Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications

Stefan Holland[1], Martin Ester[2], and Werner Kießling[1]

[1] Institute of Computer Science, University of Augsburg, D-86159 Augsburg, Germany
{holland,kiessling}@informatik.uni-augsburg.de
[2] School of Computer Science, Simon Fraser University, Burnaby BC, Canada V5A 1S6
ester@cs.sfu.ca

Abstract. Advanced personalized e-applications require comprehensive knowledge about their user's likes and dislikes in order to provide individual product recommendations, personal customer advice and custom-tailored product offers. In our approach we model such preferences as strict partial orders with "A is better than B" semantics, which has been proven to be very suitable in various e-applications. In this paper we present novel Preference Mining techniques for detecting strict partial order preferences in user log data. The main advantage of our approach is the semantic expressiveness of the Preference Mining results. Experimental evaluations prove the effectiveness and efficiency of our algorithms. Since the Preference Mining implementation uses sophisticated SQL statements to execute all data-intensive operations on database layer, our algorithms scale well even for large log data sets. With our approach personalized e-applications can gain valuable knowledge about their customers' preferences, which is essential for a qualified customer service.

1 Introduction

The enormous growth of web content and web-based applications leads to an unsatisfactory behavior for users: search engines retrieve a huge number of results and they are left on their own to find interesting web sites or preferred products. Such a behavior leads not only to frustrated users but also to a reduction of turnover in commercial businesses because customers who are willing to buy cannot do it since they do not find the right product even if it is available. In recent years, several techniques have been developed to build user adaptive web sites and personalized web applications. For instance, e-commerce applications use link personalization to recommend items based on the customer's buying history or some categorization of customers based on ratings and opinions [11]. Another technique is content personalization: web pages present different information to different users based on their individual needs. Thereby, the user can indicate his preferences explicitly using the predefined tools of the underlying portal or the preferences may be inferred automatically from his profile.

State-of-the-art personalization techniques suffer from some drawbacks. Manually customizing web sites is not very feasible to the customer since it is a very time-

consuming task to select relevant content from the huge repertoire provided by the web portal. Personalizing products or web content automatically is a more promising approach. However, the current approaches of automatic personalization lack of preference models with limited expressiveness. State-of-the-art techniques either use scores to describe preferences [7] or just distinguish between liked and disliked values [9]. Thus, complex *"I like A more than B"*-relationships as well as *preferences for numeric attributes* cannot be expressed in a natural way. Furthermore, these approaches are not able to handle *dependencies among preferences*. For example, two preferences can be of equally importance to a customer or one preference can be preferred to another one.

A very expressive and mathematically well-founded framework for preferences has recently been introduced [5]. Customer preferences are modeled as strict partial orders with "A is better than B"-semantics, where negative, numeric and complex preferences form special cases. This approach has been proven to be very suitable for modeling user preferences of almost any complexity. Standard query languages like SQL and XPATH were extended by such preferences [6] in order to deal carefully with user wishes. In this paper, we present algorithms for automatically mining such strict partial order preferences from user log data. Basic categorical and numerical preferences are discovered based on the frequencies of the different attribute values in the user log. These base preferences are then combined to detect complex preferences.

The rest of the paper is organized as follows: After a survey of related work in section 2 we describe the underlying preference model and Preference Mining requirements in section 3. In section 4, we present algorithms for mining categorical, numerical and complex preferences. Section 5 summarizes the results of an extensive experimental evaluation of the accuracy and efficiency of the proposed algorithms. We conclude our paper with a summary and outlook in section 6. All proves are omitted her, but can be found in the extended version of this paper [3].

2   Related Work

Several research groups have studied the usage of log data analysis for personalized applications. In particular, web log mining is a commonly used approach of analyzing web log data with data mining techniques for the discovery of sequential patterns, association rules or user clusters. Such mining techniques have been applied to provide personalized link recommendations to web users [10]. Thereby the user profile of the current user is matched against one or more previously discovered usage profiles.

Lin et al. applied association rule mining algorithms for collaborative recommendations [9]. They mapped user ratings (liked/disliked/not rated) of articles into transactions and used their algorithms in order to detect association rules within these transactions. Such gained rules can be used for the recommendation of articles to the users.

Beeferman and Berger analyzed query log data of search engines [1]. They developed clustering algorithms in order to find groups of URLs that match various keywords given by the user. This approach is not only helpful for delivering better search results but also for the construction of web categories and the generation of ontologies. In [4], Joachims analyzed clickthrough data to improve the results of search engines. He uses the search results that are chosen by the user as additional information. He argues that selected items are better in the opinion of the user and applies this knowledge to find better rankings for future search results.

Our Preference Mining techniques can work either on web logs or query logs. The main advantage of our approach is the semantic expressiveness of the Preference Mining results. Our algorithms compute no scores to distinguish between liked and disliked values but detect intuitive preferences like positive or negative preferences, numerical preferences or complex preferences. Personalized web applications [11] can gain significant improvements by using detailed knowledge about user preferences.

3   User Preferences in Log Data

In this section we revisit those aspects of the preference model of [5] that are relevant for the scope of this paper. We also define requirements on the user log data for mining such preferences.

3.1   Preferences as Strict Partial Orders

A preference P is defined as a strict partial order $P = (A, <_p)$, where $A = \{A_1, …, A_k\}$ denotes a set of attributes with corresponding domains $dom(A_i)$. The domain of A is defined as Cartesian product of the $dom(A_i)$, $<_p \subseteq dom(A) \times dom(A)$ and $x <_p y$ is interpreted as "y is better than x". A set of intuitive preference constructors for base and complex preferences is defined.

The constructors for base preferences on categorical domains are POS(A, POS-set), NEG(A, NEG-set), POS/NEG(A, POS-set; NEG-set), POS/POS(A, POS1-set; POS2-set) and EXP(A, E-graph). The POS-set $\subseteq dom(A)$ of a POS preference defines a set of values that are better than all other values of dom(A). Analogously, the NEG-set of a NEG preference describes disliked values. The POS/NEG preference is a combination of the previous preferences and in a POS/POS preference optimal values (POS1-set) and alternative values (POS2-set) can be specified. In E-graph of an EXPLICIT preference a user can specify any better-than relationships.

The preference constructors for numerical domains include AROUND(A, z), BETWEEN(A, [low, up]), LOWEST(A) and HIGHEST(A). In an AROUND preference the desired value is z, but if this it not available values with nearest distance apart from z are best alternatives. For a BETWEEN preference the values within [low, up] are optimal. For LOWEST (HIGHEST) preferences lower (higher) values are better.

Preferences can inductively be combined with complex preference constructors. A Pareto preference $P = P_1 \otimes P_2$ treats the underlying preferences as equally important and a Prioritized preference $P = P_1 \& P_2$ treats P_1 as more important than P_2. For instance, P = POS(author, {Douglas Adams, Edgar Wallace}) & NEG(binder, {paperback}) denotes a POS preference for the authors Douglas Adams and Edgar Wallace, and a NEG preference for paperbacks, whereby the latter preference is less important.

This definition of preference constructors has been proven to be appropriate to describe complex user wishes. Preference engineering examples are shown in [5]. Our Preference Mining developments should be consistent to this preference model. Therefore, not only all base and complex preferences should be detectable by the Preference Miner but also preference properties like preference hierarchies or preference algebra laws (see [5] for details) should be valid for the detected preferences.

3.2   Requirements on User Log Data in Web Applications

Data mining benefits from the availability of a huge amount of data since having many records ensures the statistical significance of patterns [8]. Log data of user transactions can have several sources like web server log-files or transaction logging on an application server.

Web server logs are generated by the web server when a user is visiting a web site. Such files can comply with standardized formats like the Common Logfile Format[1]. The log data includes the IP of the client host, the current timestamp and the URL (uniform resource locator) he is visiting. Valuable information about a user's wishes is stored in the URL, since it contains not only the address but also requested keywords or preferred product properties the user inserted into a web form. For example, if the user requests the book "The Raven" in the e-shop Barnes & Noble[2] the logged URL is *http://search.barnesandnoble.com/booksearch/results.asp?WRD=The+Raven*. But web server logs also have some disadvantages, especially for e-commerce applications [8]. Events like "add to cart" or "change item" are not available in web logs. Furthermore, a user can deactivate cookies in his browser, so no session information or user identification is available. Preference Mining on web server logs requires some data preprocessing. User input like "The Raven" in the above example has to be extracted from the logged URL and has to be stored in a relational database since our Preference Mining algorithms work on database relations. Furthermore, user identification is required to detect preferences for each customer separately.

Application server logs can handle user transactions much better [8]. User and session identification can be accomplished with a login and logout mechanism. Another advantage is the capability to detect business events like "add to cart" or "buy items". For example, an e-commerce application server can record queries, search results, selected items and bought products for each customer separately. Furthermore, application server log data can be stored in databases and therefore huge amount of log

---

[1] http://www.w3.org/Daemon/User/Config/Logging.html

[2] http://www.barnesandnoble.com

data can be managed by using database technology. The Preference Mining algorithms can work directly on these log relations without any data preprocessing. For instance, analyzing the properties of bought products can lead to preferences about liked and disliked features, price preferences and dependencies between such preferences.

While browsing or shopping in an online environment, a customer has typically several different types of input fields for interacting with the underlying system. Text fields allow the input of keywords and choices allow the selection of static or dynamic predefined values of an attribute. To describe these different situations we define the closed world assumption and the difference between static and dynamic domains.

Definition 1 (Closed world assumption (CWA))
The assumption that a customer knows all possible values of an attribute is called Closed World Assumption or CWA. If this assumption doesn't hold we abbreviate it with ¬CWA.

Definition 2 (Static and dynamic domains)
If a domain of attribute values is constant over time, we call it a static domain otherwise we call it a dynamic domain.

The CWA is required for the detection of negative preferences since only if the user knows all possible values we can assume dislike for values he never selected. Otherwise (¬CWA), we can't decide whether he doesn't know or doesn't like such values. For instance, in a book shop the customer knows all possible values for binder (paperback or hardcover) but doesn't know all available authors. After submitting a search query, a customer gets a set of results and chooses one or more of them as his preferred products. Such search results define dynamic domains and can lead to valuable clickthrough data, which can be used to get information about explicit user preferences since the clicked items of the query result are preferred by the user [4].

4   Preference Mining Algorithms

In this section we present algorithms for mining the strict partial order preferences introduced in section 3.1. Our methods work on log relations as described in section 3.2 and use appropriate data mining and statistical methodologies in order to detect the right preference and correct additional information like POS-sets. To detect base preferences we use the frequencies of the different values in the log relation.

Definition 3 (Frequency of a value)
Let A be an attribute of a log relation R and $x \in$ dom(A). The number of entries of x in R(A) is called frequency of x or $freq_A(x)$. If dom(A) is numerical, $freq_A([x_1, x_2])$ denotes the number of entries of all values between x_1 and x_2 ($x_1 \leq x_2$).

We have introduced the concept of user-defined preferences $P = (A, <_P)$. The actual user preferences shall be predicted from the implicit preferences hidden in the user log data. To that purpose, we define data-driven preferences denoted by $P_D = (A, <_{PD})$.

Definition 4 (Data-driven preference)
- For categorical domains dom(A) a data-driven preference $P_D = (A, <_{PD})$ is defined as: $x <_{PD} y$ iff $freq_A(x) < freq_A(y)$.
- For numerical domains dom(A) a data-driven preference $P_D = (A, <_{PD})$ is defined as: $x <_{PD} y$ iff $\exists \varepsilon > 0$: $freq_A([x-\varepsilon, x+\varepsilon]) < freq_A([y-\varepsilon, y+\varepsilon])$.

As it can easily be shown data-driven preferences define strict partial orders. Depending on the design of the log data, values can be products (e.g. search results) or just product properties like color or price. If the frequency of a value x is zero, a customer has never selected the according value. If CWA holds, $freq_A(x) = 0$ means that a customer doesn't like the property x because he never selected it although he knows it. Otherwise, if CWA doesn't hold, the customer may either not like the property x or may never have heard of it. The relation $freq_A(x) < freq_A(y)$ shows that the corresponding customer has selected y more often than x. In this sense the relation $x <_{PD} y$ denotes a preference.

Numeric domains need a slightly different approach to data-driven preferences. For instance, an attribute A may have the real numbers as domain $(dom(A) = \mathbb{R})$ and we want to test, if a user has a data-driven LOWEST(A) preference, i.e. lower values are better and should occur with higher frequencies. Since $\mathbb{R}$ consists of an infinity number of different values, the log relation only contains some of them and typically each value occurs only a few times in the log relation. Therefore, we use frequencies of intervals. E.g. for a data-driven LOWEST preference the relation $freq_A([x-\varepsilon, x+\varepsilon])$ $< freq_A([y-\varepsilon, y+\varepsilon])$ for $y < x$ must hold for some ε.

4.1   Mining Categorical Preferences

Based on $P_D = (A, <_{PD})$ we can define data-driven preferences for categorical domains.

Definition 5 (Data-driven preferences for categorical data)
Let A be a categorical attribute of a log relation R and POS-set, NEG-set, POS1-set, POS2-set, $E \subseteq dom(A)$.
- There is a data-driven POS preference, iff $\forall x \in$ POS-set, $\forall y \notin$ POS-set: $y <_{PD} x$.
- There is a data-driven NEG preference, iff $\forall x \in$ NEG-set, $\forall y \notin$ NEG-set: $x <_{PD} y$.
- There is a data-driven POS/POS preference, iff $\forall x \in$ POS1-set, $\forall y \in$ POS2-set, $\forall z \notin$ (POS1-set $\cup$ POS2-set): $y <_{PD} x$ and $z <_{PD} y$.
- There is a data-driven POS/NEG preference, iff $\forall x \in$ POS-set, $\forall y \in$ NEG-set, $\forall z \notin$ (POS-set $\cup$ NEG-set): $z <_{PD} x$ and $y <_{PD} z$.
- Let $<_E$ be a strict partial order on E. A data-driven EXPLICIT preference holds, iff (1) $\forall x, y \in$ E with $x <_E y$: $x <_{PD} y$, and (2) $\forall u \in$ E, $\forall v \notin$ E: $v <_{PD} u$.

For a data-driven POS preference the values in the POS-set must occur more often than the other values and in a data-driven NEG preference the other values must occur more often than the values in the NEG-set. POS/POS and POS/NEG run analogously. A data-driven EXPLICIT preference with underlying E-graph exists, if a

value y occurs more often than any successor x in E-graph. Values outside the E-graph occur with lowest frequencies.

The main task for an algorithm for mining categorical preferences is the detection of proper POS-sets, NEG-sets, etc. Consider the following example of frequencies for an attribute author (CWA doesn't hold, the domain is static):

Table 1. Example of frequencies for an attribute "author"

Douglas Adams	Edgar Wallace	Natalie Angier	Agatha Christie	John Grisham
50	49	2	3	2

The set {Douglas Adams} is a correct POS-set for a data-driven POS preference. But intuitively, the set {Douglas Adams, Edgar Wallace} denotes are more reasonable POS-set since these two values occurred much more frequently than Natalie Angier, Agatha Christie and John Grisham. The following algorithm for mining categorical preferences uses cluster techniques in order to detect such proper sets.

Algorithm 1: Miner for categorical preferences in static domains
INPUT: log relation R, attribute A, dom(A)
(1) Compute for each value x_i the frequency in the log relation $\text{freq}_A(x_i)$.
(2) Compute a clustering of the x_i with $\text{freq}_A(x_i) \geq 1$ by using a clustering technique.
(3) Depending on the clustering results we have the following possibilities:
   (a) There is only one cluster C_1 and CWA holds. Here we have a NEG(A, $\{x \in \text{dom}(A) \mid \text{freq}_A(x) = 0\}$) preference.
   (b) There are two clusters C_1 and C_2, where $\forall c_1 \in C_1, \forall c_2 \in C_2$: $\text{freq}_A(c_2) < \text{freq}_A(c_1)$.
   (b1) If ¬CWA, we have a POS(A, C_1) preference.
   (b2) If CWA, there is a POS/NEG(A, C_1; $\{x \in \text{dom}(A) \mid \text{freq}_A(x) = 0\}$) preference.
   (c) There are three clusters C_1, C_2 and C_3, where $\forall c_1 \in C_1, \forall c_2 \in C_2, \forall c_3 \in C_3$: $\text{freq}_A(c_3) < \text{freq}_A(c_2) < \text{freq}_A(c_1)$. Here we have a POS/POS(A, C_1; C_2) preference.
   (d) There are more than three clusters C_1, …, C_n, where $\forall c_1 \in C_1, \forall c_2 \in C_2, …, \forall c_n \in C_n$: $\text{freq}_A(c_n) < … < \text{freq}_A(c_2) < \text{freq}_A(c_1)$. Here we have an EXPLICIT preference EXP(A, $<_E$) with $c_n <_E ... <_E c_2 <_E c_1, \forall c_1 \in C_1, \forall c_2 \in C_2, …, \forall c_n \in C_n$.
   (e) In all other situations there is no data-driven preference.
OUTPUT: the detected preference or that no preference was found

Complexity: If n denotes the number of tuples in the log relation and k is the number of different values, the k-means clustering needs $O(k^2)$ [2] leading to the overall complexity $O(n + k^2)$. Typically, we have k << n and with it the complexity $O(n)$.

By using a state-of-the-art clustering technique like k-means – and silhouettes for getting the right number of clusters, see [12] – this algorithm detects two clusters C_1 = {Douglas Adams, Edgar Wallace} and C_2 = {Natalie Angier, Agatha Christie, John Grisham} leading to a POS(author, {Douglas Adams, Edgar Wallace}) preference in the above example. The data-driven preferences constructed in algorithm 1 are correct since the frequencies match to our requirements stated in definition 5. Data-

driven NEG preferences can only be detected, if the user knows all possible values (CWA).

In dynamic domains the CWA holds, because the user must know the varying values for his decisions. By selecting or clicking on one or more of the available values, the user provides preference knowledge since he prefers the selected items to the other available values. The following algorithm for mining such EXPLICIT preferences requires an advanced structure of the log relation. We assume we have the information (query_id, value, selected) within the log relation, whereby "value" contains a value available for the user, "selected" ($\in \{0,1\}$) denotes whether the according value was selected or not and "query_id" specifies which values belong to one search query. The ability of a low-cost construction of such log data has been shown in [4].

Algorithm 2 (Miner for EXPLICIT preferences in dynamic domains)
INPUT: log data in the format (query_id, value, selected)
(1) Compute the k occurring values $(x_1, \ldots, x_k)$ in the log relation. Initialize the better-than graph with E-graph = $\emptyset$.
(2) FOR($i = 1, \ldots, k$) and FOR($j = i + 1, \ldots, k$) DO:
   (a) Consider the query ids, whose according values contain x_i and x_j.
   (b) Compute the number s of query ids, where x_i was selected and x_j wasn't.
   (c) Compute the number t of query ids, where x_j was selected and x_i wasn't.
   (d1) If s > t and there is no path from x_j to x_i in E-graph, set E-graph = E-graph $\cup$ (x_j, x_i). Otherwise, if a path from x_j to x_i exists, remove it.
   (d2) If s < t and there is no path from x_i to x_j in E-graph, set E-graph = E-graph $\cup$ (x_i, x_j). Otherwise, if a path from x_i to x_j exists, remove it.
   (d3) If s = t remove within E-graph all direct and transitive connections from x_i to x_j and vice versa.
OUTPUT: the detected EXPLICIT preference based on E-graph as better-than graph.

Complexity: In the first step the n entries in the log relation have to be considered (O(n)). The two nested FOR-loops have the complexity $O(k^2)$. Within the loops all the tuples in the log relation need to be analyzed O(n) and a path between two given vertices ($O(k^2)$ by using Dijkstra's algorithm) has to be computed. Thus we have $O(n + k^2(n + k^2))$ as complexity. A main effort lies in the detection of inconsistencies in the user's shopping and browsing behavior. If we assume a consistent user behavior, we can avoid the path detection. The resulting complexity would be $O(n + k^2 n)$.

For two values x_i and x_j the algorithm computes the query ids that have both values in the result set. Now x_i is better than x_j, if the user selected it more often. In step (2d) cycles are removed. Therefore we check if there is a path from x_j to x_i in E-graph before inserting (x_j, x_i) and vice versa. Cycles can occur, if the browsing or shopping behavior of the user has inconsistencies like blue $<_P$ red $<_P$ green $<_P$ blue. In such situations the preferences of the customer are not clear and therefore we leave out such relations. If s = t, the user is indifferent between x_i and x_j and therefore existing preference relations between x_i and x_j have to be removed.

4.2   Mining Numerical Preferences

The distribution of numerical log data defines a statistical density function $\varphi(x)$. Properties of this density function provide information about data-driven preferences. For instance, if $\varphi(x)$ has a unique maximum at z and the gradient is positive for x < z and negative for x > z, there is an AROUND preference with around-value z. This approach is consistent to the definition of numeric data-driven preferences (def. 4) because an increasing density guarantees $freq_A([x-\varepsilon, x+\varepsilon]) < freq_A([y-\varepsilon, y+\varepsilon])$ for x < y and a decreasing density implicates $freq_A([x-\varepsilon, x+\varepsilon]) > freq_A([y-\varepsilon, y+\varepsilon])$. Thus, in the above example values around z are requested most frequently and the frequency decreases with increasing distance to z. Since the density is usually unknown, it has to be estimated using the underlying numerical log data. In our implementation we use histograms as an easy to use and efficient density estimation technique [3].

4.3   Mining Complex Preferences

Definition 6 (Data-driven Prioritized preference)
Let $P_D = (A, <_{PD})$ and $Q_D = (B, <_{QD})$ be two data-driven preferences and x = (x_1, x_2), y = $(y_1, y_2) \in dom(A) \times dom(B)$. A data-driven Prioritized preference P_D & Q_D =   ({A, B}, $<_{PQ-D}$) is defined as: x $<_{PQ-D}$ y iff $x_1 <_{PD} y_1 \vee (x_1 = y_1 \wedge x_2 <_{QD} y_2)$.

Data-driven Pareto preferences can be handled analogously [3]. In order to detect such complex data-driven preferences we need the definition of associate values.

Definition 7 (Associate Values)
Consider a log relation R(A, B, …). For a$\in \pi_A(R)$ the associate values in B are defined as $asv_{A, B}(a) = \pi^*_B(\sigma_{A=a}(R))$.

Thereby π^* denotes the relational projection without removing duplicates.

Algorithm 3 (Miner for Prioritized preferences)
INPUT: log relation R(A, B, …) and a data-driven preference P_D on A
(1) Compute the set M of maximal values of P_D and for all $a_i \in$ M the set of associate values $asv_{A, B}(a_i)$.
(2) If there is the same preference Q_D in all sets $asv_{A, B}(a_i)$ and P_D does not occur in the associate values of the maxima of Q_D, there is a Prioritized preference P = P_D & Q_D.
(3) Otherwise there is no Prioritized preference.
OUTPUT: the detected Prioritized preference or that no preference was found

Complexity: If n denotes the number of tuples, k_1 and k_2 the effort for mining P_D and Q_D, respectively, above algorithm has the complexity $O(n^2 + nk_1 + nk_2)$, since in maximal n values in A, the associate values in B are computed leading to $O(n^2)$. Furthermore, in maximal n sets the existence of the preference Q_D has to be tested ($O(nk_2)$) and vice versa ($O(nk_1)$).

A data-driven Prioritized preference P = P_D & Q_D exists, if, firstly, there is a data-driven preference P_D, and, secondly, in those tuples, which have equal values in A,

there is a data-driven preference Q_D in B. Thereby, we consider only the maximal values of P since users often don't care about a second-level preference, if the prioritized preference isn't fulfilled optimal. If P_D also occurs in the maximal values of Q_D, a Pareto preference has been found. Therefore, we have to eliminate this situation here.

In our previous example the preference P_D = POS(author, {Douglas Adams, Edgar Wallace}) was detected. If above algorithm detects Q_D = NEG(binder, {paperback}) (dom(binder) = {hardcover, paperback}) in the associate values of Douglas Adams and Edgar Wallace and, furthermore, P_D is not detected within the hardcover books, a Prioritized preference $P = P_D$ & Q_D is found.

5   Experimental Evaluation

In this section we present test results and performance measurements of an efficient database-driven implementation of a Preference Miner prototype.

5.1   Preference Mining Test Results

We performed an analysis of the Preference Mining algorithms on the log data of the COSIMA application [3]. Over five hundred users queried the COSIMA comparison shop almost four thousand times. COSIMA offers shopping in the three categories books, cds and computer products. The application server records for each query the timestamp, the shop category, the preferred price interval and – depending on the category – title and author in the book shop, title and performer in the cd shop and product name and product group in the computer hardware category. We applied the Preference Mining algorithms on the COSIMA log data, whereby we analyzed the log-data for each user separately. The Preference Miner detected lots of POS preferences and also one POS/POS preference for the shop category. Quite a few LOWEST preferences for price were detected by analyzing the lower price limit. NEG preferences and even complex preferences were also detected with the Preference Miner. Mining preferences works very fast: on a PC with 1,3 ghz and 1,5 gb main memory the Preference Miner needed less than one second to detect the preferences of a user.

Though these test results on real data show the practical usability of our techniques we cannot prove the correctness with it since the customer preferences are a priori unknown. Therefore we created synthetic log data using simulated users with predefined preference profiles. We defined 35 profiles, where each profile contains between two and six preferences, e.g. {P1 = POS(color, {blue}, P2 = LOWEST(price), P3 = P1 & P2}. In our simulation each user queries the product database between 25 to 50 times. In each query a preference of the considered user is chosen and a product database is requested with it using Preference SQL [6]. The results are stored in a log database. Afterwards we use the Preference Mining algorithms to detect preferences within the log data. A comparison of the detected preference profiles with the predefined user preferences will show the effectiveness of the Preference Mining algo-

rithms. To assess the quality of our results we define preference precision and prefer-
ence recall.

Definition 8 (Precision and recall for preferences)
Preference precision and preference recall are defined as

$$precision = \frac{number\ of\ correctly\ detected\ preferences\ of\ user\ i}{number\ of\ all\ detected\ preferences\ of\ user\ i}$$

$$recall = \frac{number\ of\ correctly\ detected\ preferences\ of\ user\ i}{number\ of\ all\ preferences\ of\ user\ i}$$

The algorithms for mining base preferences lead to a 60 % precision and a 39 % re-
call averaged over all users. This means that 60 % of the detected preferences occur
exactly in the predefined preference profiles and 39 % of the predefined preferences
are detected with our algorithms. Since a preference is regarded as correct only if all
underlying information (POS-set, NEG-set, around-value, etc.) is detected correctly, a
60 % precision and a 39 % recall are very promising. Mining complex preferences
yield to 55 % precision and 15 % recall. The poor recall here is caused by dependen-
cies between preferences: if a base preference of a complex preference is not de-
tected, it is not possible to detect the complex preference itself. Note, that we filled
the log relation with the search results. In real-life applications even better Preference
Mining results can be achieved, if the selected results or query information is used.

5.2   Performance Measurements

In this section we analyze the efficiency of the Preference Miner prototype for large
data sets. The underlying database system is an Oracle 8i database server on an AMD
CPU with 1,3 ghz and 1,5 gigabyte main memory. For our tests, we created relations
with 10,000, 20,000, 30,000, 40,000 and 50,000 tuples of synthetic data. Categorical
attributes contain 20 different categories. Numerical attributes have a data range of
200 (maximal minus minimal value). For mining complex preferences we assume one
categorical and one numerical attribute. Fig. 1 reports the average runtimes for detect-
ing a single preference for the different preference types w.r.t. the number of tuples.
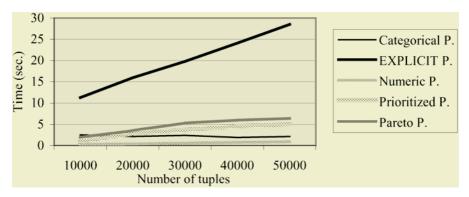

Fig. 1. Runtimes for detecting a single preference for the different preference types

Mining numerical preferences is the fastest task, since histograms can be computed very efficiently in the database layer. The miner for categorical data needs more effort since clustering is a more expensive iterative process. Mining Prioritized and Pareto preferences needs about 5 seconds in the average. The most expensive algorithm is the miner for EXPLICIT preferences (algorithm 2). The cost-intensive part is the cycle test and leads to a performance which depends linearly to the number of tuples. The efficiency of our Preference Mining algorithms allows their usage for *online Preference Mining*: while interacting with a customer an e-application can check online his preferences and react flexible to his wishes during the sales process.

6   Summary and Outlook

In this paper we have presented a novel approach for mining preferences from user log data based on the concept of strict partial order preferences. We presented several algorithms for the detection of categorical, numerical and complex preferences. Our prototype implementation executes all data-intensive operations on the database server and exhibits excellent efficiency. Our experimental results also demonstrate promising precision and recall of the detected user preferences.

Our next steps include the integration of user situations into preferences. Situations can be described with a set of parameters like current time and location, the user's role or physical and psychological condition of the user. Some preferences may only be relevant under specific situations; for example, in a bookshop a user may have different preferred categories whether he is at work or at home. A major task is the adaptation of our Preference Mining algorithms in order to detect *situated preferences*.

Another research task is the design of an appropriate storage structure for preferences. Such a *Preference Repository* should not only be able to record preferences detected with the Preference Miner but also preferences defined with Preference SQL or Preference XPATH. The integration of situations should also be possible as well as user identifiers to assign users and user groups. Finally, the Preference Repository shall also include a set of appropriate access operations for inserting, deleting and updating preferences. It can also be used to find users with similar preferences and with it product recommendations based on preferences can be offered. Therefore the Preference Repository is also a major step towards advanced personalized applications.

Acknowledgements

References

1. D. Beeferman and A. Berger: *Agglomerative Clustering of a Search Engine Query Log*. Proc. ACM SIGKDD 2000, p. 407-416, Boston, Massachusetts, USA, 2000.
2. V. Estivill-Castro and M. E. Houle: *Robust Distance-Based Clustering with Applications to Spatial Data Mining*. In 3rd Pacific-Asia Conference on Knowledge Discovery and Data Mining, p. 327–337, Beijing, China, 1999.
3. S. Holland, M. Ester and W. Kießling: *Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications*. Technical Report 2003-5, Institute of Computer Science, University of Augsburg, May 2003. http://www.informatik.uni-augsburg.de/nav/forschung.
4. T. Joachims: *Optimizing Search Engines using Clickthrough Data*. Proc. ACM SIGKDD 2002, Edmonton, Alberta, Canada, 2002.
5. W. Kießling: *Foundations of Preferences in Database Systems*. Proc. VLDB 2002, p. 311-322, Hong Kong, China, 2002.
6. W. Kießling and G. Köstler: *Preference SQL - Design, Implementation, Experiences*. Proc. VLDB 2002, p. 990-1001, Hong Kong, China, 2002.
7. S.-J. Ko, J.-H. Lee: *User Preference Mining through Collaborative Filtering and Content Based Filtering in Recommender System*. Proc. of the 3rd Intern. Conf. on E-Commerce and Web Technologies (EC-Web 2002), p. 244-253, Aix-en-Provence, France, 2002.
8. R. Kohavi: *Mining E-Commerce Data: The Good, the Bad, and the Ugly*. Proc. ACM SIGKDD 2001, p. 8-13, San Francisco, California, USA, 2001.
9. W. Lin, S. A. Alvarez, and C. Ruiz: *Efficient Adaptive-Support Association Rule Mining for Recommender Systems*. In DMKD Journal, vol. 6 (1), p. 83-105, 2002.
10. B. Mobasher, R. Cooley and J. Srivastava: *Automatic Personalization Based on Web Usage Mining*. In Communications of the ACM, vol. 43 (8), p. 142-151, August, 2000.
11. G. Rossi, D. Schwabe and R. Guimaraes: *Designing Personalized Web Applications*. Proc. 10th World Wide Web Conference (WWW 2001), p. 275-284, Hong Kong, China, 2001.
12. P. J. Rousseeuw: *Silhouettes: A Graphical Aid to the Interpretations and Validation of Cluster Analysis*. Journal of Computational and Applied Mathematics, 20, 53–65, 1987.