

Over time the scope of context analysis has become much broader and today it encompasses a wide range of context descriptors. These include physical location of entities (users, devices, etc.), time, identity, current tasks, goals, and preferences. Ongoing developments are also opening up access to other types of context information. This includes analysis of a user's emotional state [9], chemical sensors for environmental monitoring [10], and detecting body position using sensor-augmented clothing [11].

2.2 Context in CBR Recommender Systems

The majority of recommender systems require the user to explicitly define their context by directly interacting with the system. One such example is Entree [12], a CBR restaurant recommender which uses context that is provided explicitly. Each time that Entree recommends a restaurant, it allows the user to provide feedback in the form of a critique or tweak. As such the restaurants that are recommended to a user are personalised according to the user's desired preferences at a given time which they have explicitly expressed.

In a world of lazy users, it is preferable for the system to infer context implicitly, I-SPY is one such CBR recommender system. I-SPY (http://ispy.ucd.ie/) is a collaborative Web search engine that introduces a form of context into Web search by implicitly determining the context of the user's search query. Context information is inferred implicitly from the searching behaviour of communities of searchers with similar information needs. I-SPY uses the CBR technique of reuse by exploiting the information in past search sessions to re-rank new result-lists for users within a community [13].

Although neither Watson [14] nor Letizia [15] are CBR recommender systems, they both implicitly infer context from user actions to drive their recommendations. Both recommender systems take advantage of user activity immediately prior to search in order to determine a suitable search context. Watson identifies and extracts informative query terms from desktop applications that the user is currently interacting with, for example Microsoft Word or Internet Explorer. Watson continually searches the Web for related documents based on their related query terms. Similarly, Letizia analyses the content of Web pages that the user is currently browsing and uses similarity term-weighting heuristics to identify important terms within the document. Letizia then proactively searches out from the current page for related pages.

Similar to both Watson and Letizia, SmartRadio [16, 17] implicitly infers context based on the user's actions. SmartRadio is a hybrid recommender system that uses CBR and Automated Collaborative Filtering to recommend playlists of music. The contents of a playlist that the user is currently playing indicates the user's current listening preferences. One of the strategies SmartRadio uses in making recommendations is that it tends to favour playlists that match the current context, i.e. playlists that are similar to the one that the user is currently listening to.

Ticketyboo also implicitly infers context from the user in an aim to provide concert recommendations to the user. The difference between this and other systems is in the manner in which this context is obtained. Those systems demonstrate the utility of context data in case-based recommender systems. However, the providers of context in these applications are part of the recommenders themselves. We propose that separation between providers and consumers of context data is desirable and that by using a dedicated context providing system from the pervasive computing community, e.g. ContextFabric [8] or Construct [18] we remove this burden from the recommender system developer. They can then gain access to all the context data available, make decisions on what contexts are

important and focus on the generation of good domain-knowledge for the CBR system. If further sources of context data are required, they can create these without having to worry about the plumbing necessary for integrating these sources of context data with the CBR application. Ticketyboo gets its context data from Construct [18], which is described in Section 3.2.

3 Ticketyboo — the Case-based Music Concert Recommender

The development of Ticketyboo came from a visualisation project where we sought to display personalised homepages for users about which there was an abundance of context data. In UCD we have a number of sensors for all kinds of context data, including location-awareness, news-feeds, weather, music preferences, calendar information and TV information. The personalised context homepage may be customised by users to display information that is interesting to them. Ticketyboo was the first recommender system to be integrated with this webpage; it consists of a frame in the webpage that links to ticket information for the music artists that a user is listening to. Figure 1 shows a screenshot of one of the author's homepage with the Ticketyboo frame magnified. Ticketyboo runs continuously in the background, examining music listening statistics and calendar information and uses these to recommend concert tickets based on the user's context, i.e. their music tastes and availability. By clicking on any of these artists, the user is sent to the relevant page on a real ticket website where they may purchase tickets for an upcoming concert by those artists. These tickets will only be offered to the user if they are at a convenient time for the user.

Fig. 1. Personalised Web Page Built by the Context-Informed CBR System

Ticketyboo itself is only one part of the solution. Logically, the application is made up of three components: the sensors that detect context data; the context architecture that maintains and delivers context; and the recommender system

that uses context data to recommend concert tickets. These components are described in the following Sections.

3.1 Sensing Context

All context data in Ticketyboo comes from the context infrastructure, which are provided to it by sensors. These sensors might be physical (e.g. temperature sensors) or virtual (e.g. a sensor that monitors a web-page). Three sensors are used to provide the context data that Ticketyboo uses; they are:

- *Music Sensors:* These monitor music applications (e.g. iTunes) for data that might describe the user's attitudes towards an artist, e.g. last time and how regularly the user played a song from that artist.
- *Calendar Sensors:* These monitor a list of published iCal and vCal calendars for data about a user's location (e.g. city) and availability for a period of time (e.g. during the next month).
- *Online Ticket Sensor:* This can be asked for information about an individual artist's upcoming concerts and available tickets for a particular location.

These sensors convert the data they want to inject into RDF and insert it directly into Construct. Construct uses Jena [19], a semantic web framework, to manage all inserted RDF data. In Section 3.3 we show how Ticketyboo makes a simple query, for a user's availability at a date in the future, from Construct that was ultimately created and inserted by calendar sensors.

3.2 Delivering Context

Until recently, the lack of a generalised approach for dealing with context information had resulted in its use within a relatively small number of applications (e.g. Cyberguide [5]). Early examples of context-aware applications dealt mainly with context in an *ad hoc* fashion, and primarily served to demonstrate the utility of context information, rather than contribute any software engineering practices to aid its ease of use. Recent work in the field of pervasive computing has seen the development of infrastructures designed to support the collection, processing, and distribution of context information [8, 20, 7], and as such, has lowered the entry barrier for applications that wish to utilise such information.

Our own platform, Construct is a software infrastructure for integrating context information in a semantically well-founded manner. The platform supports extensive inferencing and sensor fusion [3, 21], which is used to raise the level of abstraction of available data towards that required by applications. Construct has a fully decentralised architecture; devices within a smart space each run an instance. Each instance manages the local data provided by applications and sensors (physical or logical) connected to that device. All data are modelled using the Resource Description Framework (RDF), which provides a standardised way in which to model context information and properties. Collectively, the devices maintain a global model of the data within a smart space. Applications

then work with the local view of the data present on the device on which they are running using a *Query Service*.

The Ticketyboo recommender uses Construct's query service to obtain user calendar, music preference and upcoming concert information for the generation of case data. Queries are made for each appropriate piece of context data using a SPARQL query. Figure 2 shows a simple example SPARQL query that asks for the availability and location of *lorcan* on the 5th of September 2006. These queries retrieve data in RDF form. To generate case data, the Ticketyboo recommender uses Constructs query service to obtain a user's calendar and music preferences as well as upcoming concert information.

```
SELECT ?availability, ?location WHERE
        ?Agent name "lorcan" .
        ?Agent date "05/09/2006" .
```

Fig. 2. An Example SPARQL Query

The output of the recommender acts as a new input to the construct system, and is then queried for by the presentation medium (in this case, the personalised web page). It should be noted that this data is now available as context information for other applications that use Construct.

3.3 Making Recommendations

MAC/FAC (many are called but few are chosen) is a two stage similarity based retrieval model which originates from cognitive science [22]. We use MAC/FAC in Ticketyboo where the first stage represents retrieving information for all users of the system and the second stage retrieves information at the individual user level.

Figure 3 shows how the MAC/FAC model is applied to Ticketyboo. The first stage, MAC, selects a large number of cases from the case-base. Cases — all information relating to a concert, for example artist, location, date, time, ticket price etc. — are chosen using the broad context of all music listed by *all* users which we obtain from their desktop media players. This music can encompass a wide variety of artists thus resulting in a large number of cases being selected for reuse. The second stage, FAC, refines the number of selected cases by taking into account the more rigorous context relating to individual users. Firstly a user's location context is inferred from a desktop calendar and this information is used to reduce the number of cases from the case-base based on whether or not the user could attend a particular concert. Secondly the individual user's music preferences are inferred from the media player and the cases are further reduced.

It is computationally inexpensive to extract concert information an online ticket store, even for a large number of artists. Therefore as a user adds a new

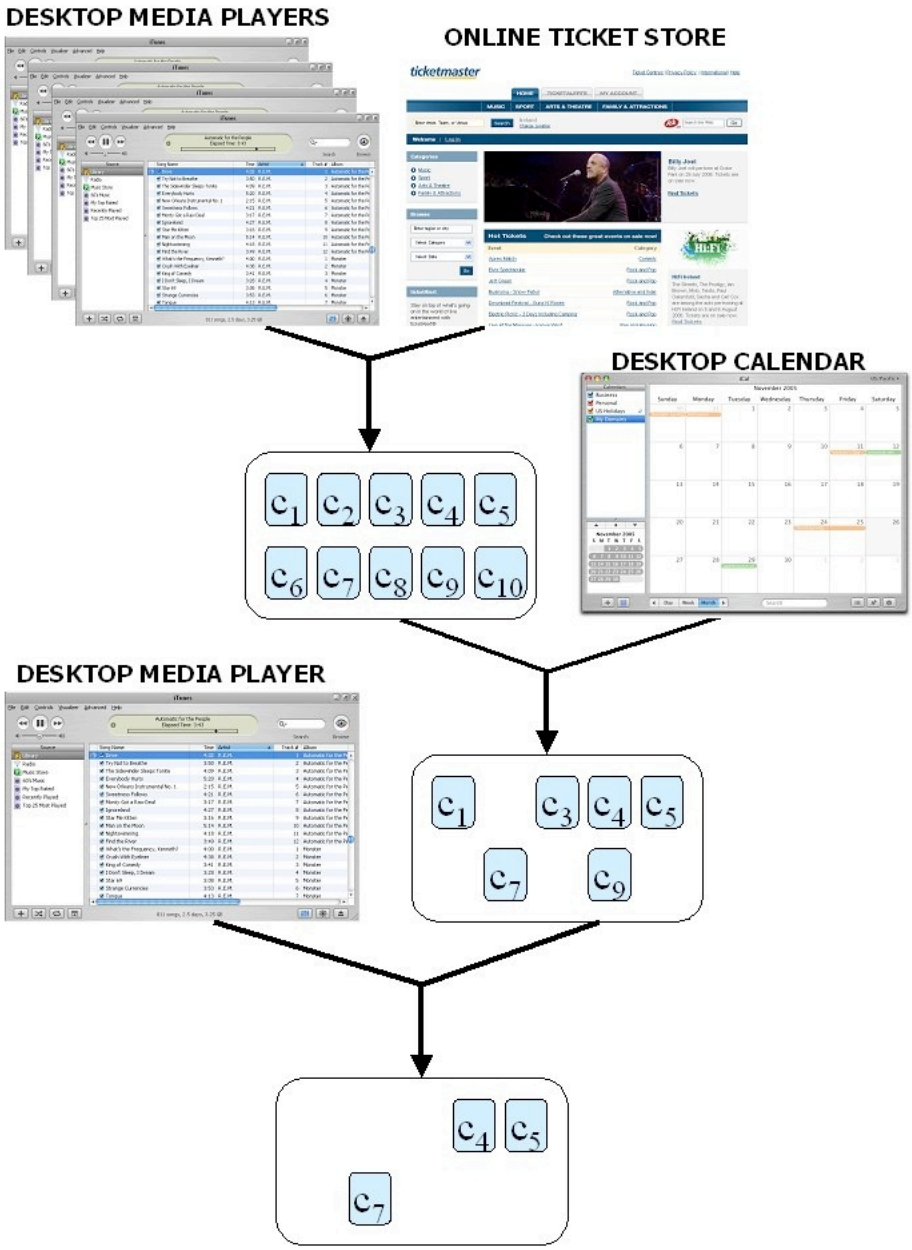

Fig. 3. The MAC/FAC retrieval stages for recommendation

album to their media player, the system can easily scrape the online ticket store for concert information relating to this new artist and update the case-base accordingly. As the recommendation process proceeds through the system, the expense increases. The most computationally expensive task is that of inferring the user's exact music preferences by firstly determining the popularity of an artist by the frequency which its albums/songs were played through the media player and secondly by examining the implicit ratings assigned to songs by the user. This multistage retrieval process is preferable because it ensures that the most expensive task is carried out on the least number of cases.

4 Discussion and Future Work

Due to the early stage that this work is at, there are several points for future work. This work has also raised many observations about context in our minds that should be further discussed. In this Section we describe an evaluation framework for Ticketyboo; some proposed improvements to the recommendation process; how context data could be used to improve some existing recommender systems; and finally the possibility of using further, less structured, context information for decision support applications.

4.1 Evaluation Framework

We intend to evaluate our approach by comparing context-boosted recommendations with recommendations made without the benefit of context. This will involve comparing the performance of Ticketyboo with a simple recommender that doesn't take into account context data (e.g. data from the calendar sensor).

While an online evaluation of any recommender system is preferable [23], it is not appropriate to evaluate Ticketyboo in this way. This is because of the fact that a successful online evaluation is dependant on a mature fully-engineered system with a large group of regular users. Therefore *implicit* feedback will be used to perform our evaluations. When users visit Ticketyboo, they are presented with a set of recommended tickets for the artists they are interested in. They can select any of these tickets for purchase, in which case they are forwarded to the appropriate ticket vendor website. We record user selections and use them to evaluate our various recommendation techniques offline. This is a *best of both worlds* evaluation framework, using online data with offline analyses of recommender strategies [24].

The recommendation process works by generating recommendation scores for each of the offered tickets and using these to order the offer-set for presentation. Our evaluations compare this ordering with the user's selection of a preferred ticket and generate a recommendation accuracy (A_{rec}) using this equation:

$$A_{rec} \quad = \quad 1 - \frac{index - 1}{N - 1} \tag{1}$$

where *index* is the position of the selected offer in the ordered offer-set and N is the size of the offer set. In this way, each session a user has with Ticketyboo

that leads to the purchase of a ticket can be used to generate a recommendation accuracy score. By averaging recommendation scores, we can compare the utility of the context-free recommendation strategy with that of the context-informed recommendation strategy.

4.2 Finding and Learning Context Features for CBR

The domain knowledge required to correctly identify context data to be used in a CBR system can be expensive. However, this context data should be inexpensive to acquire, and with the growth of pervasive computing will become more available. One approach to reducing the work needed to identify suitable context features might be to amass a dataset of system behaviour in parallel with whatever context data was available during interactions with the system. This dataset could then be data-mined to look for correlations between context data and the accuracy of the CBR process. These context features could then be added to the case structure.

The relative importance of context features used in a CBR system should also be investigated. Feature weights could be learned using techniques similar to those of Stahl and Gabel [25], Branting [26] and Coyle et al. [27]. By examining context features that are deemed important, it may also be possible to direct the application designer to investigate other, similar sources of context data.

4.3 Applying Context to Existing Recommender Systems

There are many recommender systems that could benefit from context data similar to that which we incorporate. We describe ways that popular recommender systems might be improved with a little extra context data.

PTV is a personalised TV listings service which recommends TV programs to users based on their learned viewing preferences [28]. PTV could use calendar information in a similar manner as in Ticketyboo, whereby recommendations would not be made if the user's calendar suggested that they were away or busy.

In Section 2 we introduced Watson, the browsing assistant that uses information regarding the users activity on the desktop to infer context and thus assist the user to browse the Web. If sensors that can monitor desktop search applications (e.g. Google Desktop or Spotlight) were developed, applications such as Watson and Letizia could be improved greatly. This is an example of an existing application being improved with the development of new (or simply improved) context sensors.

E-commerce websites like Amazon could benefit from using additional context data, e.g. music preferences from customers listening habits into their recommendations, as a means of improving the relevance of the recommendations that it makes to its users. In fact Google Ads that are embedded into GMail do just this, Google analyses the content of user's email to determine context and selects the best types of advertisements to present to that user. Obviously, privacy is an issue in applications where context data might be sensitive and this issue is currently being addressed in the pervasive community [29].

Entree could be improved by making on-the-go recommendations more relevant. Tweaks that take into account context data, e.g. "show me a restaurant close to me now". This is simply a gelling of location data – commonly regarded as one of the most important components of context data in the pervasive computing domain – with the regular Entree recommender system.

4.4 Using Context for Decision Support

Up to this point we have talked about using context data to improve CBR in general and case-based recommendation in particular. One point we would like to emphasise is that with access to the quantity of context data available when using a context-infrastructure like Construct, it becomes relatively easy to provide whatever additional data might be useful to the user to make a decision when selecting an item for purchase (or indeed in any decision-support system). The difficulty arises in deciding which data to present to a user, and how to present it.

5 Conclusions

Many CBR and recommender systems have shown how context data can be useful in improving performance. However to date, this context data has been limited and sparse. Meanwhile, the pervasive computing community has much experience in dealing with context data. We propose that CBR systems designers should exploit the context information that is made available by work in the pervasive computing domain. Context-infrastructures like Construct offer significant advantages to recommender systems. By providing a middleware capable of collecting, distributing and making available heterogeneous context data we allow applications to benefit from the development of new sources of context data.

An advantage of the approach implemented is that all recommendation data are re-inserted into Construct, and thus available to other applications. For this reason, other user-interfaces could easily be developed and the choice of web-interface was just one of many possible solutions, e.g. a calendar plug-in that updates your calendar with recommended concert details; or a music plug-in that presents ticket information for the artist that the user is currently listening to.

Acknowledgements

The authors would like to acknowledge Conor Hayes and Pádraig Cunningham for the invaluable feedback and suggestions they provided for this paper.

References

1. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40**(3) (1997) 56–58

2. Aha, D.W., Breslow, L., Muñoz-Avila, H.: Conversational case-based reasoning. Applied Intelligence **14**(1) (2001) 9–32
3. Dobson, S., Coyle, L., Nixon, P.: Hybridising events and knowledge as a basis for building autonomic systems. Journal of Trusted and Autonomic Computing (2006) To Appear.
4. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The Active Badge Location System. Technical Report 92.1, Olivetti Research Ltd., ORL, 24a Trumpington Street, Cambridge CB2 1QA (1992)
5. Abowd, G., Atkeson, C., Hong, J., Long, S., Kooper, R., Pinkerton, M.: Cyberguide: A Mobile Context-Aware Tour Guide (1997)
6. Hong, J.I.: The Context Fabric. (http://guir.berkeley.edu/projects/cfabric/)
7. Heer, J., Newberger, A., Beckmann, C., Hong, J.I.: liquid: Context-Aware Distributed Queries. In: 5th International Conference in Ubiqutous Computing (UbiComp), Seattle, WA, USA (2003) 140–148
8. Judd, G., Steenkiste, P.: Providing Contextual Information to Pervasive Computing Applications. In: the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), Fort Worth, Texas (2003) 133–142
9. DiMicco, J., Lakshmipathy, V., Fiore, A.: Conductive chat: Instant messaging with a skin conductivity channel. Poster Presentation in ACM Conference on Computer Supported Cooperative Work, ACM Press (2002)
10. Sequeira, M., Bowden, M., Minogue, E., Diamond, D.: Towards autonomous environmental monitoring systems. Talanta (2002)
11. Dunne, L.E., Brady, S., Tynan, R., Lau, K., Smyth, B., Diamond, D., O'Hare, G.M.P.: Garment-based body sensing using foam sensors. In Piekarski, W., ed.: Seventh Australasian User Interface Conference (AUIC2006). Volume 50 of CRPIT., Hobart, Australia, ACS (2006) 165–171
12. Burke, R.D., Hammond, K.J., Young, B.C.: The findme approach to assisted browsing. IEEE Expert **12**(4) (1997) 32–40
13. Balfe, E., Smyth, B.: Case Based Collaborative Web Search. In: Proceedings of the 7th European Conference on Cased Based Reasoning. (2004) 489–503
14. Budzik, J., Hammond, K.J.: User Interactions with Everyday Applications as Context for Just-In-Time Information Access. In: IUI '00: Proceedings of the 5th International Conference on Intelligent User Interfaces, New York, NY, USA, ACM Press (2000) 44–51
15. Lieberman, H.: Letizia: An agent that assists web browsing. In Mellish, C.S., ed.: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95), Montreal, Quebec, Canada, Morgan Kaufmann publishers Inc.: San Mateo, CA, USA (1995) 924–929
16. Hayes, C., Cunningham, P.: Smart radio - community based music radio. Knowl.-Based Syst. **14**(3–4) (2001) 197–201
17. Hayes, C., Cunningham, P.: Context boosting collaborative recommendations. Knowledge-Based Systems **17**(2-4) (2004) 131–138
18. Stevenson, G., Coyle, L., Neely, S., Dobson, S., Nixon, P.: Construct — a decentralised context infrastructure for ubiquitous computing environments. In: IT&T Annual Conference, Cork Institute of Technology, Ireland. (2005)
19. McBride, B.: Jena: Implementing the RDF model and syntax specification. In: Proceedings of the 2nd International Workshop on the Semantic Web, Hong Kong (2001)

20. Chen, G., Li, M., Kotz, D.: Design and Implementation of a Large-Scale Context Fusion Network. In: The First Annual International Conference on Mobile and Ubiquitous Systems (MobiQuitous 2004), Boston, MA (2004)
21. Coyle, L., Neely, S., Rey, G., Stevenson, G., Sullivan, M., Dobson, S., Nixon, P.: Sensor fusion-based middleware for assisted living. In: Proc. of 1st International Conference On Smart homes & heath Telematics (ICOST'2006) "Smart Homes and Beyond", IOS Press (2006) 281–288
22. Gentner, D., Forbus, K.D.: MAC/FAC: a model of similarity-based retrieval. In: Proceedings of the 13th Cognitive Science Conference, Chicago, Erlbaum, Hillsdale (1991) 504–509
23. Hayes, C., Massa, P., Avesani, P., Cunningham, P.: An on-line evaluation framework for recommender systems. In: In Workshop on Recommendation and Personalization Systems, AH 2002, Malaga, Spain, 2002., Springer Verlag. (2002)
24. Coyle, L.: Making Personalised Flight Recommendations using Implicit Feedback. PhD thesis, University of Dublin, Trinity College (2004)
25. Stahl, A.: Learning feature weights from case order feedback. In Aha, D.W., Watson, I., eds.: Case-Based Reasoning Research and Development, 4th International Conference on Case-Based Reasoning, ICCBR 2001, Vancouver, BC, Canada, July 30 - August 2, 2001, Proceedings, Springer (2001) 502–516
26. Branting, L.: Learning feature weights from customer return-set selections (2003)
27. Coyle, L., Cunningham, P.: Improving recommendation ranking by learning personal feature weights. In Calero, P.A.G., Funk, P., eds.: Advances in Case-Based Reasoning, 7th European Conference, ECCBR 2004 Madrid, Spain, August 30th through Sep 2nd, 2004, Proceedings, Springer (2004) 560–572
28. Smyth, B., Cotter, P.: Surfing the digital wave: Generating personalised tv listings using collaborative, case-based recommendation. In Althoff, K.D., Bergmann, R., Branting, L.K., eds.: Case-Based Reasoning Research and Development: Third International Conference on Case-Based Reasoning, ICCBR-99, Seeon Monastery, Germany, July 1999. Proceedings, Springer-Verlag Heidelberg (1999) 561–571
29. Nixon, P., Wagealla, W., English, C., Terzis, S.: Smart Environments: Technology, Protocols and Applications. In: Privacy, Security, and Trust Issues in Smart Environments. Wiley (2004) 220–240