Context-Aware Content Filtering & Presentation for Pervasive & Mobile Information Systems

Kaijian Xu*
School of Computer Engineering
Nanyang Technological University
Nanyang Avenue, Singapore 639798
xuka0001@ntu.edu.sg

Manli Zhu
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
mlzhu@i2r.a-star.edu.sg

Daqing Zhang
GET/INT Institut National des Télécommunications
9 rue Charles Fourier, 91011 Evry Cedex, France
daqing.zhang@int-edu.eu

Tao Gu
Institute for Infocomm Research
21 Heng Mui Keng Terrace, Singapore 119613
tgu@i2r.a-star.edu.sg

ABSTRACT

What constitutes relevant information to an individual may vary widely under different contexts. However, previous work on pervasive information systems has mostly focused on context-aware delivery of application-specific information. Such systems are only able to operate within narrow application domains and cannot be generalized to handle other heterogeneous types of information. To fill this gap, we propose a context-aware system for information integration that can handle arbitrary information types and determine their relevance to the user's current context. In contrast to existing model-based approaches to context reasoning, we log user interaction and perform usage mining using OLAP to discover context-dependent preferences for different information types. This allows us to build a more generic and adaptive system that automatically selects the most relevant content and presents it to the user in a succinct manner that supports ease of consumption and comprehension.

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*user-centered design*

General Terms

context awareness, ambient intelligence, content integration, filtering, presentation, adaptivity, OLAP, preference mining

1. INTRODUCTION

A key area in ubiquitous computing is the design of pervasive and mobile information systems which embed information technology into the physical environment, delivering relevant information on demand to users through ubiquitous and connected computing devices. Such systems aim to support, augment and enrich daily experiences by supplying information in a context-aware manner.

One of the most crucial considerations in content provision for such systems is that people have different information needs depending on their current context. Consider an archetypal user named Alice, living in a smart home. Installed in her living room is a wall-mounted display panel on which she can summon up useful information for quick reference. As she leaves for work in the morning, she may check the weather forecast and her schedule for the day. However, such information would be much less relevant to her when she is relaxing after work in the evening. Instead, she might want to refer to the television listings for interesting programs to watch. Clearly, what constitutes useful content to Alice varies according to the situation, reflecting her contextual preferences.

Existing work on pervasive information systems has largely focused on providing information within very specific application domains, such as tour guides [6, 8, 9, 17, 19, 28], alerts and reminders [5, 11, 18] and location-based search [21, 32] and recommendation systems [22, 27]. Although these systems serve their intended purpose well, they are limited in scope and can only handle information types specific to their particular application.

User information needs in the real world are much more varied, however, and could include a wide range of different content, such as weather, news, stock and financial data, personal schedule, television listings, movie show times, RSS feeds, etc. Clearly, it would be infeasible to build a separate system to provide of each type of information; a single unified system that could integrate and filter such a variety of heterogeneous information types based on their relevance to the user's context would be the ideal solution. Yet, not enough attention has been given to solving this problem.

Besides selecting the right type of content for the context, it is also important to present it in an appropriate format that

*This work was carried out while the author was on internship at the Institute for Infocomm Research.

AMBI-SYS 2008, February 11-13, Quebec, Canada

DOI 10.4108/ICST.AMBISYS2008.2907

makes it easy to comprehend. This is especially so in ambient and mobile computing environments, where we need to effectively deliver the required information without overburdening the user. In Weiser's treatise on calm computing, pervasive systems are envisioned as able to smoothly and easily transition between the center of our attention and the periphery, increasing our knowledge without causing information overload [29]. In a pervasive information system, the user should be able to conveniently bring up the information they require, quickly peruse it, and return to what they were originally doing. The challenge here is to condense information succinctly – providing enough, but not too much.

These are the two key problems that must be solved in order to develop a unified, comprehensive pervasive information system. There are also a few lesser associated issues. First, it is a natural extension of the first requirement that such a system should be generic enough to handle arbitrary content types in order not to be bound to any single application area. We also observe that user preferences are not static, but may change over time. As such, the system needs to be adaptive in order to accommodate the evolution of user contextual preferences over time. We further note that the ability to support arbitrary context parameters will help enable the system to be used for a wider variety of applications. Most present context-aware systems generally deal with only few and fixed types of context information, or *context parameters*, like time, day of week, device, location and the presence of other people. However, many more potentially useful possibilities exist, such as sensor data like temperature and ambient light and sound levels, or output obtained from other context-reasoning systems such as user activity recognition or mood detection.

In this paper, we propose a method for content selection and presentation that meets all the abovementioned requirements. Support for multiple arbitrary information types is achieved by taking a modular approach, encapsulating each information type in its own *widget*. By tracking user interaction with the system in a database and applying online analytical processing (OLAP) techniques, our system is capable of self-learning the optimal selection and presentation of content for a given user in a given context.

The rest of this paper is organized as follows. In Section 2, we briefly review previous work in user modeling, context-awareness and personalized content filtering and presentation. We introduce our proposed technique and outline its implementation with a demonstrative prototype in Section 3. A discussion of implementation issues and other considerations is given in Section 4. Future directions for research are explored in Section 5. Finally, we conclude with a summary of our contributions.

2. PREVIOUS WORK

To provide a background, we will first survey existing work in user modeling, context-awareness and content personalization to examine how they tackle problems similar to those we have identified earlier.

2.1 User Modeling

Many different techniques for modeling and mining user preferences exist in the literature, especially in Web and E-commerce applications. Since most of it is concerned with content preferences rather than contextual preferences, we will only briefly characterize some of the techniques here; for a more in-depth introduction, the survey by Eirinaki and Vazirgiannis [12] on Web personalization is recommended.

2.1.1 Rule-Based

Rule-based techniques build deep models of user preferences and content characteristics, and relate preferences to content through a set of rules. For example, given demographic information about customers who have and have not purchased a particular product in the past, we can build a decision tree to classify whether a new customer is likely to buy the product or not. The decision tree thus serves as a model of customer preference towards the product based on customer demographics.

2.1.2 Co-occurrence-Based

Co-occurrence-based methods such as association rules [2, 14] and sequential patterns [3] revolve around detecting repeated patterns in historic data, such as different products that tend to be purchased together, or recurring navigation trails of different visitors to the same Web site. The discovered patterns can then be applied to predict the preferences of new users.

2.1.3 Content-Based

Another technique is to characterize the sort of content that users are interested in, and then use traditional information retrieval techniques to find relevant content. Such content-based methods are most readily applied in text-based contexts such as Web pages and news articles. A common example is using TF-IDF [24] vectors to represent text content and user preferences. We can also leverage semantics to characterize both users and content using ontologies. Nearest neighbor algorithms [10] can then be applied find content that is similar to the user's interests.

2.1.4 Collaborative Filtering

Collaborative filtering collects opinions or ratings from many users in order to provide recommendations. Individual algorithms may vary, but the underlying assumption is always the same - that users with similar interests will agree with each other in their ratings. Without building an explicit model of user preferences, collaborative filtering suggests items or content based on user similarity. For more on collaborative filtering, we suggest referring to the survey by Adomavicius and Tuzhilin [1].

2.2 Personalized Information-at-a-Glance

The overwhelming amount of information that the World Wide Web offers gave rise to the need to help users filter and organize information, and to present it in a manner that is convenient and easy to digest. This led to the development of Web portals (e.g. iGoogle, MSN, My Yahoo!, Netvibes), which consolidate and integrate various types information, (e.g. news, weather, financial updates, personal schedule, notes) to create a customized browser start page providing "information-at-a-glance". However, most existing Web portals are not adaptive nor context-sensitive, and remain heavily dependent on user configuration.

Some effort has gone into developing information-at-a-glance applications that are more intelligent and adaptive. Anderson and Horvitz demonstrated MONTAGE [4], a personalized start page for Web browsers that automatically clips content from different Web pages and aggregates it into a single page. It achieves this by learning some user preferences which it uses to compute the expected utility of available content. The final layout of the page is obtained by solving a knapsack problem.

An early foray into personalized information-at-a-glance in pervasive computing by Zhu et. al. [31] uses proximity sensors and RFID tags to detect and identify users who step within range of a wall-mounted display panel. Content extracted from various sources are selected according to the user's preferences, summarized, and presented according to predefined layout templates.

2.3 Context-Aware Content Provision

Most existing pervasive computing applications that support context-aware content provision follow the paradigm illustrated in Figure 1. Sensors are used to collect *context parameters*, which are variables relevant to determining the current context, such as location and time. These are used for context reasoning based on predefined or learned context models. Content filtering is then performed based on the inferred context. Due to space constraints, we will only highlight a few examples to illustrate this.

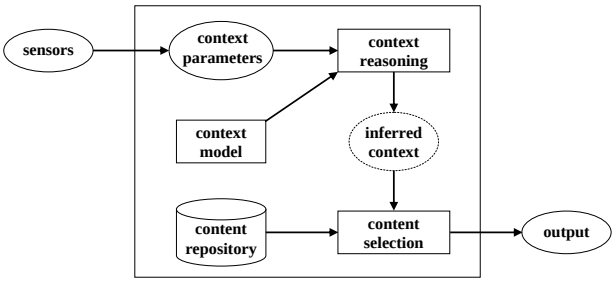


Figure 1: Model-based context reasoning paradigm for content filtering.

This paradigm is most evident in the "tour guide" genre of applications. For instance, the Museum Tour Guide [9] of Chou et. al. uses rule-based methods to build a tour and give directions to museum visitors based on their preferences. It is able to adjust the tour as it progresses based on context, such as whether the user is ahead of time or lagging behind.

Another example is context-aware information provision to the mobile phone standby screen, proposed by Nakatsuru et. al. [20]. They demonstrated a system which allowed users to make word-of-mouth recommendations of shops and restaurants in a peer-to-peer fashion. The recommendation content is stored away upon receipt; only when the recipient's context matches the target context is it displayed on the mobile phone standby screen.

Yu et. al. described a framework for context-aware media personalization in [30] which performs context acquisition, reasoning and learning to build an ontology-based context model for in content filtering and recommendation.

2.4 Context as OLAP dimensions

Since our proposed system will employ OLAP to handle context, we review previous work in this area as well. The concept of treating context parameters as dimensions in OLAP was pioneered by Stefanidis et. al. in their work on the Context-Aware Preference Database System [25, 26] that stores explicit context-dependent preferences in a database as scalar values. The database stores only *basic preferences*, which contain only one context parameter, some non-context parameters, and the user-defined preference, or "degree of interest." More complex preferences involving more than one context parameter need to be computed as a weighted sum of basic preferences. Data cubes are used to store associations between preferences and database relations, and OLAP is employed to process context-aware queries at different levels of abstraction.

3. OUR APPROACH

We will now describe our proposed system in detail.

3.1 Modular Approach for Information Types

Recall that we have identified two major needs in pervasive and mobile information systems: filtering heterogeneous information types according to their relevance to the user's context, and condensing information in a succinct manner.

The "information-at-a-glance" approach of condensing and integrating content (Section 2.2) fits our needs quite well. It encapsulates each type of information within its content block, or *widget*, allowing a modular approach to display layout. The concept of selecting and aggregating content based on predicted utility, as demonstrated in MONTAGE [4], can also be employed. Borrowing from these ideas, we only need to devise a means of predicting the utility of content based on context in order to meet the first requirement.

To tackle the second requirement, we observe that amount of content as perceived by a user depends on two dimensions - *quantity*, that is to say, the number of distinct items, and *granularity*, or the level of detail. For example, if we were to consider news content, quantity would refer to the number of stories being displayed, whereas granularity determines whether news items are displayed as headlines only, with a brief summary, or as a full story. Therefore, we design each widget to support control over how information is presented through *presentation settings*, given by the two parameters quantity and granularity. With this, we simply have to come up with a means of determining the most appropriate presentation setting depending on the context.

This modular approach is well suited for supporting a wide range of different content since it defines a protocol for controlling the presentation format for content independently of information type; it also enables adaptive layout for display on a variety of devices with different screen sizes. Any arbitrary content can be incorporated, and third-party content provision is easily supported.

3.2 Devices and User Interface

We design the system to be usable with any device with a display and simple selection controls, including desktop and laptop computers, PDAs, mobile phones and touch screen

displays. Devices with attached sensors for collecting relevant context parameters would be advantageous, e.g. mobile devices with built-in GPS to supply location. To access the system, devices connect to a central server and issue a request containing the user and current context parameters. The server will perform automatic selection and layout of widgets and respond with content presented in the "information-at-a-glance" format described earlier.

However, the information desired by the user may occasionally be omitted, either because the system has not yet fully learnt the correct preferences, or because of evolving user preferences. To handle such cases, we include a button that brings up a "start menu" list of all available widgets for the user to choose from. Similarly, each widget is designed to include menus with which the user can fine tune its presentation settings. Although having to issue such explicit commands may be a slight hassle, it will only happen infrequently. Furthermore, users have the incentive to use the commands in order to a) obtain the information they want, and b) help the system to register their preferences correctly.

The system is designed to operate primarily on a "pull" basis, activated by explicit user access such as firing up an application on a computer or approaching a wall-mounted display like in [31]. It deactivates once the application is closed or the user walks away from the display. Although the interface supports interaction through the start menu and widget presentation controls, their use would be minimal once the system has learnt the user's preferences.

Besides serving the center of our attention when demanded, the system can also function as a peripheral display [29]. Possible examples include mobile phone standby screens, desktop docklets and wall-mounted screens serving as ambient displays when no users are in range. Unlike the interactive mode where multiple widgets may be included, the peripheral mode should be less cluttered; one possibility is to gradually cycle through the most relevant widgets one by one instead of displaying everything at the same time.

3.3 System Architecture

An overview of how our system works is given in Figure 2. Unlike the current paradigm of context-aware content provision in Figure 1, we do not perform any context reasoning. Instead, we rely on co-occurrence analysis between content and context parameters from usage logs to discover preferences for certain content under various (latent) contexts.

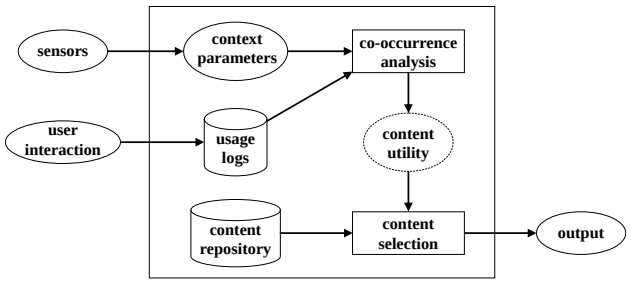


Figure 2: Proposed basic system architecture.

3.3.1 Co-Occurrence Analysis

We will begin by explaining the intuition behind using co-occurrence analysis and how it works.

In pervasive and mobile information systems, the objective is to support the user in their daily activities, allowing them to summon relevant information as and when desired. We observe that such information needs are *episodic*, that is to say, they are recurring, and there is a correspondence between the information required and the context in which the user requests it. We can leverage this by studying how often the various information types and contexts occur together. Intuitively, the greater the frequency of co-occurrence, the more closely related the information and context.

To illustrate, we join Alice on her weekly shopping trip to the supermarket on a Saturday morning. When she accesses the system using a PDA, analysis of past usage would likely reveal that the `shopping list` widget is accessed much more frequently than other types of information in previous situations with context parameters (i.e. device, day of week, time, location) that closely or fully match the present ones. This implies that `shopping list` is most useful in this situation, so we assign a higher content utility score to it compared to other information types. It will then be favored by the content selection step. Notice that context of 'grocery shopping' does not need to be explicitly defined or identified.

Besides the type of information, we can also determine the quantity and granularity of content favored by the user under different contexts. For example, Alice might access the system for news in two different situations - on a mobile phone while commuting to work, and from the office computer later in the day. If she prefers a headlines-only format on the mobile phone and full stories on the desktop computer, we will be able to discover this using the same analysis as above, and assign a higher utility to the headlines-only format in the former case, and vice versa.

Hence, given the user and context parameters, we can calculate how useful each widget is to the user for each possible presentation setting.

3.3.2 Content Selection and Presentation

With the utility scores obtained from co-occurrence analysis, we can choose to employ different content selection strategies depending on our needs.

The simplest way is to rank the information types according to their utility and choose the top-k items for display. Widgets would then be displayed from top to bottom in decreasing order of relevance, minimizing the need for scrolling. This strategy is suitable for mobile phones, for instance, since their small screens do not support fancy layouts.

For more complex layouts or where scrolling is not possible, knapsack packing algorithms [23] can employed. Knowing the predicted utility and screen space needed by each widget under each setting, knapsack algorithms can select the set of widgets that will provide the greatest total utility to the user while maximizing use of available screen space, thereby optimizing the selection and presentation of the content that we display to the user.

3.3.3 Usage Logging

To support the co-occurrence analysis, we need to keep track of past usage in a log. Based on what we have described in Section 3.3.1, it would be most convenient if each log entry corresponds to a single access to a single widget, since we could then simply select the entries with similar and matching context parameters and count them. As such, we design each log entry to store the user, various context parameters, the widget and its presentation settings, quantity and granularity. If we lack information about any of the fields or if they are not relevant, we simply store a `null`.

As for when log entries are created, we define two cases - explicit commands and implicit feedback. In the first case, we assume that when a user accesses the system, they are seeking information. If the information they want is not shown, or is presented in the wrong format, the user issues commands to rectify the situation. Since this represents an explicit expression of contextual preference for a certain information type or presentation format, we create a log entry with the user ID, current context parameters, widget ID, and where applicable, the presentation settings specified.

However, once the system has learnt the user's contextual preferences, the information they seek will be automatically displayed, and the user will dismiss the system once they have obtained the information they want, without issuing any explicit commands. We can detect such instances and treat them as implicit feedback that the desired information was correctly selected and displayed in the right format.

3.4 Co-Occurrence Analysis with OLAP

We will now discuss specific details of how co-occurrence analysis can be performed on the usage logs following the OLAP paradigm. Unlike transactional databases that serve day-to-day operations, OLAP is designed to assist "knowledge workers in the role of data analysis or decision making" [13], and are most widely employed in business intelligence applications to support marketing decisions.

Using the log as defined in Section 3.3.3, we can formulate a query to find entries that match the current context parameters and count them. But relatively few entries will match the current context exactly. As such, we also need to look for entries that are *contextually similar*. Intuitively, we could start looking for entries that match all but one of the context parameters, and then all-but-two, etc. Also, instead of naively counting all entries, we should perform weighting according to contextual similarity so that a closer match to the current context counts as more.

However, given more than a few context parameters in the logs, this approach will require an inordinately large number of queries to service one request. This is untenable since we expect the system to respond quickly to user requests. To alleviate this problem, we can employ OLAP to precompute and cache the row counts that we need. The application of OLAP here is straightforward - we treat the user and context parameters as OLAP dimensions describing a hypercube. For any point in the cube, we are interested in the number of rows in the original table that satisfy the conditions given by that point. We thus define measures to count the number of instances of each widget, as well as each presentation setting of each widget. The dimensions and measures are shown in the fact table of our schema in Figure 3. Since `count()` is a distributive aggregation function, it allows for highly efficient computation of measures in materialization of cuboids.

OLAP also allows us to define concept hierarchies that allow us to map low-level concepts such as raw timestamp to higher-level, general concepts such as the hour-of-day and day-of-week. The star schema in Figure 3 shows sample dimension tables for `user`, `device`, `day`, `time` and `location`. Using concept hierarchies, we can detect richer patterns by matching at various concept levels. For instance, we can now identify contextual preferences relating to "Saturday morning" or "weekday evenings", which we could not do by simple matching on raw timestamps. By defining a concept hierarchy on users based on demographics, we can also mine for preferences that are shared across people of similar age groups, occupation, etc.

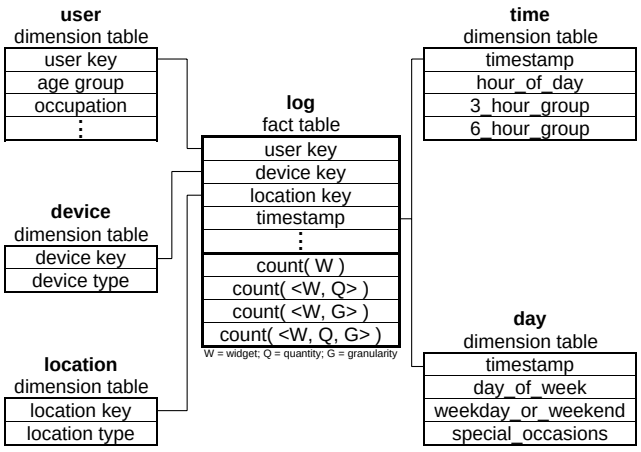


Figure 3: Star schema for OLAP usage mining.

As a further improvement, we can discard old log entries by setting an expiry. By considering only recent entries, the system will be able to handle evolving user preferences. Another possibility is to introduce weighting according to how recent a log entry is, so that the influence of an entry gradually diminishes as it becomes older. By giving higher priority to recent preferences, the system can adapt even faster to preferences changes.

Using OLAP for co-occurrence analysis means that we are not bound to any particular context model, and can incorporate arbitrary context parameters without having to change the system significantly. It is also not bound to any particular type of content, and allows us to arbitrarily add or remove any widget from the system. This is especially appealing from the user's point of view since all sorts of custom and third-party content can be supported easily.

Figure 4 visually summarizes the complete architecture of our context-aware content filtering and presentation system.

3.5 Implementation and Deployment

Our demonstrative prototype runs as a Web application on the Apache Tomcat servlet container. It is built on the

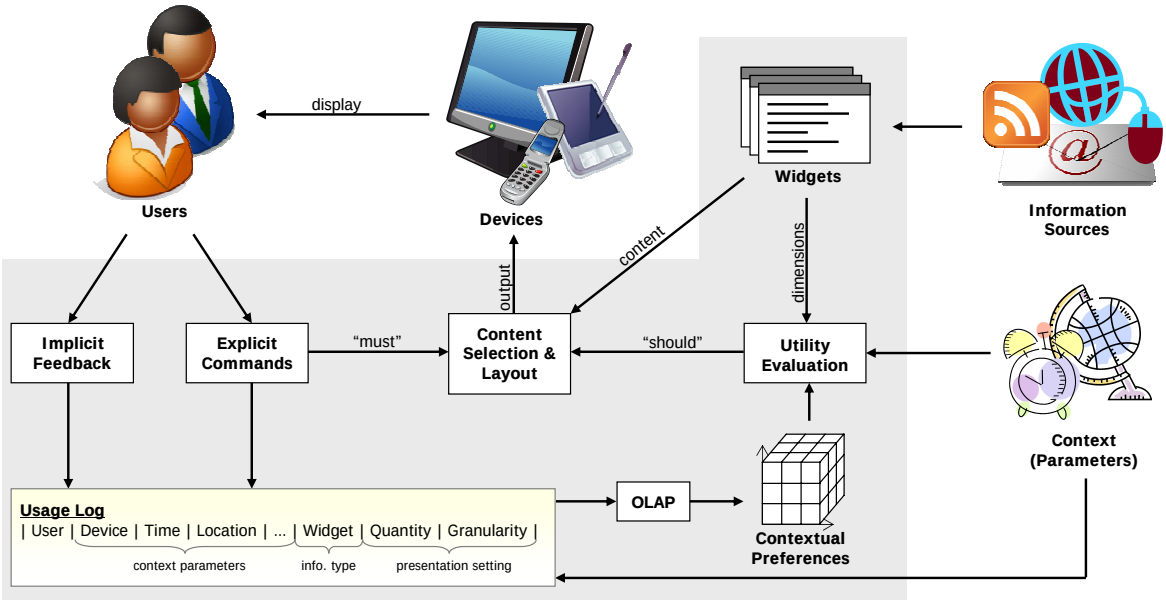


Figure 4: Complete system architecture.

Apache Tapestry framework[1], with a MySQL database for user interaction logging and Mondrain OLAP server[2] for analysis. Our setup places the server in Alice's smart home setting, serving a single household. Alice has many options for accessing the system - from computers at home and at work through a normal Web browser, on GPRS- and WLAN-enabled mobile phones and PDAs, and on the wall-mounted touch screen display we mentioned earlier. The touch screen is powered by a computer that simply displays a Web browser in full-screen mode.

For OLAP analysis, we use the star schema illustrated in Figure 3, with device types falling under {phone, PDA, laptop, desktop, wall display} and location types {home, work, shopping, friend, transport, other}. The time dimension is grouped by hour, 3-hr groups {12mn-3am, 3am-6am, ...} and 6-hr groups {morning, afternoon, evening, night}. The context hierarchy for users is not used. Since the setup is only intended to serve as proof of concept, we have not fully optimized the OLAP server for speed or storage efficiency.

A demonstration of the system in operation is given in Figure 5. Figure 5(a) shows a screenshot taken from the wall-mounted display as Alice relaxes at home on a Saturday evening. Television listings from her favorite cable channels are displayed to help her choose what to watch later tonight. The weather outlook helps her to plan activities for the next few days. Since Alice is moderately superstitious, her horoscope is also displayed.

Figure 5(b) shows Alice accessing the system from a mobile phone while commuting to work. Compared to the previous example, the information shown here is of a more serious strain. She is reminded of important meetings on her schedule for the day, and a selection of news headlines keeps her up to date on current events.

The system could potentially be operated on a larger scale, running from a shared public server serving many people, but such a large deployment may encounter scalability problems. Furthermore, such a setup may raise questions about user privacy. These issues are discussed in the next section.

4. DISCUSSION

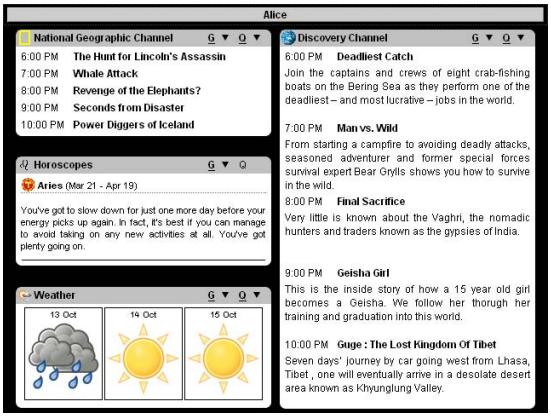
Having described our system, we will now compare it to other existing methods, discuss potential implementation issues and other considerations.

4.1 Comparison with Context Reasoning

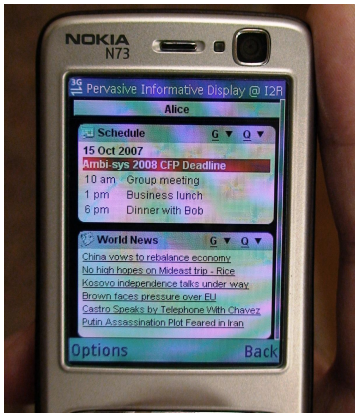
Since the most significant difference between our proposed approach and that of other existing context-aware content provision applications is in the use of co-occurrence analysis instead of context reasoning; a discussion of their relative merits is warranted.

When context reasoning is used for content provision, it requires a context model to be built, based on a fixed set of context parameters and supporting a limited range of contexts - predefined or learned. This approach does not generalize well since different applications require support for different contexts and parameters, leading to different context models. Furthermore, human effort and intervention is required to predefine the contexts and in annotation where supervised learning is used. In contrast, co-occurrence analysis is able to support a wide range of contexts, constrained only by the context parameters used. Contexts are treated as latents in the system and do not need to be predefined. Nonsensical combinations of context parameters are not a concern either; since we log actual user interactions, only meaningful values will be recorded.

[1]http://tapestry.apache.org/
[2]http://mondrian.pentaho.org/



(a) Screenshot from wall-mounted display.



(b) Accessing the system from a mobile phone.

Figure 5: Demonstration of prototype system.

A shortcoming of co-occurrence analysis is its reliance on historical log data. This results in situations where new users to the system will have to go through a period in which their preferences cannot be correctly determined because there is not yet enough data in the logs. Fortunately, this problem can be alleviated using having default preferences based on unpersonalized co-occurrence analysis. By defaulting to generic preferences, the system will be able to learn personalized preferences for new users within a shorter time compared to if it had started off with no preferences at all.

4.2 Comparison with Rule Mining

Our approach is related to the application of association rule mining (ARM) by Kammanahalli et. al. on past usage to determine users' contextual preferences with respect to presentation format and medium in a Web-based collaboration tool [15]. Both methods rely on the association between context parameters and preferred settings.

However, application of ARM is plagued by the difficulty of selecting an appropriate minimum support, or *minsupp*. Too high a *minsupp* will result in many patterns being missed and lead to poor personalization, whereas too low a setting will result in an oversized ruleset. Also, since it is an offline technique, rules must be recomputed periodically in order to accomodate preference changes. On the other hand, our OLAP hypercube efficiently summarizes all the associations that can be derived from the log data, and dynamically adjusts to reflect shifting preferences.

Furthermore, OLAP is inherently designed to handle multiple levels of abstraction, allowing us to use concept hierarchies to finding preferences at different levels. Multi-level ARM could also accomplish this, but it is more complex than simple ARM and has substantially higher storage and computational costs.

4.3 Content Personalization

Clearly, content personalization using methods such as those discussed in Section 2.1 can be applied independently for each widget to further improve the user experience. However, we do not devote too much attention to this since our focus is on contextual rather than content preferences.

4.4 Scalability

The use of OLAP for analysis leads to potential scalability problems - if we perform full materialization of the data cube, storage space required may explode given too many dimensions and multiple level concept hierarchies. As such, partial materialization is preferred. Other potential optimizations include computing only iceberg cubes [7].

4.5 Complexity

The knapsack problem is known to be NP-hard [16]; this may impact content selection and layout using the knapsack packing approach. However, exact solutions can be found within linear to polynomial time for most knapsack problems encountered in practice [23], and approximations can be obtained even faster. Since accuracy is not highly critical in our application, we may settle for a reasonable approximation so as to service user requests in a timely fashion.

4.6 Privacy

The tension between privacy and personalization is a difficult problem in user design. Since our system relies on logging user interactions, it may lead to privacy concerns. We believe these concerns are manageable through responsible implementation and deployment. For instance, our prototype is set up as a private system serving one household, rather than a public server shared by many users. As long as the server does not share any user information or preferences with external parties, it offers an acceptable level of privacy to the users. Also, since the server is physically located in the home, appropriate network security measures can be taken to help safeguard sensitive information.

5. FUTURE WORK

We would like to explore two main directions for future work on this system.

First, there is a need to evaluate the system for its speed, accuracy and adaptability. Based on this evaluation, we could then optimize the system to return more relevant results with shorter latency, and to learn and adapt to new and evolving preferences within a shorter time. Various weighting options and OLAP optimizations can be explored to improve accuracy, efficiency and scalability.

We are also interested in potential feature improvements, such as the ability to support multiple concurrent users at a single wall-mounted display. The potential of the system as a peripheral display can also be further explored. Perhaps widgets could have a special "peripheral mode" in which information is presented in a simple and intuitive graphical format rather than in text form.

6. CONCLUSION

The contributions of this work are as follows. We have proposed a method for integrating arbitrary information types that is capable of choosing between various types of information depending on their relevance to the current context. By characterizing the "amount of information" by means of two parameters, *quantity* and *granularity*, we are also able to determine the preferred format for presenting the information in the given context. All this is achieved by using OLAP techniques to perform co-occurrence analysis to determine contextual preferences from usage logs, as opposed to the current paradigm of context reasoning using models. The techniques we have developed enable us to build pervasive information systems that can effectively deliver a large variety of information to assist users in their daily lives.

7. REFERENCES

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE TKDE*, 17(6):734–749, 2005.

[2] R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proc. SIGMOD*, 1993.

[3] R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. ICDE*, 1995.

[4] C. R. Anderson and E. Horvitz. Web montage: A dynamic personalized start page. In *Proc. WWW*, 2002.

[5] M. Beigl. MemoClip: A location based remembrance applicance. In *Proc. HUC*, 2000.

[6] F. Bellotti, R. Berta, A. D. Gloria, and M. Margarone. User testing a hypermedia tour guide. *IEEE Pervasive Computing*, 1(2):33–41, 2002.

[7] K. S. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg CUBEs. In *Proc. SIGMOD*, 1999.

[8] K. Cheverst, N. Davies, K. Mitchell, and A. Friday. Experiences of developing and deploying a context-aware tourist guide: the GUIDE project. In *Proc. MobiCom*, 2000.

[9] S.-C. Chou, W.-T. Hsieh, F. L. Gandon, and N. M. Sadeh. Semantic web technologies for context-aware museum tour guide applications. In *Proc. WAMIS*, 2005.

[10] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE ToIT*, 13(1):21–27, 1967.

[11] A. K. Dey and G. D. Abowd. CybreMinder: A context-aware system for supporting reminders. In *Proc. HUC*, 2000.

[12] M. Eirinaki and M. Vazirgiannis. Web mining for Web personalization. *ACM TOIT*, 3(1):1–27, 2003.

[13] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers, 2000.

[14] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining — a general survey and comparison. *SIGKDD Explorations*, 2(1):58–64, 2000.

[15] H. Kammanahalli, S. Gopalan, S. Varadarajan, and K. Ramamritham. Context-aware retrieval in web-based collaborations. In *Proc. PERCOMW*, 2004.

[16] R. M. Karp. *Reducibility Among Combinatorial Problems*, pages 85–103. Plenum, New York, NY, USA, 1972.

[17] R. Kramer, M. Modsching, J. Schulze, and K. ten Hagen. Context-aware adaptation in a mobile tour guide. In *Proc. CONTEXT*, 2005.

[18] N. Marmasse and C. Schmandt. Location-aware information delivery with comMotion. In *Proc. HUC*, 2000.

[19] A. Maruyama, N. Shibata, Y. Murata, K. Yasumoto, and M. Ito. P-Tour: A personal navigation system for tourism. In *Proc. 11th ITS World Congress*, 2004.

[20] T. Nakatsuru, K. Murakami, and H. Sakai. Context-aware information provision to the mobile phone standby screen. In *Proc. MDM*, 2006.

[21] V. Naresh, P. Pingali, V. Varma, M. Krishna, and P. Venkata. Location based web search on GSM/GPRS mobile phones. In *Developers Track Proc. WWW*, 2006.

[22] M.-H. Park, J.-H. Hong, and S.-B. Cho. Location-based recommendation system using bayesian user's preference model in mobile devices. In *Proc. UIC*, 2007.

[23] D. Pisinger. *Algorithms for Knapsack Problems*. PhD thesis, Dept. of Computer Science, University of Copenhagen, 1995.

[24] G. Salton. *Automatic Text Processing*. Addison-Wesley Longman, Boston, MA, 1988.

[25] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Modeling and storing context-aware preferences. In *Proc. ADBIS*, 2006.

[26] K. Stefanidis, E. Pitoura, and P. Vassiliadis. Adding context to preferences. In *Proc. ICDE*, 2007.

[27] Y. Takeuchi and M. Sugimoto. CityVoyager: An outdoor recommendation system based on user location history. In *Proc. UIC*, 2006.

[28] M. J. Weal, D. T. Michaelides, D. E. Millard, D. C. D. Roure, and G. Fitzpatrick. Observations on pervasive information systems design. In *Proc. Workshop on Principles of Pervasive Information Systems Design*, 2007.

[29] M. Weiser and J. S. Brown. *The coming age of calm technolgy*, pages 75–85. Copernicus, New York, NY, USA, 1997.

[30] Z. Yu, X. Zhou, Z. Yu, D. Zhang, and C.-Y. Chin. An OSGI-based infrastructure for context-aware multimedia services. *IEEE Communications Magazine*, 44(10):136– 142, 2006.

[31] M. Zhu, D. Zhang, J. Zhang, and B. Y. Lim. Context-aware informative display. In *Proc. ICME*, 2007.

[32] Loki. Free location-based search and navigation toolbar. http://loki.com/.