Comparing two Recommender Algorithms with the Help of Recommendations by Peers *

Andreas Geyer-Schulz[1] and Michael Hahsler[2]

[1] Universität Karlsruhe (TH), D-76128 Karlsruhe, Germany,
Andreas.Geyer-Schulz@em.uni-karlsruhe.de,
[2] Wirtschaftsuniversität Wien, A-1090 Wien, Austria,
Michael.Hahsler@wu-wien.ac.at

Abstract. Since more and more Web sites, especially sites of retailers, offer automatic recommendation services using Web usage mining, evaluation of recommender algorithms has become increasingly important. In this paper we present a framework for the evaluation of different aspects of recommender systems based on the process of discovering knowledge in databases introduced by Fayyad et al. and we summarize research already done in this area. One aspect identified in the presented evaluation framework is widely neglected when dealing with recommender algorithms. This aspect is to evaluate how useful patterns extracted by recommender algorithms are to support the social process of recommending products to others, a process normally driven by recommendations by peers or experts. To fill this gap for recommender algorithms based on frequent itemsets extracted from usage data we evaluate the usefulness of two algorithms. The first recommender algorithm uses association rules, and the other algorithm is based on the repeat-buying theory known from marketing research. We use 6 months of usage data from an educational Internet information broker and compare useful recommendations identified by users from the target group of the broker (peers) with the recommendations produced by the algorithms. The results of the evaluation presented in this paper suggest that frequent itemsets from usage histories match the concept of useful recommendations expressed by peers with satisfactory accuracy (higher than 70%) and precision (between 60% and 90%). Also the evaluation suggests that both algorithms studied in the paper perform similar on real-world data if they are tuned properly.

1 Introduction

Since recommender systems are becoming widely used by retailer Web sites (e.g. Amazon.com, Barnes & Noble.com), a careful evaluation of their performance gets increasingly important. However, recommender systems are complex applications that are based on a combination of several models (mathematical and psychological), algorithms, and heuristics. This complexity makes evaluation very difficult and results are

* In O.R. Zaiane, J. Srivastava, M. Spiliopoulou, and B. Masand, editors, WEBKDD 2002 - Mining Web Data for Discovering Usage Patterns and Profiles 4th International Workshop, Edmonton, Canada, July 2002, Revised Papers, Lecture Notes in Computer Science LNAI 2703, pages 137-158. Springer-Verlag, 2003

hardly generalizable, which is apparent in the literature about recommender systems (for a survey see table 2 in this paper).

In this paper we try to improve the evaluation of recommender systems by developing a more systematic approach in respect of what actually is evaluated. For this purpose we develop in section 2 a framework for the systematic evaluation of recommender systems which is in principle based on the process of knowledge discovery in databases by Fayyad et al. [1]. The expected benefit of this is that the evaluation of various aspects of a recommender system can be separated and thus more properly targeted by the different stakeholders responsible for the introduction of such a system. Therefore, our framework is more suitable for continuous process improvement by focusing on the most promising areas of improvement. In section 3 we review common performance measures for recommender system evaluation and how they are used in the recommender systems literature.

In the framework in section 2 we identified a performance evaluation method which evaluates how well the interpretation of patterns extracted by recommender algorithms matches the concept of useful recommendations given by a human peer. Although, Resnick and Varian [2] define recommender systems explicitly as systems supporting the social process of recommending products to others, this question is often neglected in the literature of recommender systems based on usage mining. To fill this gap we apply a performance evaluation method to compare two data mining methods for the generation of recommendations with recommendations by peers.

There are many possible sources for generating recommendations including expressed preferences, opinions of experts, characteristics of people and items, and many more. For recommendation services the information of all available sources should be combined to provide the best recommendations possible. However, for the performance evaluation of usage mining algorithms in this paper we restrict our recommendation generation to the case where the only available information is a collection of past usage data. The active user is anonymous and we only know the last item the user chose. This seems to be very restrictive. However, Web retailers and Internet information providers have to deal with such a situation every day since the majority of users browse the Web most of the time anonymously and still services like recommendations right after the first click are needed to improve sales. In sections 4 and 5 we give a brief introduction to the two data mining algorithms used in the paper. The first algorithm uses the in the knowledge discovery in databases (KDD) society well-known support-confidence framework, the second algorithm is based on Ehrenberg's repeat-buying theory originating from marketing research. In section 6 we describe the experimental setup and the data set. In section 7 we present and discuss the evaluation results. We conclude with a short summary of the findings and open research questions in section 8.

2 A Framework for the Evaluation of Recommender Systems

Recommender systems are complex applications which have to perform the whole process of knowledge discovery in databases (KDD process). In [1] Fayyad et al. give an overview of the steps of the KDD process. An application of this model to recommender systems is straightforward. In order to separate the influence of the user interface from

the effects of the choice of a data mining method we add *presentation* as additional step to the process model of Fayyad et al. (see figure 1). The five major steps are:

1. *Selection:* The data set used to produce recommendations can stem from various sources. For example these sources can be already existing transaction logs (e.g. point of sale data, Web server logs) or the data can be collected especially for the purpose of generating recommendations (e.g. ratings for movies).
2. *Preprocessing and transformation:* In these steps the data set is cleaned from noise, inconsistent data is removed, and missing data is inferred. After this treatment the cleaned data is transformed into a representation suitable for data mining.
   For example, for collaborative filtering the data normally consists of explicit ratings by users which are collected for the purpose of creating recommendations (e.g. for music see [3]). Preparation mainly involves discovering and removing inconsistent ratings.
   For Web usage mining (see [4, 5]) data is collected by observing the behavior of users browsing a Web site. Since observation, especially server-side observation on the Web, is far from perfect, much effort has to be put on data preprocessing, cleaning and transformation. Problems involve the identification of users, dynamic (rotating) IP-addressing, session identification, missing data due to proxy servers and system crashes, requests by Web robots – to name just a few.
3. *Data mining:* The objective of this step is to find interesting patterns in the data set that are useful for recommendation purposes. The output of data mining in recommender systems can be: groups of users with similar interests, items that are frequently used together, often used sequences of items,... Frequently, extracting patterns means learning the parameters of a specified model from the data set.
4. *Interpretation and evaluation:* In order to build knowledge, the found patterns (the model and its parameters) have to be understandable to humans. Only with this property the process can be called knowledge discovery and the results can be interpreted. A recommender system interprets found patterns for the user.
   Finally the validity (patterns are also valid for new data), novelty (involves a new way of finding patterns), usefulness (potentially lead to useful action), and understandability (build and increase knowledge) of the patterns need to be evaluated.
5. *Presentation:* A recommender system presents this interpretation in a suitable form as a recommendation. There are many presentation options. For example, the recommendation can be a top-n list of recommended items for a user, or a list of items that are similar to an item the user likes, or it can consist of information about how other users with similar interests rated a specific item.

Wirth and Hipp included in their process model for data mining called CRISP-DM [6] a step called deployment. In this step they emphasize the need to deliver the results or even a ready-to-use implementation of the data mining process to the customer. Their definition is similar to software engineering, where the deployment phase usually includes the actual introduction and operation of a system in an organization. Developing an evaluation methodology for the deployment phase of a recommender system is beyond the scope of this paper.

Because of the complexity of recommender systems, and the many available choices for each step of the knowledge discovery process described above, detailed evaluation

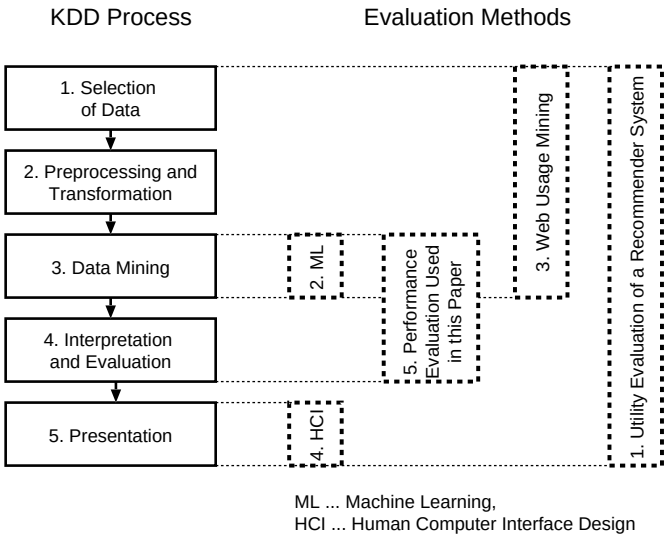

Fig. 1. Mapping evaluation methods to the steps of the KDD process

of the different aspects of this process is a necessity. In figure 1 we mapped five evaluation methods for recommender systems to the steps of the KDD process to provide a systematic reference framework. We summarize these evaluation methods in the following:

1. The evaluation of the utility of a recommender system is targeted to the stakeholders of the organization. The evaluation spans the whole process and assesses the utility of process as a whole as indicated by the right-most evaluation process labeled "1. Utility Evaluation of a Recommender System" in figure 1. In practice the utility of the whole recommender system is usually evaluated by the impact of the recommender system on the overall performance of the process the recommender supports. If the process supported is the purchase process in a supermarket, the impact is measured by comparing sales in two test markets, one with and one without the recommender system. This approach is used in Lawrence et al. [7] for the evaluation of a personalized product recommender on personal digital assistants (PDAs) for the Safeway supermarket chain. The performance measure used for evaluation is the revenue from the sales volume triggered by the recommender. Lawrence et al. reported that they observed an increase in revenue of about 1.8% after introducing the recommender to a new store. This is in line with NetPerception's estimation of a 2.0% increase in sales volume [8]. NetPerception measures this by comparing the impact of random product lists versus recommended product lists on the buying behavior of the consumer. For non-profit sites, such an overall performance measure could be a conversion rate, a contact rate or a task achievement rate.
2. Most researchers in recommender systems focus on the evaluation of mining algorithms with methods known from *machine learning.* A common way from ma-

chine learning for comparing the performance of several algorithms is to use a prepared data set and divide it into a set for training and a set for evaluation (cross-validation is often used). This approach only takes into account how well patterns in the data set can be learned and not how useful the patterns are for recommendation purposes. However, in addition, the underlying theoretical framework of these algorithms must be evaluated with respect to its consistency. For association rule algorithms, a recent discussion can be found in Adamo [9, pp. 151-184]. For collaborative filtering based on explicit product ratings a comparison with regression models and a survey of studies in this area can be found in [10]. Furthermore, for model-based approaches the correspondence of the model with reality must be evaluated. For statistical models this is done by testing the underlying behavioral assumptions in a piece-meal fashion, by diagnostic-checking. For the repeat-buying theory used below in this paper, see e.g. [11]. Model comparison is performed along several dimensions, most notably performance, robustness, parsimonity of parameters, sensitivity to misspecification of parameters, computational complexity, ease of use and understandability.

3. *Web usage mining* covers the first three steps of the KDD process. In addition to the evaluation of the data mining method evaluation of the data selection and preprocessing steps is important for Web usage mining [4, 5]. Correctly observing user behavior on the Web requires automatically handling difficulties e.g. with filtering automatic Web robots [12], detecting sessions [13] and path completion [14]. For example, for evaluating session identification heuristics Cooley recommends comparing the results of log files with session identification (e.g. by cookies or by server-side session identifiers embedded in the link) with the results of analyzing the same log files stripped (see [14, pp. 118-123]). Another evaluation method is testing preprocessing statistics on synthetic data. For episode identification this approach has been used by Cooley [14]. Berendt et al. [15] define quality measures for session reconstruction from log files and compare different heuristics based on session duration, time spent on a page and referrer values. In their analysis they found considerable performance differences between the heuristics depending on the structure of the Web site.
   However, while these studies ([14] and [15]) make a considerable contribution to the evaluation of Web usage mining, they are still based on real data sets. The evaluation of the quality of preprocessing heuristics is still biased from measurement errors (e.g. unidentified robots, ...) and on assumptions on actual use and spread of technology (e.g. dynamic IP addresses). Therefore, the construction of synthetic data sets with known properties (or the use of fully instrumented Web sites and browsers in a controlled environment) and subsequent masking of information as evaluation suites for Web usage mining methods would be of considerable value. With the coming of ad-hoc networking and a considerable growth in mobile, embedded and wearable devices progress in this direction is necessary.
4. Evaluation of the presentation of recommendations to the consumer/user is dealt with in the area of *human-computer interface (HCI)* research and includes methods such as usability labs and field-experiments. In connection with recommender systems Herlocker et al. [16] compared 21 different representations of recommendations for movies. The findings were that – similar to previous experiences with

Table 1. 2x2 confusion matrix

actual / predicted	negative	positive
negative	*a*	*b*
positive	*c*	*d*

expert systems – suitable explanation of the recommendations to the users increases the acceptance of the recommender system.

5. *The performance evaluation of recommender systems used in this paper* evaluates how well the interpretation of patterns extracted by a recommender algorithm matches the concept of useful recommendations given by a peer. This evaluation combines the data mining step as well as part of the interpretation step of the KDD process shown in figure 1. Since such a performance evaluation is not common in the very machine learning oriented recommender literature, we provide a evaluation of a simple association rule algorithm and a novel repeat-buying based algorithm in this paper.

3 Performance Measures for Recommendation Algorithms

For measuring the performance of recommender algorithms measures originating from statistics, machine learning, and information retrieval are used. To give definitions of the performance measures we first have to define the meaning of the terms item, recommendation, and recommendation list. *Items* are the products under consideration that can be purchased (consumed/used) by the customer (user). A *recommendation list* is a list of items which the algorithm recommends for a user in a certain situation. We call each item in the list a *recommendation* which can be either correct or incorrect.

Since all measures use similar information it is useful to define them with the help of the so called *confusion matrix* depicted in table 1 (see [17]) which corresponds exactly to the outcomes of a classical statistical experiment. The confusion matrix shows how many of the possible recommendations were predicted as recommendations by the algorithm (column predicted positive) and how many of those actually were correct recommendations (cell *d*) and how many not (cell *b*). The matrix also shows how many of the possible recommendations the algorithm rejected (column predicted negative), were correctly rejected (cell *a*) or should have actually been recommended (cell *c*). Statistically we test the hypothesis H_0 that an item is a recommendation (positive in table 1) against the hypothesis H_1 that an item is not a recommendation (negative in table 1). Cell *c* is known as type I error with probability α and cell *b* is known as type II error with probability β.

Performance measures from machine learning. For the data mining task of a recommender system the performance of an algorithm depends on its ability to learn significant patterns in the data set. Performance measures used to evaluate these algorithms have their root in machine learning.

Commonly used measures are accuracy and coverage. *Accuracy* is the fraction of correct recommendations to total possible recommendations (see formula 1). *Coverage*

measures the fraction of items the system is able to provide recommendations for (see formula 2). We can not define coverage directly from the confusion matrix, since the confusion matrix only represents information at the level of recommendations (relationships between items) and not at the level of individual items with recommendation lists.

$$Accuracy = \frac{correct\ recommendations}{total\ possible\ recommendations} = \frac{a+d}{a+b+c+d} \tag{1}$$

$$Coverage = \frac{items\ with\ recommendations}{total\ number\ of\ items} \tag{2}$$

A common error measure is the *mean absolute error (MAE, also called mean absolute deviation MAD)* shown in formula 3. N is the length of the learned pattern from the training set (the total number of items for which recommendations are produced = items with recommendations in formula 2) and $|\varepsilon_i|$ is the absolute error of each component (number of incorrect classifications in the recommendation list for each item) of the pattern compared to the evaluation set.

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|\varepsilon_i| = \frac{b+c}{N} \tag{3}$$

Performance measures from information retrieval. Recommender systems help to find items of interest from the set of all available items. This can be seen as a retrieval task known from information retrieval. Therefore, standard information retrieval performance measures are frequently used to evaluate recommender performance.

Precision and *recall* are the best known measures used in information retrieval [18, 19] (see formula 4 and 5 for the definitions).

$$Precision = \frac{correctly\ recommended\ items}{total\ recommended\ items} = \frac{d}{b+d} \tag{4}$$

$$Recall = \frac{correctly\ recommended\ items}{total\ useful\ recommendations} = \frac{d}{c+d} \tag{5}$$

Often the number of *total useful recommendations* needed for recall is unknown since the whole collection would have to be inspected. However, instead of the actual *total useful recommendations* often the total number of known useful recommendations is used as an estimate.

Precision and recall are conflicting properties, high precision means low recall and vice versa. To find an optimal trade-off between precision and recall a single-valued measure like the *E-measure* [19] can be used. The E-measure is defined in formula 6. The parameter α controls the trade-off between precision and recall.

$$E\text{-}measure = \frac{1}{\alpha(1/Precision) + (1-\alpha)(1/Recall)} \tag{6}$$

A popular single-valued measure is the *F-measure.* It is defined as the harmonic mean of precision and recall given in formula 7. It is a special case of the E-measure

with $\alpha = .5$ which places the same weight on both, precision and recall. In the recommender evaluation literature the F-measure is often referred to as the measure *F1*.

$$F\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} = \frac{2}{1/Precision + 1/Recall} \tag{7}$$

Other performance measures used for recommender systems. Other measures are often model dependent like *r-square* and *root mean squared error (RMSE)* for regression models using ratings. Various *hit rates* are also possible, comparing user ratings with the computed recommendations.

A measure used in the literature to compare two algorithms or parameter settings is the *Receiver Operating Characteristic* (ROC). It is a measure used in signal detection and goes back to the Swets model [19]. The ROC-curve is a plot of the system's *probability of detection* (also called *sensitivity* or true positive rate or recall as defined in formula 5) by the *probability of false alarm* ($1 - specificity$, where $specificity = \frac{a}{a+b}$ which is the true negative rate) with regard to model parameters. A possible way to compare the efficiency of two systems is by comparing the size of the area under the ROC-curve, where a bigger area indicates better performance.

In the context of real-time personalization for Web sites Mobasher et al. introduced in [20] variants of performance measures which evaluate a recommendation list R with regard to a single user session t ("a transaction") and the window $w \subseteq t$ used for production of the recommendation list R. $\mid t \mid$, $\mid w \mid$, and $\mid R \mid$ denote the sizes of these sets. The variants for precision and coverage are:

$$Precision(R,t) = \frac{\mid R \cap (t-w) \mid}{\mid R \mid} \qquad Coverage(R,t) = \frac{\mid R \cap (t-w) \mid}{\mid t-w \mid} \tag{8}$$

In addition, they propose the R-measure (coverage divided by the size of R) which favors smaller recommendation lists.

In table 2 we summarize recent papers that evaluated recommender algorithms using the presented measures.

4 A Simple Recommender using Association Rules

The first recommender we use is based on association rules with thresholds for minimum support and minimum confidence. This is known as the support-confidence framework. The problem of finding association rules for market basket data that satisfy minimum support and minimum confidence was first presented by Agrawal et al. [27]. Association rule algorithms require no model assumptions. This makes the approach very attractive and simple because checking whether the model assumptions are met is not required.

The problem is formalized in [27] as follows: Let $I = \{i_1, i_2, ..., i_n\}$ be a set of items. Let D be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. A transaction T contains X if $X \subseteq T$ and Y if $Y \subseteq T$. An association rule is an implication in the form $X \Rightarrow Y$, where $X \subset I$, $Y \subset I$, and X, Y disjoint. X is called the antecedent and Y is called the consequent of the rule.

Table 2. Recommender algorithm evaluation papers

Paper	Domain	Algorithm	Measures
Shardanand and Maes [3]	Music ratings	Prediction algorithms based on similarity between user profiles	MAE
Herlocker et al. [21]	Movie ratings	Neighborhood based prediction algorithms	MAE, ROC, Coverage
Sarwar et al. [22]	Movie ratings, E-Commerce purchases	Dimensionality reduction	MAE, F-measure
Mobasher et al. [23]	Web site usage	Aggregate user profiles (clustering user transactions and pageviews)	Accuracy
Vucetic and Obradovic [24]	Movie ratings	Regression based item-to-item relationship	MAE, ROC
Yu et al. [25]	Movie ratings	Instance selection for collaborative filtering	MAE, Accuracy
Mobasher et al. [20]	Web site usage	Aggregate user profiles (clustering user transactions and pageviews)	Precision, Coverage, F-measure, R
Lin et al. [26]	Movie ratings	Adaptive-support association rule mining	Accuracy, Precision, Recall
Mild and Natter [10]	Movie ratings	Collaborative filtering, linear regression models e.g. with model selection	MAE, RMSE, R-square, hit-rate

For association rules the thresholds of the two measures – minimum support and minimum confidence – are commonly used to identify a subset of rules for detailed analysis or as input for a recommender application. Support is a measure of statistical significance. The rule $X \Rightarrow Y$ has support sup in the transaction set D if more than sup% of transactions in D contain X and Y together. Confidence is a measure of strength. The rule $X \Rightarrow Y$ holds in the transaction set D with confidence $conf$ if $conf$% of transactions in D that contain X also contain Y. In formula 9 and formula 10 support and confidence are defined by probabilities.

$$sup(X \Rightarrow Y) = sup(Y \Rightarrow X) = P(X \wedge Y) \tag{9}$$

$$conf(X \Rightarrow Y) = \frac{P(X \wedge Y)}{P(X)} = \frac{sup(X \Rightarrow Y)}{sup(X)} \tag{10}$$

Agrawal and Srikant presented in [28] the APRIORI algorithm to efficiently compute association rules with several items in the antecedent of the rule using frequent itemsets. A frequent itemset is a set of items that satisfy minimum support in the set of transactions. The algorithm generates larger frequent itemsets with every pass by combining frequent itemsets from the last pass and pruning away the itemsets without sufficient support. The algorithm stops when no larger itemset can be produced without falling under minimum support. Then all large itemsets (frequent itemsets that could not be combined to larger frequent itemsets) are used to produce the association rules.

For the simple recommender system used in this paper we only need association rules with one single item in the antecedent of the rule. Then for the item in the an-

tecedent of each rule we use the items in the consequent as the recommendation list. The number of recommendations in the list can be varied by changing the support and confidence thresholds.

Recently support and confidence were subject to criticism [29, 30]. The main point of criticism is that confidence ignores the frequency of the consequent in the data set and therefore spurious rules with items in the consequent that appear often in the data set cannot be separated from more accurate rules. *Lift* [31], also known as *interest* [29, 30] is an alternative measure for confidence that takes the frequency of the consequent into account. But in contrast to implication it measures only the co-occurrence of X and Y as a symmetric measure. Lift is defined as the relation of the (observed) probability of the co-occurrence of two items to the probability under the assumption that they occur independently. The definition of lift for the rules $X \Rightarrow Y$ and $Y \Rightarrow X$ is given in formula 11.

$$lift(X \Rightarrow Y) = lift(Y \Rightarrow X) = \frac{P(X \wedge Y)}{P(X)P(Y)} = \frac{conf(Y \Rightarrow X)}{sup(X)} \tag{11}$$

Another alternative measure for confidence is *conviction* (see [29] and [31]). Conviction measures the deviation of the implication $X \Rightarrow Y$ from the assumption that X and Y occur independently. It is derived from the fact that the implication $X \Rightarrow Y$ can logically be reformulated as $\neg(X \wedge \neg Y)$. If we divide this by the individual probabilities $P(X)$ and $P(\neg Y)$ and invert the ratio to compensate for the outside negation we reach the definition of conviction as given in formula 12.

$$conviction(X \Rightarrow Y) = \frac{P(X)P(\neg Y)}{P(X \wedge \neg Y)} = \frac{1 - sup(Y)}{1 - conf(X \Rightarrow Y)} \tag{12}$$

Although in the literature conviction is claimed to be superior to confidence [29] most papers still use the support-confidence framework. For an extensive survey of variants of association rule algorithms and a discussion of their intrinsic short-comings, see Adamo [9]. We will report results for confidence, lift, and conviction in this paper.

5 A Simple Recommender using the Repeat-buying Theory

The second recommender algorithm is based on the repeat-buying theory introduced by Ehrenberg [11]. In the context of this paper, buying an item means to visit a Web site. The idea behind the repeat-buying theory is that in a stationary situation with the purchases of all items being independent from each other the buying behavior follows a process that produces a specific distribution for the number of repeat buys, namely the negative binomial distribution (NBD). The statistical model is based on several strong behavioral assumptions about consumer purchase behavior. Although these assumptions invite criticism of the model, Ehrenberg [11] and other authors empirically showed that this model holds for various consumer markets.

In [32] we showed how to apply a simplified form of the model (using the logarithmic series distribution (LSD), a limit case of the NBD also described in [11]) to generate recommendations for the Internet information broker used in this paper. In this setting the LSD in formula 13 gives the probability of the observation that the same

Table 3. Algorithm for computing recommendations based on repeat-buying

Algorithm GenerateRecommenderLists

for each item x in all transactions **do** {

1. generate the frequency distribution of co-occurrences from all transactions containing the item x
2. approximate the (only) parameter q of the LSD distribution from the mean w of the frequency distribution
3. generate an empty recommendation list for item x
4. **while** the expected type II error rate β is below a set threshold **do** {
   (a) add the item with the highest number of co-occurrence to the list of recommendations
   (b) compute the expected type II error rate for the new list }
5. store the recommendation list }

pair of two independent items are used together in a specified period of time a total of 1, 2, 3, ..., r times by pure chance. q is the only parameter of the distribution and can be easily estimated. First we calculate the mean of the observed usage frequency w for all item pairs that contain one specific item. And then we approximate q from w by their relationship stated in formula 14.

$$P(r) = \frac{-q^r}{r\ln(1-q)}, \quad r \geq 1 \tag{13}$$

$$w = \frac{-q}{(1-q)\ln(1-q)} \tag{14}$$

In [32] we showed that the LSD model can be fitted to co-occurrences of information products in our information broker reasonably well. However, the LSD model is only applicable to co-occurrence of items under the strict assumption of independence between the usage of all items. Of course, this assumption is not valid for some items in the data, caused by dependencies between the usage of items that cover the same topic or that are useful as complementary information sources. For the recommender system we need to find the items with dependencies. Since the dependencies between items violate the strict independence assumption of the model outliers that do not follow the fitted LSD are produced. For each pair of items we can calculate the probability that it comes from the LSD model by dividing the observed number of co-occurrences by the predicted number (using $P(r)$ from formula 13). This results is an estimate of the type II error for each possible recommendation.

The algorithm in table 3 produces for each item x a recommendation list from the co-occurrences with different y_i that has an expected type II error rate (β) below a predefined threshold. By changing this threshold, recommendation lists with higher or lower expected β and potentially more or less recommendations can be produced by the algorithm.

The algorithm produces similar results as association rules using variable confidence and support thresholds. If we set support to 0 and we order all association rules

with the same antecedent x by confidence we get the same set of co-occurrences that is used to calculate the LSD. Since confidence preserves the rank order of the number of co-occurrences with the same antecedent the selection of recommended items is made in the same sequence for both algorithms. The difference between the algorithms is the way how the threshold for the recommended items is calculated. For association rules recommendations consist of rules that satisfy a predefined minimum support and confidence threshold, where support is calculated over all transactions. An exception to this is the approach of Lin et al. [26] which requires the specification of a target range for the number of recommendations offered to a specific user and which adapts the support threshold so that a rule set with the desired size is generated. For the repeat-buying algorithm the set of recommendations is selected for each item x individually from the frequency distribution of co-occurrences, so that the total expected type II error rate (in comparison with the fitted model) is below a set threshold. In association rule terminology this procedure amounts to automatically finding for all rules with the same antecedent an individual confidence threshold that satisfies the error constraint.

Association rule algorithms require the computation of support and confidence (or lift or conviction) and the specification of two threshold parameters for these. The computational complexity is in general of exponential order. For the simple association rules with only one item in the antecedent used in this paper, the computational complexity is of quadratic order.

Production recommender systems in organizations require periodical updates. For association rule algorithms updating techniques which consider the effects of the update on all current association rules are presented in [33], [29], and [34]. Cheung et al. [33] design an updating algorithm which exploits necessary conditions on the support of an itemset in the update increment to be a large itemset to efficiently identify winners (itemsets that become large itemsets after the update) and loosers (itemsets which cannot become large itemsets after the update) and thus realize considerable improvements in efficiency. Nevertheless, all support and confidence values of all association rules must be recomputed. Brin et al. [29] discuss the handling of incremental updates for finding large itemsets as a potential, but straightforward extension of the DIC algorithm. Unfortunately, however, this helps only in computing large itemsets, support and conviction have to be recomputed for all association rules. Ng and Lam [34] improve the work of Cheung et al. and Brin et al. with the help of sophisticated dynamic itemset counting techniques. However, similar to the other approaches, all effort is invested in identifying large itemsets, all support and confidence values must be recomputed. Therefore, the update of the confidence and support of all current association rules requires a complete update of all support and confidence values which is of quadratic order in the number of items.

Like the simple association rule algorithm used in this paper the computational complexity of the repeat-buying algorithm is $O(n^2)$, where n is the number of items. However, an incremental update version of the algorithm is easy to implement. The algorithm simply requires an update of the count of all co-occurrences in the increment. Only for actually updated items LSD-models must be recomputed and the complexity is reduced to $O(n_u^2)$ with n_u the number of items updated. The reason for this is that the theoretical LSD depends only on the sample mean of all co-occurrence counts with

the same item (see equation 14) and not on any global parameter of the dataset. This is an important advantage over simple association rule algorithms which gets more pronounced as the ratio of the total number of items to the number of items used between two updates increases. A real-world example where we benefit from this difference is computing recommendations for the users of a large research library's online public access catalog (OPAC) with millions of items and $n_u \ll n$.

6 Experimental Setup

In this section we compare the performance of the two recommender algorithms for the data mining step as well as for the interpretation and evaluation step of the KDD process. Most papers (except [7] and [8]) only evaluate the data mining step using techniques from machine learning. The evaluation methods in [7] and [8] are not applicable for non-profit organizations as e.g. universities, research libraries, and government agencies because of missing data on sales volume or profits. However, Resnick and Varian's seminal paper [2] defined recommender systems as systems supporting the social process of recommending products to others – as systems supporting the word-of-mouth effect well known in marketing. Having this definition in mind, it is a natural question if the recommendations produced by a system are similar to the recommendation produced by a peer (a person in the target group of the system). Therefore, we asked for each item in a recommendation list the question if the interviewed person would recommend it to a person interested in the item the list was produced for. The advantage of this approach is that it does not require the setup of test markets (which are quite costly to run) and that it can easily be applied in non-profit organizations, too. In addition, by judiciously controlling the subjects in the sample – a sort of control not yet sufficiently exercised in this paper – tuning recommender systems to heterogenous target groups may be improved.

For the study we use a data set from the Virtual University information system at the Wirtschaftsuniversität Wien. This information system is an educational Internet information broker that provides access to online information products (e.g. lecture notes, research material and enrollment information) for students and researchers. An information product in this setting is defined as a set of Web pages (including other media like word processor files) that present a logical unit of information on a specific topic, e.g. a Web site on software development. The information products are distributed all over the Internet and the information broker logs the mediations of the products at the application level. The transaction data is anonymous but includes a cookie based session management and Web robots are removed.

To generate recommendations for the information system we provide the recommender algorithms with a data set containing market basket data obtained from the log of the information broker. A market basket is the set of all items (products) used together in a single transaction (a session). We use 6 months of usage data (January to June 2001) containing 25522 transactions with more than one item. These transactions contain 3774 different items with 114128 co-occurrences of items that are possible recommendations.

For evaluation we produced for each item a list of all other items that co-occurred together with the first item in at least one transactions and randomly selected 300 lists. For each selected list we produced a computer-based questionnaire with the list of items in randomized order. The items were represented in the questionnaire with their names and we also provide the links to the corresponding Web pages so the interviewed person could browse through the actual information. For each item we asked the question if the interviewed person would recommend it to a person interested in the item the list was produced for. The possible answers were ”recommend” or ”don’t recommend” which correspond to the perceived usefulness of a possible recommendation. We asked people from the information system’s target group (6 students, 1 system administrator, 1 secretary, and 5 researchers from the Universität Karlsruhe (TH)) to evaluate each an exclusive subset of the lists. The number of possible recommendations in each subset was approximately equal, so that no single subject had a dominating influence on the evaluation set. Exclusive subsets implies that inter-rater reliability can – unfortunately – not be assessed and compared with the performance of the algorithms. In addition, subjects were not randomly selected. However, any uncontrolled effects because of the missing randomization of the subjects applies to both algorithms in the same way.

In total 1661 possible recommendations (co-occurrences of items) were evaluated. For 561 co-occurrences the interviewed persons said they would recommend the latter item, and for 1100 they would not. More than 120 lists are of the size 2 (the smallest size analyzed) and only few lists have a size larger than 50. This is in line with the characteristics of other real-world datasets used in Zheng et al. [35]. After visual inspection the proportion of good items in the lists seems to be independent from the size of the list with an average of 0.338.

7 Evaluation Results

In this section we present and discuss the evaluation results for the different recommender algorithms. The algorithms used the data set described above as input and the produced recommendations were evaluated using the useful recommendations identified by the users.

To find sensible support values for the association rules based algorithms, we varied minimum confidence between 0.3 and 0.000001 (0.3, 0.2, 0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001) and plotted precision by recall for several support thresholds. Figure 2 depicts the precision/recall plots for the best performing support thresholds. In the classic setting for association rules, e.g. in a supermarket setting, the analyst is interested in a manageable, very small set of rules with high support. In this setting high support means to only consider products with high sales and therefore to justify the effort to physically rearrange the products in the market. However, for an association rule algorithm producing only a small set of rules means low recall and low coverage. In an on-line recommender system there is no (or very small) additional cost for displaying more (useful) recommendations in the interface. Therefore, we can also efficiently present recommendations for products with low support – as long as those recommendations are useful. To obtain reasonable recall for our data set, the minimum support threshold has to be chosen relatively small with values between 0.001 and 0.00001 (see

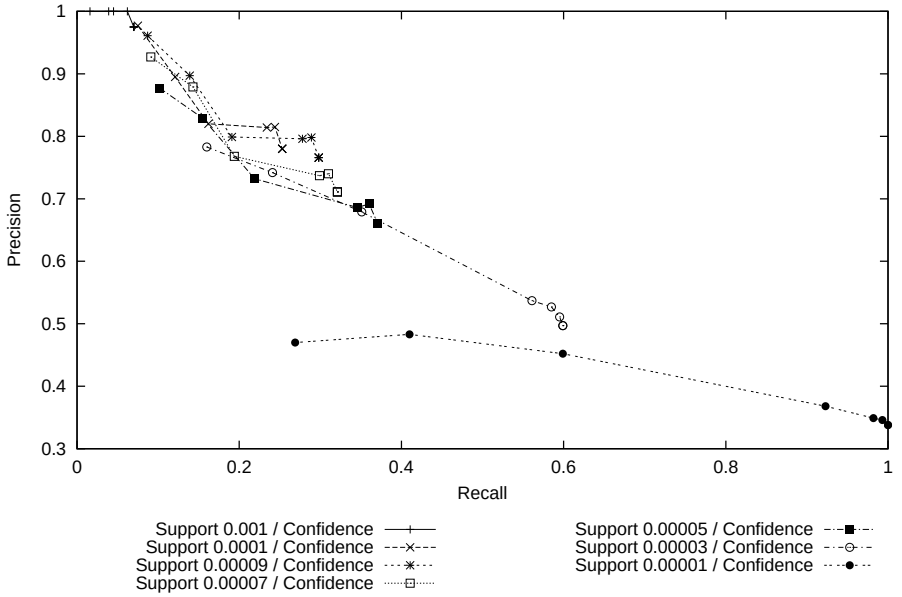

Fig. 2. Precision/recall plot for association rules with different minimum support

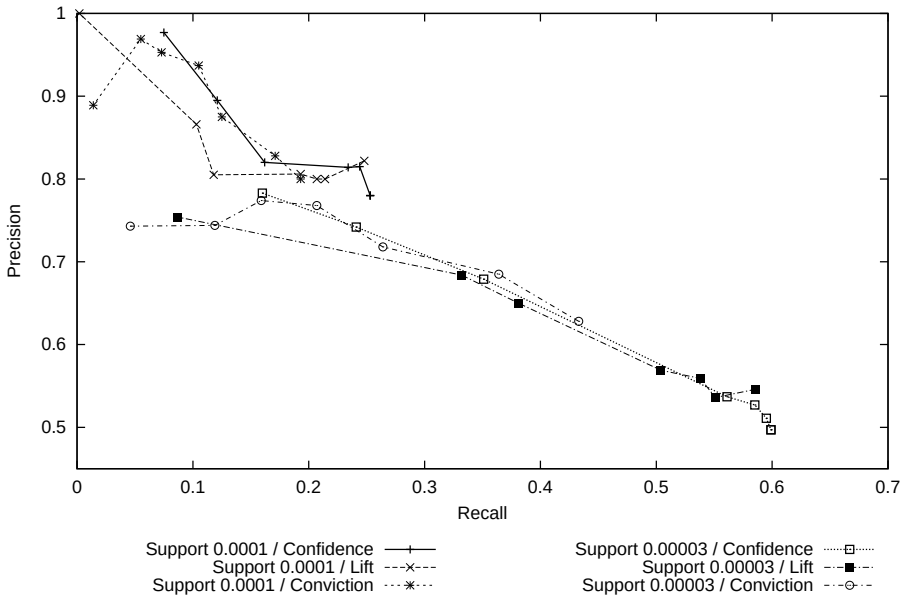

Fig. 3. Precision/recall plots for confidence, lift, and conviction

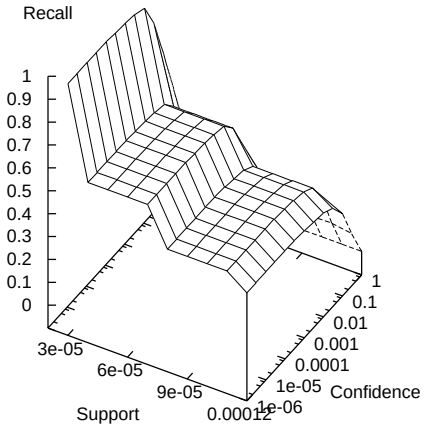

Fig. 4. Recall for association rules by support and confidence threshold

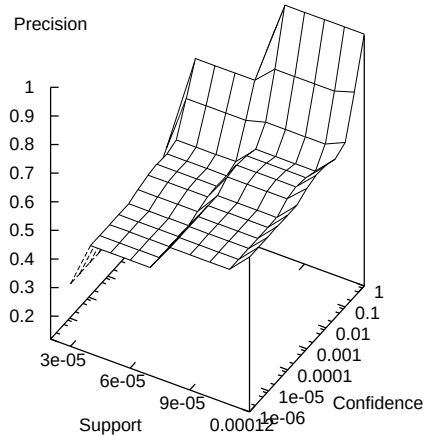

Fig. 5. Precision for association rules by support and confidence threshold

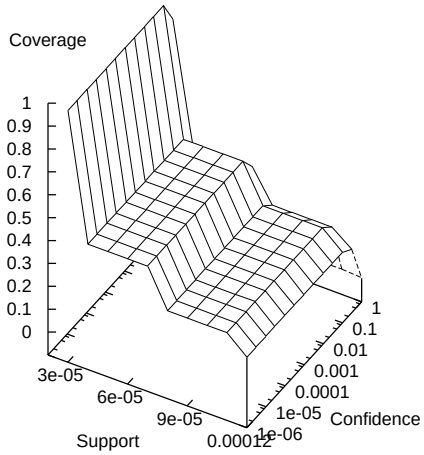

Fig. 6. Coverage for association rules by support and confidence threshold

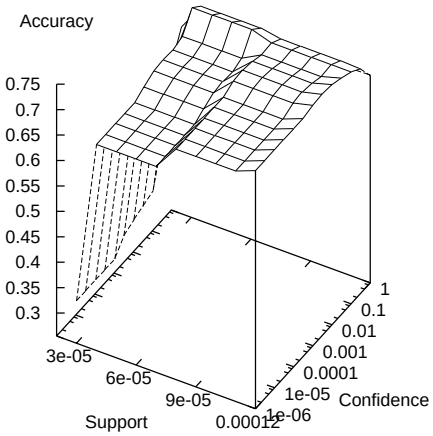

Fig. 7. Accuracy for association rules by support and confidence threshold

Table 4. Results of the repeat-buying algorithm for different threshold values

Threshold	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
Precision	0.78	0.87	0.83	0.79	0.76	0.77	0.7	0.67	0.66	0.61	0.52
Recall	0.01	0.17	0.22	0.25	0.28	0.32	0.34	0.36	0.41	0.44	0.62
Accuracy	0.67	0.71	0.72	0.72	0.73	0.74	0.73	0.73	0.73	0.72	0.68
Coverage	0.02	0.13	0.17	0.19	0.22	0.23	0.25	0.31	0.38	0.42	0.44
Avg. type II error rate	0.17	0.13	0.22	0.23	0.26	0.26	0.3	0.33	0.33	0.36	0.46

figure 2). This is due to relatively short transactions and the high number of items in the data set.

Next we compare the performance of confidence with the alternative measures of interestingness, lift, and conviction. For recommender systems recommendation lists with low precision are problematic since recommending unrelated items annoys the users. Since reasonable precision starts with 0.5 and up, we use only minimum support of 0.0001 and 0.00003 (see figure 2) for the comparison. Figure 3 shows the precision/recall plots for the two selected minimum supports. For the plots we varied lift between 1.5 and 1000 and conviction between 1.05 and 2. Since neither lift nor conviction perform significantly better than confidence on the data set we use the support-confidence framework for all further investigations.

Figures 4 to 7 show the surface plots of recall, precision, coverage, and accuracy by minimum support and confidence (confidence is plotted on a logarithmic scale). Note, that these measures must be optimized simultaneously. Choosing a certain combination of support and confidence determines all four measures simultaneously, not every combination of performance levels (e.g. a recall larger than 0.9 and a precision larger than 0.9) can be attained by the algorithm. As expected, recall (see figure 4) and coverage (figure 6) decrease with increasing minimum support, but stay almost constant for most of the plotted range of minimum confidence. Only for confidence values bigger than 0.01 (0.1 for coverage) the measures decrease fast. With support the measures decrease in several steps, indicating that many rules have the same level of support. Precision (see figure 5) and accuracy (figure 7) show a less stable behavior. Precision decreases with smaller minimum support and minimum confidence and strives to 1 for very high minimum confidence (producing very few rules). However, there is unevenness at the same level of support which is also visible in the accuracy plot. Precision and accuracy deteriorate fast for very small values of minimum support (< 0.00003). Between minimum support of 0.00006 and 0.00007 appears an abrupt step, indicating that below this level many association rules with a high error rate are accepted by the algorithm. Minimum confidence has less influence on accuracy than minimum support, however, the confidence level for maximum accuracy changes with the set minimum support between 0.03 for higher support (> 0.00007) to 0.3 for lower support (< 0.00004), indicating that there is a strong interdependence between both parameters and the structure of the data set.

For the repeat-buying algorithm we varied the threshold between 0 and 1 and calculated the same quality measures as above. Table 4 and figures 8 and 9 contain the results. With a higher threshold the algorithm becomes less selective and more recom-

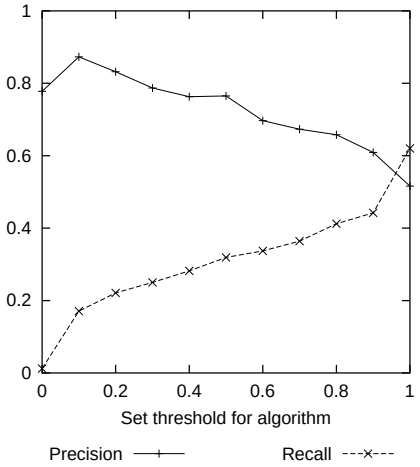

Fig. 8. Precision and recall by the threshold of the repeat-buying algorithm

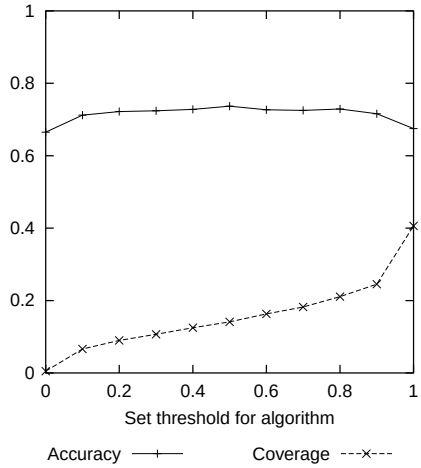

Fig. 9. Accuracy and coverage by the threshold of the repeat-buying algorithm

mendations are produced. Therefore, recall and coverage are rising with an increasing threshold and precision is decreasing. With only a few exceptions (for small threshold values, where only few recommendations are generated) these quality measures change monotonously with the threshold. Accuracy is almost constant at a level around 0.7 which means that with higher thresholds the type I error decreases at a similar rate as the type II error increases.

In figures 10 and 11 we compare the performance of the association rule algorithm and the repeat-buying algorithm in terms of precision by recall and accuracy by coverage. For comparison, we included in both plots a recommender that chooses recommendations randomly from the co-occurrence list with the probability varying between 0 and 1. Both recommender algorithms perform significantly better in predicting the items that users qualify as useful recommendations than choosing recommendations randomly. The repeat-buying recommender performs similar to the association rule algorithm with the support-confidence framework. Figure 10 shows that at reasonable recall (between 20% and 50%) both algorithms reach a precision between 60% and 80% which is acceptable for most applications. Both algorithms provide accuracy above 70% (see figure 11), however, the support-confidence framework is very brittle with respect to changes of minimum confidence and, what is more problematic, the optimal value for the confidence threshold changes with minimum support (from 0.001 for a support of 0.0001 to 0.1 at 0.00003).

Figure 12 shows the mean absolute error of the recommendations by the coverage produced by the algorithms for different parameters. Again, for the support-confidence framework performance deteriorates significantly for very small misspecifications of the confidence threshold. The repeat-buying algorithm shows a more robust behavior with respect to changes to its one threshold which represents the maximum expected type II error rate β in the produced recommendation lists. In contrast to minimum support and minimum confidence, β is independent of the structure and the properties of

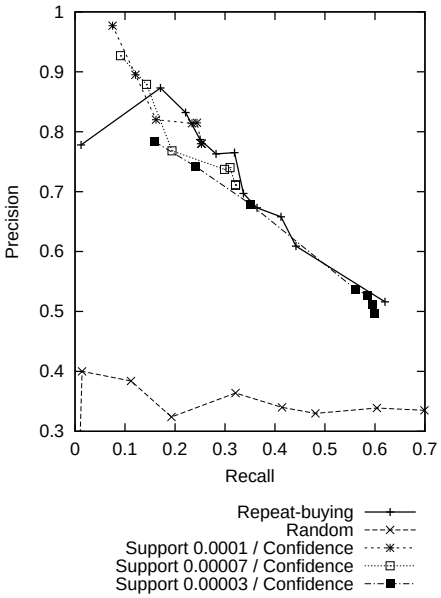

Fig. 10. Precision by recall plot

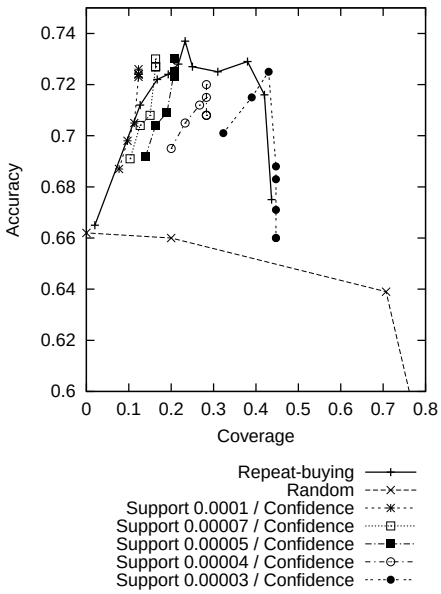

Fig. 11. Accuracy by coverage

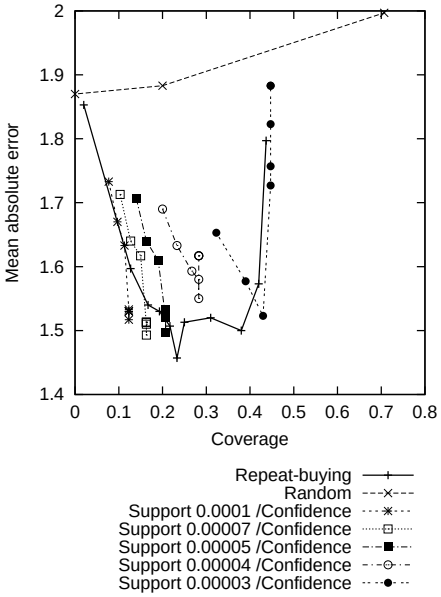

Fig. 12. Mean absolute error by coverage

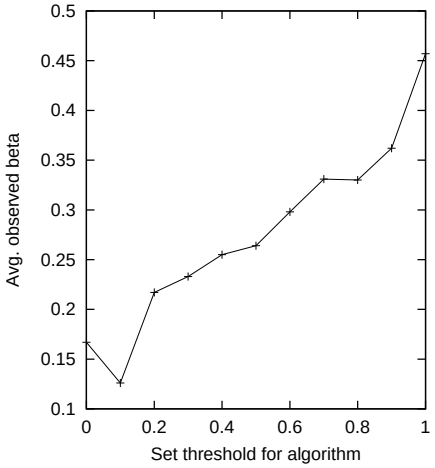

Fig. 13. β by threshold of the repeat-buying algorithm

the analyzed data set. β is a very convenient parameter since it can be interpreted as the proportion of incorrect recommendations in a recommendation list. Figure 13 shows the dependence of the *average observed* β on the threshold of the repeat-buying algorithm. With the exception of the threshold values 0, 0.1, and 0.2 (where only few lists are produced) the *average observed* β is, as expected by the model, below the threshold (the *maximum expected* β). This property gives the algorithm a very stable behavior.

8 Conclusion

Evaluation of the performance of recommender algorithms is an important part of the knowledge discovery process. However, for the data mining part of recommender systems the question of how well found patterns match the user's concept of useful recommendations is often neglected. In this paper we studied this question by comparing recommendations by human peers with the recommendations produced by two recommender algorithms. In the following we summarize the results and remaining research questions:

- The result of the presented evaluation supports the widely accepted assumption that frequent itemsets from purchase histories or from Web usage data represent useful recommendations and therefore support the social process of recommending items. With a well-parameterized algorithm an accuracy of more than 70% and a precision between 60% and 90% have been reached in this study.
- Association rules are free of model assumptions, whereas the repeat-buying algorithm requires several quite strong model assumptions on user behavior which are hard to verify and which invite critic on the validity of the model. However, the assumptions tend to hold approximately for a wide range of applications for consumer products [11]. The good results of the presented recommender algorithm based on repeat-buying is strong evidence that usage patterns in the online world can be described with similar models as consumption behavior in the physical world (e.g. stores).
- The results of both algorithms, when properly tuned, are quite similar. However, the repeat-buying algorithm uses only one parameter which is independent of the data set and has a simple interpretation, whereas the association rule algorithm uses two parameters which strongly depend on each other and on properties of the data set. Furthermore, the repeat-buying algorithm seems to be more robust with regard to a misspecification of its parameter, whereas the association rule algorithm is more flexible and allows fine tuning.
- Both algorithms studied in this paper have a computational complexity of quadratic order. However, for incremental updates the repeat-buying algorithm has a computational complexity of quadratic order in the number of updated items, whereas the simple association rule algorithm is of quadratic complexity on all items. For applications with millions of items the repeat-buying algorithm's possibility to perform efficient incremental updates is essential. Strategies for incremental updates of association rules need to be developed.
- The parameters for the association rule algorithm depend on the size and the structure of the data set, whereas the single parameter of the repeat-buying algorithm,

the sample mean, is robust and sufficient. It is independent of the data set, because in a statistical sense the LSD-model is an approximation to a zero truncated negative binomial distribution. For the automatic operation in a dynamically changing environment this seems to be a considerable advantage.

Since we only used one data set for this paper, additional studies of other data sets from different sources are needed to confirm our findings. We are currently working on the evaluation of large data sets from a B2B-merchant for computer accessories and from the South-West German Library Network.

References

1. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P. In: From Data Mining to Knowledge Discovery: An Overview. MIT Press, Cambridge, MA (1996) 1–36
2. Resnick, P., Varian, H.R.: Recommender systems. Communications of the ACM **40** (1997) 56–58
3. Shardanand, U., Maes, P.: Social information filtering: Algorithms for automating 'word of mouth'. In: Conference proceedings on Human factors in computing systems (CHI'95), Denver, CO, ACM Press/Addison-Wesley Publishing Co. (1995) 210–217
4. Spiliopoulou, M.: Web usage mining for web site evaluation. Communications of the ACM **43** (2000) 127–134
5. Mobasher, B., Cooley, R., Srivastava, J.: Automatic personalization based on web usage mining. Communications of the ACM **43** (2000) 142–151
6. Wirth, R., Hipp, J.: CRISP-DM: Towards a standard process modell for data mining. In: Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, Manchester, UK (2000)
7. Lawrence, R.D., Almasi, G.S., Kotlyar, V., Viveros, M.S., Duri, S.: Personalization of supermarket product recommendations. Data Mining and Knowledge Discovery **5** (2001) 11–32
8. Quadt, A.: Personalisierung im e-commerce. Diplomarbeit, AIFB, Universität Karlsruhe (TH), D-76128 Karlsruhe, Germany (2001)
9. Adamo, J.M.: Data Mining for Association Rules and Sequential Patterns. Springer, New York (2001)
10. Mild, A., Natter, M.: Collaborative filtering or regression models for internet recommendation systems? Journal of Targeting, Measurement and Analysis for Marketing **10** (2002) 304–313
11. Ehrenberg, A.S.C.: Repeat-Buying: Facts, Theory and Application. Charles Griffin & Company Ltd., London (1988)
12. Tan, P.N., Kumar, V.: Discovery of web robot sessions based on their navigational patterns. Data Mining and Knowledge Discovery **6** (2002) 9–35
13. Cooley, R., Mobasher, B., Srivastava, J.: Data preparation for mining world wide web browsing patterns. Journal of Knowledge and Information Systems **1** (1999) 5–32
14. Cooley, R.W.: Web usage mining: Discovery and application of interesting patterns from web data. Ph. d. thesis, Graduate School of the University of Minnesota, University of Minnesota (2000)
15. Berendt, B., Mobasher, B., Spiliopoulou, M., Nakagawa, M.: The impact of site structure and user environment on session reconstruction in web usage analysis. In: Proceedings of the 4th WebKDD 2002 Workshop, at the ACM-SIGKDD Conference on Knowledge Discovery in Databases (KDD'2002), Edmonton, Alberta, Canada (2002)

16. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: Explaining collaborative filtering recommendations. In: Proceedings of the ACM 2000 Conference on Computer Supported Cooperative Work. (2000) 241–250
17. Kohavi, R., Provost, F.: Glossary of terms. Machine Learning **30** (1988) 271–274
18. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
19. van Rijsbergen, C.: Information retrieval. Butterworth, London (1979)
20. Mobasher, B., Dai, H., Tao Luo, M.N.: Discovery and evaluation of aggregate usage profiles for web personalization. Data Mining and Knowledge Discovery **6** (2002) 61–82
21. Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 1999 Conference on Research and Development in Information Retrieval. (1999) 230–237
22. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Application of dimensionality reduction in recommender systems–a case study. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
23. Mobasher, B., Dai, H., Luo, T., Nakagawa, M., Sun, Y., Wiltshire, J.: Discovery of aggregate usage profiles for web personalization. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
24. Vucetic, S., Obradovic, Z.: A regression-based approach for scaling-up personalized recommender systems in e-commerce. In: ACM WebKDD 2000 Web Mining for E-Commerce Workshop. (2000)
25. Yu, K., Xu, X., Ester, M., Kriegel, H.P.: Selecting relevant instances for efficient accurate collaborative filtering. In: Proceedings of the 10th CIKM, ACM Press (2001) 239–246
26. Lin, W., Alvarez, S.A., Ruiz, C.: Efficient adaptive-support association rule mining for recommender systems. Data Mining and Knowledge Discovery **6** (2002) 83–105
27. Agrawal, R., Imielinski, T., Swami, A.: Mining associations between sets of items in large databases. In: Proc. of the ACM SIGMOD Int'l Conference on Management of Data, Washington D.C. (1993) 207–216
28. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: Proc. 20th Int. Conf. Very Large Data Bases, VLDB, Santiago, Chile (1994) 487–499
29. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, Tucson, Arizona, USA (1997) 255–264
30. Aggarwal, C.C., Yu, P.S.: A new framework for itemset generation. In: PODS 98, Symposium on Principles of Database Systems, Seattle, WA, USA (1998) 18–24
31. Bayardo, R., Agrawal, R., Gunopulos, D.: Constraint-based rule mining in large, dense databases. Data Mining and Knowledge Discovery **4** (2000) 217–240
32. Geyer-Schulz, A., Hahsler, M.: A customer purchase incidence model applied to recommender systems. In Kohavi, R., Masand, B.M., Spiliopoulou, M., Srivastava, J., eds.: WEBKDD 2001 Mining Web Log Data Accross All Customer Touch Points. Volume 2356 of LNAI., Berlin, Springer (2002) 25–47
33. Cheung, D.W., Jiawei, H., Ng, V.T., Wong, C.Y.: Maintenance of discovered association rules in large databases: An incremental updating technique. In: Proceedings of the 12th International Conference on Data Engineering, 1996. New Orleans., Piscataway, IEEE (1996) 106–114
34. Ng, K.K., Lam, W.: Updating of association rules dynamically. In: International Symposium on Database Applications in Non-Traditional Environments, 1999 (DANTE'99) Proceedings., Piscataway, IEEE (2000) 84–91

35. Zheng, Z., Kohavi, R., Mason, L.: Real world performance of association rule algorithms. In Provost, F., Srikant, R., eds.: Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining (ACM-SIGKDD), ACM Press (2001) 401–406