

Preserving Privacy when Preference Searching in E-Commerce

Rhys Smith
School of Computer Science,
Cardiff University, Cardiff, CF24 3XF, UK.
R.O.Smith@cs.cardiff.ac.uk

Jianhua Shao
School of Computer Science,
Cardiff University, Cardiff, CF24 3XF, UK.
J.Shao@cs.cardiff.ac.uk

ABSTRACT

The idea of using user preferences to assist with information filtering and with providing the most “relevant” answers to queries has recently received some attention from the research community. This has resulted in the proposition of several frameworks for formulating preferences and their direct embedding into relational query languages. In this paper we discuss major exploitation issues and privacy concerns inherent in the basic paradigm used by these proposed approaches when used with e-businesses not implicitly trusted by the user. We then outline an alternative approach geared specifically towards using user preferences when interacting with e-businesses.

Categories and Subject Descriptors

K.4.1 [Public Policy Issues]: Privacy; H.3.5 [Online Information Systems]: Web-based services; H.3.4 [Systems and Software]: User profiles and alert services

General Terms

Algorithms, Security

Keywords

Privacy, User Preferences, E-Commerce, Exploitation

1. INTRODUCTION

The idea of capturing and using a user’s personal preferences by information systems has attracted some attention from the research community in recent times [1]. Knowledge of a user’s preferences can be utilized in the area of information filtering and retrieval, in building user profiles, and in formulating heuristics to improve decision-making algorithms. In this paper we specifically consider the use of user preferences in e-commerce applications.

As commerce conducted online becomes a steadily increasing area of revenue, one area of research in the field of e-

commerce is that of improving information filtering and retrieval techniques. If users are to be able to find items that match their preferences sufficiently in ever expanding catalogues and in increasingly competitive times, new search techniques that go beyond the standard query based paradigm are required. This is because conventional constraint based queries can produce the problems of *information overload* and *empty result sets*. *Information overload* occurs when a user defines too broad a search resulting in a large amount of matching items being returned, many of which will probably be irrelevant. *Empty result sets* occur when a user defines too specific a search resulting in no matching items being returned. In the context of e-commerce, both problems are equally undesirable from the perspective of both user and e-business - a user who could not locate a desired item quickly would likely get frustrated, leave, and buy the item from a different e-business after a few failed search attempts.

One technique proposed to improve information filtering and retrieval involves the use of user preferences. These can be used by taking conventional constraint based search queries (illustrated in Example 1) and extending them by adding information regarding the user’s personal preferences (illustrated in Example 2). This helps a system to focus search queries and order the results according to the user’s stated preferences.

EXAMPLE 1. Conventional queries are entirely constraint based, allowing a user to define a search (e.g. on a database of real estate for sale) such as:

Find all apartments located in London or Cardiff whose price is between £100,000 and £125,000.

This would return an unordered set of items whose attributes satisfy the constraints of the query.

EXAMPLE 2. Preference based queries add the user’s preferences to the conventional constraints, allowing a user to define a preference based search such as:

Find all apartments or flats (preferably apartments) located in London or Cardiff whose price is between £100,000 and £125,000 (the lower the better!).

This would return a set of items with respect to the user’s preferences, usually ordered by how preferred each item is.

1.1 The Current Approach

To realise the idea of including a user’s preferences into information filtering and retrieval techniques, several frameworks have been proposed in recent years for formulating preferences and their direct embedding into relational query

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WPES’03, October 30, 2003, Washington, DC, USA.
Copyright 2003 ACM 1-58113-776-1/03/0010 ...\$5.00.

languages [3, 5, 6, 8, 10, 11, 21]. All of the proposed frameworks share the same basic paradigm, whereby the evaluation of the users' preferences occurs on the e-business' server side - a user expresses a preference query (consisting of constraints and preferences) which is transmitted in full to a preference-enabled DBMS. The DBMS then endeavours to return to the user what it considers the "best matching" items apropos the query (Figure 1).

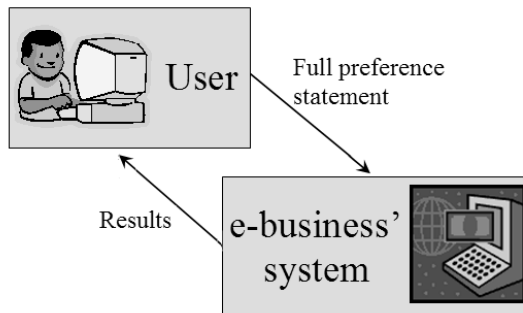


Figure 1: The Complete Release Paradigm

While this paradigm works well when applied in some situations, when it is specifically applied to e-commerce (and e-businesses not implicitly trusted by the user) there are two key problems inherent in the very fundamentals of the paradigm. The first is an issue regarding exploitation of user preferences, while the second is an issue regarding general privacy concerns. We now expound on these problems.

Exploitation of User Preferences

When users' preferences are sent *in full* to an e-business' system, an untrustworthy e-business can exploit the knowledge contained in this full preference statement. This could result in the user being returned a set of results that are not optimal for them, instead being optimal to the e-business.

To illustrate this problem, consider the following:

EXAMPLE 3. *Snappee Cameras is a company that sells digital cameras. Alice wants to buy a digital camera. She would like it to be made by either Nikon or Canon (preferably Nikon), costing between £175 and £300 (the lower the better).*

The company have in stock four cameras matching her constraints - one Nikon camera costing £199, and three Canon cameras costing between £215 and £285.

If the company were trustworthy, the search should retrieve the Nikon camera since that is the item that best matches her preferences. However, the company is not trustworthy, and have somewhat unscrupulously set up their system so their own preferences are added to, and supersede, the user's preferences.

Due to these extra imposed preferences, the user is presented with a result set consisting of only the three Canon cameras, as the company knows that these will still satisfy Alice's preferences, while at the same time maximising the company's profit and keeping the last Nikon camera in stock. Alice would consequently pay more for an item she desires less.

As this simple example illustrates, an unscrupulous e-business can design their system to exploit the knowledge

contained in a customer's full preferences, to the detriment of that customer. This goes a step beyond the well known practice of price discrimination (where the price is adjusted depending on what a customer is willing to pay) as instead of paying more for the same item, a user could pay more for an item they desired less. Thus, for a user to be able to obtain trustworthy results even from sources they do not trust, the current paradigm is not adequate.

Privacy Issues

There is a major conflict between user and e-business in the area of privacy - e-businesses are thirsty for ever-increasing amounts of personal information about their users, which undermines the users' fundamental "right to privacy".

When a user's preferences are sent in full to an e-business' system, the user actually releases a substantial amount of personal information to the e-business - information about their likes and dislikes. While this information can help the e-business to adapt their service to each customer's needs through personalisation, it is breaching the customer's right to privacy. A person has a moral "right" to be able to limit as much as possible the amount of personal preference information held about them, if they so wish. Also, this would limit other exploitation of the information that could occur in other ways than as in the previous example - for example, data mining can be performed on it.

Recent studies [2, 18, 22] have shown that although a large percentage of internet users are highly concerned about privacy, many of these privacy-concerned users disclose large amounts of information about themselves. It appears that users cannot be trusted to keep sensitive information about themselves private!

If there were a method available of retrieving optimal results for a user (according to their preferences) while only releasing a *part* of the full preference knowledge, there is no reason to release the full knowledge. In fact, doing so allows exploitation of the preferences and impacts on a user's privacy for no adequate reason. Thus, for a user to be able to keep as much as is possible of their personal preference information private when interacting with an e-business, while still retrieving the "best" results, the current paradigm is not adequate.

1.2 A New Approach

To address the concerns of exploitation and privacy when interacting with an e-business using user preferences, a new privacy enhancing approach is required. Such approaches are often termed PETs (Privacy Enhancing Technologies). We propose here a PET whereby rather than releasing all of the user's preferences to the DBMS together, we instead gradually release the preferences as necessary, putting the onus of selecting the most preferred items on the user end of the interaction rather than the e-business end. This maximises the user's privacy by minimising the preference knowledge transmitted, while preventing exploitation of the knowledge contained in the full preference statement. This approach is shown in Figure 2.

In this paper, we describe a system that implements this proposed new method, whereby an agent acting on behalf of the user is used to select the most preferred items from the untrusted remote system. The agent collects its user's preferences (by some means), analyses these preferences and separates them into elements. It then gradually releases

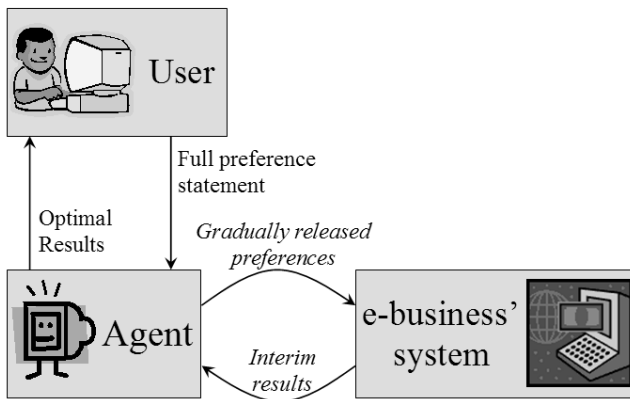


Figure 2: The New PET Approach

these elements to the remote DBMS, analysing the results returned, until the results were satisfactory or it ran out of preferences. The agent then presents the results to the user.

Transferring the job of selecting the most preferred items from the untrusted e-business' system to a trusted agent helps to maximize privacy and prevent exploitation of preference knowledge. However, there are some challenges associated with this transfer:

- How do we best model a user's preferences in order to split them up?
- How do we split up a user's preferences into elements in order to be gradually released?
- How do we gradually release the elements so that the "best" items are retrieved for the user, without allowing the preference knowledge to be exploited, while also maintaining as much privacy as possible?
- Is it actually possible to measure (in any meaningful way) the "amount" of privacy preserved, to help us quantitatively evaluate our approach?
- How can performance be optimised, and the trade-off between quality of results, privacy and performance be balanced?

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 defines the basic notations used throughout the paper. In Section 4 we analyse the structure of preferences and show how to use this knowledge to decompose them into parts. Section 5 describes our approach, showing our method of gradually release preferences. It discusses performance issues related to our approach, privacy, and gives an example. Finally, we conclude and discuss future work in Section 6.

2. RELATED WORK

There are three main areas of previous work relevant to the topic of this paper - that of user preferences in database environments, general agent technologies, and privacy.

Preferences in Databases

The work done in this area thus far is based upon the complete release paradigm, assuming that the user's preferences are to be sent in full to the remote system, which

will analyse the preferences and return the best matching results. This work on defining frameworks for formulating preferences and their direct embedding into relational query languages has received growing attention in recent times [3, 5, 6, 8, 10, 11, 21]. The work can be split into two main approaches - *quantitative* and *qualitative* based approaches.

The *quantitative* approach specifies preferences between tuples in an instance of a relation using *scoring functions*, whereby a numeric score is associated with each tuple of a query result set [3, 8]. The scoring function is defined according to the user's stated preferences. If the score of a tuple t_1 is greater than the score of a tuple t_2 , then t_1 is the preferred tuple. If two tuples had the same score then no preference between the two would be defined.

In the *qualitative* approach, preferences between tuples are specified more indirectly, usually using *binary preference relations* which act upon attributes of tuples. The user specifies the preferred values of the attributes. Several different implementations of this basic idea have been proposed, such as *skyline* operators [5], the *winnow* operator [6], the *BMO* (Best Match Only) operator [10, 11], and the *Best* operator [21].

The qualitative approach is more flexible than the quantitative approach as some preference relations which cannot be expressed in the quantitative approach can be expressed in the qualitative approach.

All of the work done in this area assumes that the remote system that the user sends all of their preferences to is trustworthy. While this is a reasonable assumption to make in *some* cases, it cannot be made in the specific case of e-commerce. Also, the majority of the work done so far assumes that a user has expressed his or her preferences in some way, and then goes on to analyse the properties contained within the data itself. The only main body of work in how a user actually expresses his or her preferences, and in how to model these preferences, is given in [10, 11].

Agent Technologies

A lot of work has been carried out into the idea of using agents to aid e-commerce, the majority of the attention being focused on B2B agents, with B2C agents receiving a little attention. Sen and Hernandez discuss in [19] the fact that many e-business have "seller's agents" whose function it is to push merchandise or services to customers, and that the time has arrived for users to follow the trend and have "buyer's agents" whose goal is to best serve the user's interests. They propose a basic architecture of a buyer's agent. Relating this to our approach, our agent can be classed as a "buyer's agent", as its function is to best serve the user's interests.

In [13] Maes discusses how agents used as "personal assistants" that collaborate with the user can be used to reduce work carried out by the user. They can also be used to help with information overload by learning a user's preferences and filtering information presented to the user accordingly. Conceptually, this is similar to our proposed approach. However, the privacy of the user's preferences is not addressed in this work.

Recommendation agents are agents that calibrate a model of user preferences and use that model to make personalised product recommendations based upon these inferred preferences. If the agent in our proposed approach was able to learn and infer a user's preferences, it could be extended to

fulfil this function. Häubl and Murray discuss in [12] the fact that recommendation agents have the ability to influence a user's preferences according to the composition of the set of product attributes used by the agent.

Comparison agents are agents that, once given details of a specific product, can search multiple e-business' websites for that product and find the best price for the user. An example of a comparison agent is "ShopBot" [7], a comparison-agent that can visit websites of e-business', automatically learn how to extract product information from these sites, and then presents a summary of the results to the user. It uses heuristic search methods, pattern matching and inductive learning techniques. Yang *et al* present a similar agent, but is more robust when it comes to learning [24]. Although our work concentrates on using an agent to search a single site whilst attempting to maintain privacy, there is no logical reason why our agent based approach cannot be combined with the idea of comparison agents, creating a comparison agent that preserves the privacy of the user's preferences.

One final topic that has received attention is the area of how the user and their agent communicate [15]. One of the relevant conclusions drawn is that people need to be able to trust their agents to maintain as much of their privacy as possible, and that privacy and confidentiality will be among the major issues confronting the use of intelligent agents. This is where our proposed approach steps into the arena.

Privacy

Privacy is a concept that is difficult to define and has inspired much debate. In [23] Walters discusses the legal, social and philosophical conceptions of privacy. One of the points made is that "Privacy is not simply the absence of information about us by others, but our own control over the information about ourselves". This is where our technology comes in - whatever amount of privacy is actually preserved by the agent, the user (via their agent) is controlling the flow of information to the e-business. Also, in [20] Tavani discusses the concept of privacy in the specific context of the internet, and categorises privacy threats in this context as internet enhanced privacy threats - those which existed before the advent of the internet and are simply exacerbated by it, and internet specific privacy threats - threats that are specific to the internet and were created along with it.

A variety of techniques to enhance the privacy of internet users have been created in recent times, which can be broadly split into two categories - techniques designed to "anonymize" browsing of the internet, and techniques designed to manage privacy where preserving anonymity is not possible.

The first category, that of using anonymity to preserve privacy, aims to make personal information private by making it unlinkable with the identity of the user. One well known tool available to achieve this is a tool called *Anonymizer* (www.anonymizer.com). When a user is using this service to view a webpage or submit information (including personal preferences) to a remote site, it is done through Anonymizer's servers, thus the remote site has no way of detecting the IP address or identity of the user. However, this kind of technique requires a trusted third party - as the Anonymizer servers (or the user's ISP) can certainly identify the user. To achieve *complete* anonymity on the internet, tools are needed that do not rely on a trusted third-party. Two such tools are *Crowds* [17] and *Onion Routing* [9]. Crowds is a system

for protecting anonymity which operates by grouping users into large groups (crowds), where the crowd issues requests on behalf of its users. Webservers thus cannot identify who in the crowd issued the request as it is equally likely to have been any of the members of the group. Onion Routing is another solution to the problem that has been proposed. It provides anonymous connections that are highly resistant to eavesdropping and traffic analysis. The user submits an encrypted request in the form of an "onion" - a layered data structure specifying the properties of the connection at each point along the route - including cryptographic information and keys. Each point decrypts its layer to find out where the next point in the route is. Thus, a webserver only knows the identity of the person at the end of the chain. A specific example of enhancing privacy when using personal information to personalise services is the work done by Alamaki *et al* in [4], where a conceptual framework for designing privacy enabled service architectures is presented.

While these anonymising tools are useful for activities on the internet where users have no need to reveal their identity, some activities require that the identity of the user be revealed. One example of this is in e-commerce - as soon as a user decides to purchase an item they will need to provide delivery details and payment information, losing their anonymity. Previously anonymous preference information can then be linked directly with the user's identity. Other privacy preserving technologies that do not rely on remaining anonymous are therefore required. In 1997 the World Wide Web Consortium (W3C) proposed a standard known as the Platform for Privacy Preferences (P3P). Here, a user expresses his or her web privacy settings - what the user personally considers acceptable. When they visit a website, the browser only allows data to be exchanged if the site's published privacy policy matches the user's privacy settings. However, P3P only states what the website's policies are, it does not ensure these policies are actually enforced, thus the user has to trust that the e-business will keep to its promises. Our proposed technology enables users to conduct business with e-businesses that they do not even trust.

3. BASIC NOTATIONS

In describing the system we have designed that implements our proposed new approach, we need to use some terms and definitions. This section introduces the basic definitions used throughout the rest of the paper.

DEFINITION 1. A relation R consists of a set of attributes labelled $[A_1, A_2, \dots, A_k]$. Each attribute A_i has an associated Domain, D_i , which is a set of values. Any instance of R consists of a finite set of tuples, where each tuple $t \in D_1 \times D_2 \times \dots \times D_k$.

The following is an example of a relation, and we use this throughout the rest of the paper to illustrate our approach.

EXAMPLE 4. "Cameras" is a relation whose attributes are (Make, Model, MegaPixels, Memory, Zoom, Price). An instance of this relation is given in the following table:

Make	Model	Pixels	Memory	Zoom	Price
Canon	Ixus V3	3.2	64MB	2	£260
Canon	Powershot A70	3.0	64MB	3	£245
Canon	Powershot G3	4.0	256MB	4	£449
Fuji	FinePix S304	3.2	64MB	6	£275
Nikon	Coolpix 2100	2.1	64MB	3	£168
Nikon	Coolpix 3100	3.1	128MB	3	£235
Olympus	PowerCam 3	2.2	128MB	2	£260
Olympus	HandyCam D45	2.8	128MB	3	£245

In the following, we introduce the concept of preferences. Without loss of generality, we define user preferences in terms of a single relation.

DEFINITION 2. Attribute Preference. *An Attribute Preference, denoted by AP , is a preference of a user acting over a single attribute. There are two types of APs - Value-Specific APs and Numeric-Operator APs. Value-Specific APs are defined in the form of a list of specific values the user would like the attribute to take, along with their relative preferences, while Numeric-Operator APs take the form of numeric operators over the attribute. We define $|AP|$ to be the size of a particular Value-Specific AP - the amount of values specified by the user in that particular AP.*

DEFINITION 3. Table Preference. *A Table Preference, denoted by TP , is a preference expressed over a set of Attribute Preferences. We define $|TP|$ to be the size of a particular TP - the number of Value-Specific APs in the TP.*

Some examples of APs and TPs are:

- AP_1 : I'd prefer the Make to be Canon or Olympus, otherwise Fuji or Nikon;
- AP_2 : I'd prefer the lowest Price;
- TP_1 : AP_1 and AP_2 are equally as important to me;
- TP_2 : AP_1 is more important to me than AP_2 .

Note that AP_1 is a Value-Specific AP where $|AP_1| = 4$, AP_2 is a Numeric-Operator AP, and TP_1 and TP_2 are TPs where both $|TP_1|$ and $|TP_2| = 2$.

To keep our approach general, we define only two abstract operators for expressing the preference relation between values and between APs. We use the same operators as defined in [10]. These are the Pareto (\otimes) and the prioritised (&) accumulations.

DEFINITION 4. Pareto and Prioritised Accumulation. *The pareto accumulation operator (\otimes) defines a relation between two values or two APs (P_1 and P_2), where P_1 is considered equally as preferred as P_2 , whereas the prioritised accumulation operator ($\&$) defines a relation where P_1 is considered more preferred than P_2 .*

To give some examples, the user's preferences given previously are expressed using the two operators as follows:

- AP_1 : Make = (Canon \otimes Olympus) & (Fuji \otimes Nikon);
- AP_2 : Lowest(Price);
- TP_1 : $AP_1 \otimes AP_2$;
- TP_2 : $AP_1 \& AP_2$.

DEFINITION 5. Preference Relation. *A Table Preference defines a binary preference relation \succ over pairs of tuples. We denote preferences between pairs of tuples (t_1, t_2) in the following fashion:*

$$t_1 \succ t_2 \text{ reads as: } t_1 \text{ is preferred to } t_2$$

When working with preferences acting over finite domains, a common method of representing the preferred ordering of tuples is by constructing a preference graph.

DEFINITION 6. Preference Graph (PG). *In a Preference Graph, the nodes represent the tuples of a relation and directed edges between the nodes represent their relative preferences. The predecessor of a specific node is more preferred than that node. Nodes with no predecessor are the maximal elements - the most preferred.*

For example, given $(t_2 \succ t_1)$, $(t_4 \succ t_5)$, $(t_1 \succ t_4)$ and $(t_1 \succ t_3)$, the corresponding PG is:

$$\begin{array}{c} t_2 \rightarrow t_1 \rightarrow t_3 \\ \downarrow \\ t_4 \rightarrow t_5 \end{array}$$

In this paper we extend the notion of a Preference Graph to create a *User Preference Graph*, which we use to model the preference relationship between preferred values of a user.

DEFINITION 7. User Preference Graph (UPG). *In a User Preference Graph, the nodes represent the preferred values of a user and the directed edges between the nodes represent their relative preferences. When we display a UPG, we denote explicitly stated (by the user) preference edges using the symbol \Rightarrow . Each node has an associated level which corresponds to the position of the node in the prioritised preference hierarchy.*

For example, if the user preferred the Make "Nikon" to "Canon" (Make = Nikon & Canon), the UPG would look like this:

$$\begin{array}{cc} \text{Level 1:} & \text{Level 2:} \\ \text{Make: "Nikon"} & \Rightarrow \text{"Canon"} \end{array}$$

A UPG is *complete* if there is no node that is not connected to at least one other node, and we are able to evaluate the most preferred node out of any single pair of nodes - thus representing a set of user preferences with a fully defined prioritised ordering between every single item.

Nodes sharing a common level are equally preferred, as defined by the user in their preference statement. Level one holds the most preferred item(s) in each AP, level two holds the second most preferred item(s), and so on. The item(s) in Level one (the maximal elements in the UPG) correspond to what we term the *ideal* constraints - the query that would search for the user's ideal item. As you move down the levels, what we term *non-ideal* constraints ("alternates") appear. These alternates are what the user specifies as second choices, third choices, etc.

4. MODELLING USER PREFERENCES

Before we can describe the architecture of our proposed new approach and methods used to gradually release a user's preferences, we must first be able to obtain the user's preferences and model them in a way that enables our methods to work.

The user's agent needs full knowledge of the user's preferences in order to release them gradually. Thus, the obvious first step is for the user's agent to obtain the user's preferences. The agent then analyses the preferences and decomposes them into elements. It then uses these to create a model of the user's preferences, in the form of a *complete* User Preference Graph (UPG). We discuss each of these steps in turn.

4.1 Obtaining the User’s Preferences

There are two major methods available for the agent to actually obtain these preferences. The simplest is for the user to communicate their preferences to their agent directly through some simple interface, each time they wish to conduct a preference based search. Alternatively, the agent could learn the user’s preferences over time, remembering and inferring them. For simplicity in explaining our new approach, in this paper we assume that the user states their preferences directly to the agent. However, the method of obtaining the user’s preferences does not impact on our approach, as long as they are obtained by some means.

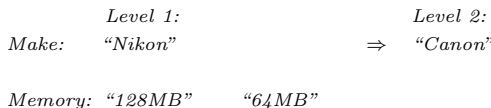
The form that the user’s preferences take when stated are as discussed in Definitions 2 and 3. The user simply states their Attribute Preferences and their Table Preference. APs come in two flavours - Value-Specific APs and Numeric-Operator APs. Value-Specific APs are expressed as a list of specific preferred values and their relative preferences, and are used to select relevant items from a database of items, while Numeric-Operator constraints are expressed in the form as operators such as “Lowest(Price)”, and are used to order the search results, and possibly pick a certain amount of the “best” of the relevant items returned. TPs are expressed as a list of APs and their relative preferences. The user states the preference relation between values in an AP and between APs in a TP using one of two defined accumulation operators - \otimes and $\&$.

Once the agent has possession of the user’s preferences, the next step is to create a complete UPG to model them.

4.2 Creating a Complete UPG

In order that the agent can gradually release the user’s preferences, a model representing these preferences is needed. The model we have created to achieve this is a *User Preference Graph* (see Definition 7). The agent takes the user’s TP and constructs a UPG to represent it.

When we consider a TP (containing combined preferences acting over multiple attributes), we see that while a *complete* ordering can be initially present (if the user used the prioritised accumulation operator ($\&$) between *every* pair of values specified in each AP and between *every* AP pair) it does not *necessarily* contain such an ordering. To explain this, simply imagine a case where a user expresses a pareto accumulation between a pair of APs ($AP_1 \otimes AP_2$), where AP_1 states that the user would prefer the make of a digital camera to be “Nikon” followed by “Canon” (Make = Nikon $\&$ Canon), and AP_2 states that the user would prefer a memory size of “128MB” or “64MB” (equally preferred) (Memory = 128MB \otimes 64MB). The second AP defines two nodes in a UPG with no priority between them, hence no edge between them. The two APs define two unrelated partial orders with no particular priority specified between them, hence no edge between them. When a UPG was built, we would end up with three disjoint sets of nodes (over two disjoint APs) in the graph:



In previous work in the area of preferences embedded directly into a DBMS, this occurrence would not pose a prob-

lem as all items are sent to the remote system together. In our proposed method however, this occurrence does pose a problem. If we are going to release preferences gradually, having two nodes with no preference relation between them creates an ambiguity if we have to choose one to release next. To avoid encountering these ambiguities, we have put a constraint on the model used such that it must be *complete* - there should be a defined ordering present between any and all values in each AP, and a defined ordering present between each AP. We should be able to evaluate the most preferred item out of any single pair of items in the entire UPG. Expressed in terms of the UPG of the Table Preference, there must be no disjoint nodes in the UPG of the TP.

To achieve a complete UPG, we need to decide where to place an edge between the disjoint values in each AP, and where to place an edge between the disjoint APs, and in which direction these edges should be. This can be done in two steps:

- Firstly, we individually create a UPG for each of the APs, and make it complete by introducing edges between disjoint nodes.
- Secondly, we join these separate UPGs to form a UPG for the TP, and make it complete by introducing edges between the maximal nodes of disjoint APs.

Note that when creating the UPG for the TP, we only include Value-Specific APs, ignoring any Numeric-Operator APs (such as “Highest(Price)”), as these are not used to select items from the database, but are used to order the search results or pick the “most” relevant item(s) from the search results. Although ignored now, they become pertinent later as they are used by the agent when presenting the final search results to the user.

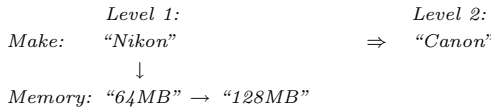
A naive method for making a UPG complete could be to just choose each disjoint item’s (or AP’s) relative importance randomly. We would take all of the sets of disjoint, equally preferred values (or APs), and for each set randomly choose an ordering between pairs of values (or maximal values), until there was a defined ordering between every value (or AP). We would do this repeatedly until there are no disjoint nodes or orders, thus we now a complete UPG.

Another possible method that could be used to make a UPG complete could be for the agent to request for the user to decide on the relative importance of disjoint items, as they might have a preference between items which they previously neglected to state.

A more interesting method however could be to use statistics from the relation we are to query to decide on the relative importance of the values and APs. For example, using the TP above, if the domain of the attribute “Make” only contained five distinct values while the domain of the attribute “Memory” contained 25 distinct values, the optimal attribute to select as “most important” would be “Make”, as releasing the next value of this AP first is more likely to return satisfactory tuples. The agent could query the remote system for the cardinality and distribution of a particular set of attributes in order to decide where to place the edges between the disjoint orders, providing of course that the remote system provided this capability.

When we display a UPG, we denote explicitly stated (by the user) preference edges using the symbol \Rightarrow , and we denote calculated preference edges using the symbol \rightarrow . To

illustrate this, following on from the previous example, imagine we have now randomly chosen the ordering between the disjoint values in the “Memory” AP and between the two disjoint APs, such that “64MB” \succ “128MB”, and $AP_1 \succ AP_2$. The complete UPG would look like this:



As you can see, there is now a definite ordering in the UPG. The two items in the “Make” AP remain unchanged, as there was already a defined ordering present in that AP. There is now a preferred ordering between the two APs, and between the two “Memory” items. These items share the same level (level one) since they were described as equally preferred by the user, however, there is a calculated preference edge between them to indicate which should be released first.

Using a method such as described above, we can now join together all of the disjoint values in each AP’s UPG, and all of the disjoint APs in the TP by creating directed edges between nodes, according to the calculated order. This gives us a complete UPG and therefore a definite order between *all* values. Also, we see that the user’s “ideal query” corresponds to all of the maximal nodes, while all of the nodes below correspond to the user’s non-ideal constraints. Figure 3 shows the basic algorithm for constructing the TP’s complete UPG.

1. Construct a complete UPG for each AP:
 - If the AP is a Numeric-Operator AP (such as Highest(x)) ignore it and skip to the next AP;
 - Otherwise, insert each value specified by the user onto the graph, creating an initial UPG for that AP (putting each equally preferred item on the same level);
 - Insert directed edges between nodes where there is a user specified priority;
 - For every set of disjoint nodes (equally preferred items) in the UPG, introduce directed edges between pairs of nodes, according to some defined method.
2. Construct a UPG for the TP:
 - Insert each AP’s UPG onto the graph, creating an initial UPG for the TP;
 - Insert directed edges between the maximal nodes of APs where there is a user specified priority;
 - For every set of disjoint APs (equally preferred APs), introduce directed edges between pairs of AP’s maximal nodes, according to some defined method.

Figure 3: Constructing the TP’s complete UPG

5. GRADUAL PREFERENCE RELEASE

Assuming that we now have captured the user’s preferences and modelled them, we are now able to gradually release the user’s preferences in a way that guarantees the trustworthiness of the results and preserves as much privacy as possible.

The basic method of gradual release is fairly simple, however, certain elements of the method prove rather intricate, and involve trade-offs between the amount of privacy preserved, the quality of results returned, and the performance.

The gradual release algorithm consists of only a few main steps (see Figure 4 for the full algorithm). Firstly, we take the complete UPG representing the user’s preferences. Next, we take all of the maximal nodes in the UPG and use them to construct the initial interim query. This query effectively searches for the user’s *ideal* item. The results of this query are returned to the agent. If enough results were returned, then the agent takes the previously discarded Numeric-Operator APs and applies them to the result set to give them another level of preferred ordering. The agent would also discard some of the results if the user specified a limit to the amount of results they wanted - it would remove the least preferred of the results. However, if no (or not enough) results were returned, then the agent would relax the query a step (see Section 5.1) and resend this to the e-business’ system. It would do this repeatedly until enough results were returned, or the agent ran out of preferences.

1. Get the complete UPG created in Figure 3;
2. Take all of the maximal values and use them to construct an *interim query*;
3. Release the interim query to the remote system, get the results;
4. If enough results were returned, or if we’ve run out of non-ideal constraints to add to the interim query:
 - Order the results by any Numerical Operator APs specified by the user;
 - If a limit to the amount of results was specified, remove the least preferred results until the limit was met;
 - Return the results to the user and quit.
5. Else, relax the interim query (see Figure 5) and go back to step 4.

Figure 4: A Gradual Release Algorithm

5.1 Relaxing the query

The method by which we relax the query is the point at which trade-offs between amount of privacy preserved, quality of results and performance come to the fore.

There are two extremes by which we can relax the query. At one end of the scale, we can add all of the preferences at once. This results in only one query being sent (thus high performance from the point of view of information transmitted), resulting in results whose trustworthiness cannot be guaranteed and no privacy being preserved (as explained in the introduction). Thus, we can rule out this approach.

At the other end of the scale, we can release the user’s preference one item at a time. This could result in a large amount of queries being sent (see Section 5.2 for details), thus possibly very low performance. However, this does guarantee trustworthy results, and guarantees that we retain as much privacy as possible. Experimentation needs to be carried out in order to verify the performance of this approach, but it seems likely that it will not be acceptable.

Thus, it seems that the best approach to use will be somewhere in between the two ends of the scale, effectively releasing the user’s preferences a few at a time. This would result in an acceptable amount of queries being sent (thus acceptable performance), whilst achieving results that are as trustworthy as possible and the amount of privacy preserved is as high as possible.

In an attempt to keep the demonstration and discussion of our algorithm as simple as possible however, in the remainder of the paper we will assume that the agent has been instructed to preserve as much privacy as possible by releasing only one item at a time when relaxing the query, ignoring performance issues. The algorithm for doing this is shown below.

1. Get the complete UPG created in Figure 3 and the previous query;
2. Take the least preferred AP with unused values specified at the current level, backtrack the query a step, and replace the current value of that AP with the new value
3. If there are no APs with unused values left at that level, go to the next level and go to step 2.
4. If there are no more levels, there are no more non-ideal constraints to add, so return nothing.

Figure 5: Relaxing the query

5.2 Performance Discussion

When attempting to maintain as much privacy as possibly (by changing only one item at a time when relaxing the query), performance is a big issue. The initial interim query should consist of $|TP|$ items (the amount of Value-Specific APs). As we introduce non ideal constraints we replace existing ones, so each query sent would remain $|TP|$ items long.

In the best case, the user’s ideal item would be present initially, and so there would only be one query sent by the agent, $|TP|$ items long. In the worst case, if no items matching the user’s preferences were present, the amount of queries sent would be the product of the size of all Value-Specific APs. Thus, the lower and upper bounds on the amount of queries sent using this method of relaxing the query is:

$$\text{Min}(queries) = 1$$

$$\text{Max}(queries) = \prod_{i=1}^{|TP|} AP_i$$

5.3 Privacy Discussion

To preserve the highest amount of privacy possible, the initial interim query would only have a single item in it. However, if we do this, we may get huge amounts of results back (if searching a large database), which is undesirable for obvious performance reasons. This is in fact only one step away from the agent requesting the entire database be transmitted to it in order that it can search the data itself. Thus, we have to make a trade-off at this point of amount of privacy preserved versus performance. We have decided in our system that the initial interim query should contain the maximal node of each AP. Although this somewhat contradicts our goal of releasing as little preference information as possible, this is intuitively the best method. as this should result in a small amount of results returned, whilst only revealing a little of the user’s preferences - only one value from each AP.

5.4 An Example

We now demonstrate our approach with an example.

EXAMPLE 5. *Bob wishes to buy a digital camera. He expresses his preferences over a series of attributes available to search over to his agent, along with his relative preferences between the attributes, as below. He also tells the agent that he would like to view only the two most preferred items:*

AP_1 : Make = (Nikon \otimes Canon) & (Olympus \otimes Fuji), AP_2 : MegaPixels = 2-2.9 & 3-3.9 & 1-1.9, AP_3 : Memory = 128MB & 64MB, AP_4 : Highest(Zoom), AP_5 : Lowest(Price), TP : AP_2 & ($AP_1 \otimes AP_3$) & AP_5 & AP_4

The user’s agent creates a UPG for each of the APs (apart from the Numeric-Operator APs (AP_4 and AP_5)). The UPGs for these are given below (note the disconnected nodes in AP_1):

AP_1 :	Level 1:	Level 2:		
Make:	“Nikon”	“Canon”	\Rightarrow	“Olympus” “Fuji”
AP_2 :	Level 1:	Level 2:	Level 3:	
Pixels:	“2-2.9”	\Rightarrow “3-3.9”	\Rightarrow “1-1.9”	
AP_3 :	Level 1:	Level 2:		
Memory:	“128MB”	\Rightarrow “64MB”		

Next, the agent attempts to complete the UPG. In this example, it simply chooses where to place the edges between disjoint nodes randomly. As such, it introduces directed edges in AP_1 between “Nikon” and “Canon”, and “Olympus” and “Fuji”. No action is needed in AP_2 and AP_3 as there are no disjoint nodes in these APs. Next, the agent introduces edges between APs that were defined by the user (using the & operator). In this case, it introduces an edge between AP_2 and AP_1 . To make it a complete UPG, the agent would have to introduce edges between any disjoint APs, using some method. As before, the method used in this example is simply to introduce edges randomly, for simplicity. As such, it introduces a directed edge between AP_1 and AP_3 .

The complete UPG generated for the TP is shown below:

<p><i>Level 1:</i></p> <p><i>Pixels:</i> "2-2.9"</p> <p style="text-align: center;">↓</p> <p><i>Make:</i> "Nikon" → "Canon"</p> <p style="text-align: center;">↓</p> <p><i>Memory:</i> "128MB"</p>	<p><i>Level 2:</i></p> <p>⇒ "3-3.9"</p> <p>⇒ "Olympus" → "Fuji"</p> <p>⇒ "64MB"</p>	<p><i>Level 3:</i></p> <p>⇒ "1-1.9"</p>
--	---	---

Now that the agent has a complete UPG of the user's preferences, it can now gradually release them, according to the method defined in Figure 4. To achieve this, it first creates the "ideal query" for the user, and gets the results back:

```
Query 1: Find Cameras WHERE (Pixels=2-2.9)
                AND (Make=Nikon)
                AND (Memory=128MB)
Results: None
```

As the first query generated no results, the agent now relaxes the query a step, according to the method defined in Figure 5. As such, it rewrites the query and resends it:

```
Query 2: Find Cameras WHERE (Pixels=2-2.9)
                AND (Make=Canon)
                AND (Memory=128MB)
Results: None
```

Again, the query generates no results, so the agent relaxes the query another step:

```
Query 3: Find Cameras WHERE (Pixels=2-2.9)
                AND (Make=Nikon)
                AND (Memory=64MB)
Results: * Nikon - Coolpix 2100 - 2.1 - 64MB - 3 - 168
```

This query returns the first result to the agent. However, as the user requested two results and the agent currently only has one, the agent needs to relax the query again to get more.

The agent carries on doing this - relaxing the query and gathering the results, until 3 or more results in total have been returned to the agent. The following series of queries will therefore be generated and sent to the server:

```
Query 4: Find Cameras WHERE (Pixels=2-2.9)
                AND (Make=Canon)
                AND (Memory=64MB)
Results: None

Query 5: Find Cameras WHERE (Pixels=2-2.9)
                AND (Make=Olympus)
                AND (Memory=128MB)
Results: * Olympus - PowerCam 3 - 2.2 - 128MB - 2 - 260
        * Olympus - HandyCam D45 - 2.8 - 128MB - 3 - 245
```

After five queries, we now have three results. The agent now orders them according to the order in which they were received so that the items more closely matching the user's preferences are towards the top. If two or more results were received together (therefore matching the user's preferences equally), they are ordered by the numerical constraint APs (Lowest(Price) & Highest(Zoom)). The agent then discards the least preferred item of the three items (as the user requested only two), returning the following to the user (ordered by how preferred the items are):

Make	Model	Pixels	Memory	Zoom	Price
Nikon	Coolpix 2100	2.1	64MB	3	£168
Olympus	PowerCam 3	2.2	128MB	2	£260

Thus, the agent has obtained results that are the best results for the user, while minimising the preference information sent to the e-business, thus preserving some of the user's privacy. In this example, of the nine items specified by the user in their full preference statement (over five attributes), the three most "important" items to the user were never sent to the e-business. The two Numeric-Operator APs were also not released.

6. CONCLUSION

In this paper we have identified a specific area of application (namely e-commerce) where the current preference based search approach is not adequate (due to issues with the trustworthiness of results and the privacy of users' preferences) and have proposed the use of a gradual release mechanism to counter the problems. The main characteristic of our approach is that an agent acts on behalf of the user, gradually releasing the user's preferences to the DBMS, thus ensuring the results received are the "best" results for the user whilst retaining as much privacy as possible. We outlined the main points of this new approach and its basic functionality.

However, future work is needed in several areas in order to discover the most efficient way for the new approach to operate. One area is in discovering the location of the optimal point on the performance vs. amount of privacy preserved scale (if indeed there is a single point). Once this point (or points) are found, work needs to be carried out on creating the most efficient method of relaxing the query according to the location of the point(s) - how many items do we release simultaneously when relaxing the query, and what is the most efficient way of doing this? Intuitively, if the answer is more than one item at a time, this will have to involve the use of heuristics. Another area of future work is in investigating different methods to choose the relative importance of disjoint nodes and APs when completing a UPG. Finally, work needs doing on assessing how easy or difficult it would be for an e-business *determined* to gather user preferences to simply return no items in response to each query, until all preferences were recovered. Of course, doing this would mean that the e-business would make no sale, but is there any way for our agent to automatically detect this and stop it from happening?

7. REFERENCES

- [1] Special issue of the Communications of the ACM on Personalization, vol. 43, Aug. 2000
- [2] M.S. Ackerman, L.F. Cranor and J. Reagle, "Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences", In Proc' EC99 Conference on Electronic Commerce, 1999
- [3] R. Agrawal and E.L. Wimmers, "A framework for Expressing and Combining Preferences", In Proc' ACM SIGMOD, May 2000, pp.297-306
- [4] T. Alamaki *et al.*, "Privacy Enhancing Service Architectures", In Proc' PET 2002, LNCS 2482, pp 99-109
- [5] S. Borzsonyi, D. Kossmann, and K. Stocker, "The skyline operator". In Proc' IEEE Conf' on Data Engineering, 2001
- [6] J. Chomicki, "Querying with Intrinsic Preferences", In Proc' 8th EBDT, Mar. 2002, pp.34-51

- [7] R.B. Doorenbos, O. Etzioni and D.S. Weld, "A Scalable Comparison-Shopping Agent for the World-Wide Web", In Proc' First International Conference on Autonomous Agents, 1997
- [8] V. Hristidis, N. Koudas, and Y. Papakonstantinou, "PREFER: A System for the Efficient Execution of Multiparametric Ranked Queries", In Proc' ACM SIGMOD, 2001, pp. 259-270
- [9] D. Goldschlag, M. Reed, P. Syverson, "Onion Routing", Communications of the ACM, Volume 42, Number 2 (February 1999)
- [10] W. Kießling, "Foundations of Preferences in Database Systems", In Proc' 28th VLDB, Aug. 2002
- [11] W. Kießling, G. Köstler, "Preference SQL - Design, Implementation, Experiences", In Proc' 28th VLDB, Aug. 2002
- [12] G. Häubl and K.B. Murray, "Recommending or persuading?: the impact of a shopping agent's algorithm on user behavior", In Proc' EC01 Conference on Electronic Commerce, 2001
- [13] P. Maes, "Agents that reduce work and information overload", Communications of the ACM, Volume 37, Number 7 (July 1994)
- [14] H. Nwana *et al.*, "Agent-Mediated Electronic Commerce: Issues, Challenges and some Viewpoints", In Proc' Second International Conference on Autonomous Agents, 1998
- [15] D.A. Norman, "How might people interact with agents", Communications of the ACM, Volume 37, Number 7 (July 1994)
- [16] D. Riecken, "Intelligent Agents", Communications of the ACM, Volume 37, Number 7 (July 1994)
- [17] M. K. Reiter, A. D. Rubin, "Crowds: anonymity for Web transactions", ACM Transactions on Information and Systems Security (TISSEC), Volume 1, Issue 1 (November 1998).
- [18] S. Spiekermann, J. Grossklags and B. Berendt, "E-privacy in 2nd Generation E-Commerce: Privacy Preferences versus actual Behaviour", In Proc' EC01 Conference on Electronic Commerce, 2001
- [19] S. Sen and K.Hernandez, "A buyer's agent", In Proc' Fourth International Conference on Autonomous Agents, 2000.
- [20] H. Tavani, "Privacy Online", ACM SIGCAS Computers and Society, Volume 29, Issue 4 (December 1999)
- [21] R. Torlone and P. Ciaccia, "Finding the Best when it's a Matter of Preference", In Proc' 10th Italian National Conference on Advanced Database Systems (SEBD 2002), Portoferraio, Italy, 2002.
- [22] M. Teltzrow and A. Kobsa, "Impacts of User Privacy Preferences on Personalized Systems - a Comparative Study", In Proc' CHI2003 Conference on Human Factors in Computing Systems, 2003
- [23] G. Walters, "Privacy and Security: An Ethical Analysis", ACM SIGCAS Computers and Society, Volume 31, Issue 2 (June 2001)
- [24] J. Yang, J. Choi, J. Kim, H. Ham and K. Lee, "A More Scalable Comparison Shopping Agent", In Proc' EIS2000 Engineering of Intelligent Systems, 2000, pp. 766-772.