AWESOME – A Data Warehouse-based System for Adaptive Website Recommendations

Andreas Thor Erhard Rahm

University of Leipzig, Germany
{thor, rahm}@informatik.uni-leipzig.de

Abstract

Recommendations are crucial for the success of large websites. While there are many ways to determine recommendations, the relative quality of these recommenders depends on many factors and is largely unknown. We propose a new classification of recommenders and comparatively evaluate their relative quality for a sample website. The evaluation is performed with AWESOME (Adaptive website recommendations), a new data warehouse-based recommendation system capturing and evaluating user feedback on presented recommendations. Moreover, we show how AWESOME performs an automatic and adaptive closed-loop website optimization by dynamically selecting the most promising recommenders based on continuously measured recommendation feedback. We propose and evaluate several alternatives for dynamic recommender selection including a powerful machine learning approach.

1 Introduction

Recommendations are crucial for the success of large web sites to effectively guide users to relevant information. E-commerce sites offering thousands of products cannot solely rely on standard navigation and search features but need to apply recommendations to help users quickly find "interesting" products or services. With many users and products manual generation of recommendations is much too laborious and ineffective. Hence a key question becomes how should recommendations be generated automatically to optimally serve the users of a website.

Proceedings of the 30th VLDB Conference, Toronto, Canada, 2004

There are many ways to automatically generate recommendations taking into account different types of information (e.g. product characteristics, user characteristics, or buying history) and applying different statistical or data mining approaches ([JKR02], [KDA02]). Sample approaches include recommendations of top-selling products (overall or per product category), new products, similar products, products bought together by customers, products viewed together in the same web session, or products bought by similar customers. Obviously, the relative utility of these recommendation approaches (*recommenders* for short) depends on the website, its users and other factors so that there cannot be a single best approach. Website developers thus have to decide about which approaches they should support and where and when they should be applied. Surprisingly, little information is available in the open literature on the relative quality of different recommenders. Hence, one focus of our work is an approach for comparative quantitative evaluations of different recommenders.

Advanced websites, such as Amazon [LSY03], support many recommenders but apparently are unable to select the most effective approach per user or product. They overwhelm the user with many different types of recommendations leading to huge web pages and reduced usability. While commercial websites often consider the buying behaviour for generating recommendations, the usage (navigation) behaviour on the website remains largely unexploited. We believe this a major shortcoming since the navigation behaviour contains detailed information on the users' interests not reflected in the purchase data. Moreover, the web usage behaviour contains valuable user feedback not only on products or other content but also on the presented recommendations. The utilization of this feedback to automatically and adaptively improve recommendation quality is a major goal of our work.

AWESOME (Adaptive website recommendations) is a new data warehouse-based website evaluation and recommendation system under development at the University of Leipzig. It contains an extensible library of recommender algorithms that can be comparatively evaluated for real websites based on user feedback. Moreover,

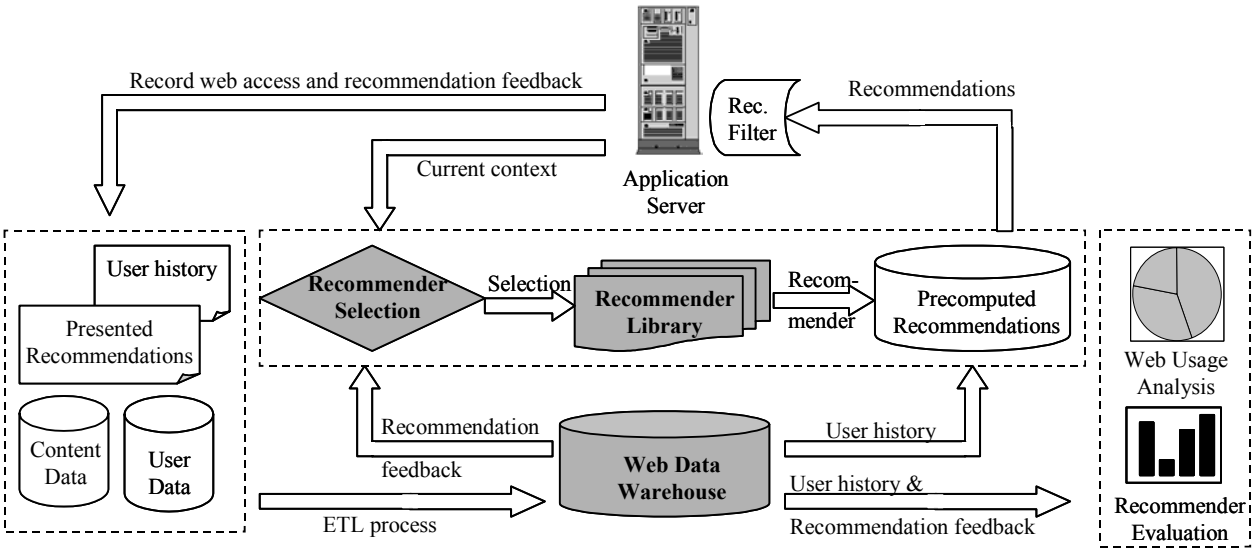

Figure 1: AWESOME architecture

AWESOME can perform an automatic closed-loop website optimization by dynamically selecting the most promising recommenders for a website access. This selection is based on the continuously measured recommendation quality of the different recommenders so that AWESOME automatically adapts to changing user interests and changing content. To support high performance and scalability, quality characteristics of recommenders and recommendations are largely precomputed. AWESOME is fully operational and in continuous use at a sample website; adoption to further sites is in preparation.

The main contributions of this paper are as follows:

- Presentation of the AWESOME architecture for warehouse-based recommender evaluation and for scaleable adaptive website recommendations
- A new classification of recommenders for websites supporting a comparison of different approaches. We show how sample approaches fit the classification and propose a new recommender for users coming from search engines.
- A comparative quantitative evaluation of several recommenders for a sample website. The considered recommenders cover a large part of our classification's design space.
- Description and comparative evaluation of several rule-based approaches for dynamic recommender selection. In particular, a machine learning approach for feedback-based recommender selection is presented.

In the next section we present the AWESOME architecture and the underlying data warehouse approach. We then outline our recommender classification and sample recommenders (Section 3). Section 4 contains the comparative evaluation of several recommenders for a non-commercial website. In Section 5 we describe and evaluate approaches for dynamic recommender selection. Related work is briefly reviewed in Section 6 before we conclude.

2 Architecture

2.1 Overview

Fig. 1 illustrates the overall architecture of AWESOME which is closely integrated with the application server running the website. AWESOME is invoked for every website access, specified by a so-called *context* including information from the current HTTP request such as URL, timestamp and user-related data. For such a context, AWESOME dynamically generates a list of recommendations which are displayed by the application server together with the requested website content. Recommendations are automatically determined by a variety of algorithms from an extensible *recommender library*. The recommenders use information on the usage history of the website and additional information maintained in a *web data warehouse*. The recommendations are subject to a final filter step to avoid the presentation of unsuitable or irrelevant recommendations (e.g., recommendation of the current page or the homepage).

Dynamic selection of recommendations is a two-step process. For a given context, AWESOME first selects the most appropriate recommender(s). This recommender selection is controlled by a moderate number of *selection rules*. For evaluation purposes, we support several selection strategies for determining and adapting these rules, in particular automatic approaches based on user feedback on previously presented recommendations. This recommendation feedback is also recorded in the web data warehouse. For the chosen recommender(s), the best recommendations for the current context are selected in the second step. For performance reasons, these recommenda-

tions are precomputed (and periodically refreshed) and can thus quickly be looked up at runtime.

Separating the selection of recommenders and recommendations makes it easy to add new recommenders. Moreover, using recommendation feedback at the level of recommenders is simpler and more stable than trying to use this feedback for individual recommendations, e.g. specific web pages or products. One problem with the latter approach is that individual pages/products are frequently added and that there is no feedback available for such new content. Conversely, removing content would result in a loss of the associated recommendation feedback.

AWESOME is based on a comprehensive web data warehouse integrating information on the website structure and content (e.g., product catalog), website users and customers, the website usage history and recommendation feedback. The application server continuously records the users' web accesses and which presented recommendations have been and which ones have NOT been followed. During an extensive ETL (extract, transform, load) process (including data cleaning, session and user identification) the usage data and recommendation feedback is added to the warehouse.

The warehouse serves several purposes. Most importantly it is the common data platform for all recommenders and keeps feedback for the dynamic recommender selection thus enabling an automatic closed-loop website optimization. However, it can also be used for extensive offline evaluations, e.g., using OLAP tools, not only for web usage analysis but also for a comparative evaluation of different recommenders and of different strategies for recommender selection. This functionality of AWESOME allows us to systematically evaluate the various approaches under a large range of conditions. It is also an important feature for website designers to fine-tune the recommendation system, e.g. to deactivate or improve less effective recommenders.

The current AWESOME implementation runs on different servers. The warehouse is on a dedicated machine running MS SQL server. The recommendation engine runs on a Unix-based application server where the precomputed recommendations and selection rules are maintained in a MySQL database. In the following, we provide some more details on the ETL process and the warehouse. More information on the recommenders and selection strategies are presented in the subsequent sections.

2.2 ETL process, data warehouse

The ETL workflow to refresh the data warehouse is executed periodically, e.g. once a day. It processes the web log files of the application server and other data sources (e.g., on the website and users). The standard log files of web servers are not sufficient for our purposes because to obtain sufficient recommendation feedback we need to record all presented recommendations and whether or not they have been followed. We thus decided to use tailored application server logging to record this information. Application server logging also enables us to apply effective approaches for session and user identification and early elimination of crawler accesses, thus supporting high data quality.

The AWESOME extensions of the application server are implemented by PHP programs and run together with standard web servers such as Apache. We use two log files: a web usage and a recommendation log file with the formats shown in Table 1. The recommendation log file records all presented recommendations and is required for our recommender evaluation. It allows us to determine positive and negative user feedback, i.e. whether or not a presented recommendation was clicked. For each presented recommendation, we also record the relative position of the recommendation on the page, the generating recommender and the used selection strategy.

The web usage log file adds two elements to the standard Common Log Format (CLF) of common web servers: session ID and user ID. The session ID is generated by the application server and stored inside a temporary cookie on the user's computer (if enabled). These cookies allow a highly reliable session reconstruction [SMB+03]. If the user does not accept temporary cookies, we use heuristic algorithms using client and referrer information for session identification [CMS99]. User IDs are stored inside permanent cookies and are used for user identification. If the user does not accept permanent cookies, user recognition is not done. About 85% of the users of our prototype website accept at least temporary cookies. Hence, cookies support a good balance between data quality and user acceptance, in contrast to client side tracking approaches requiring application of java applets or the like [SBF01].

We use several approaches to detect and eliminate web crawler requests. First, we utilize an IP address list of known crawlers (e.g., from Google) to avoid logging their

Page	requested page
Date, Time	date and time of the request
Client	IP address of the user's computer
Referrer	referring URL
Session ID	session identifier
User ID	user identifier

a) Web usage log

Pageview ID	page view where recommen-dation has been presented
Recommendation	recommended content
Position	position of this recommendation inside a recommendation list
Recommender	recommender that generated the recommendation
Strategy	strategy that selected the applied recommender

b) Recommendation log

Table 1: Log file formats

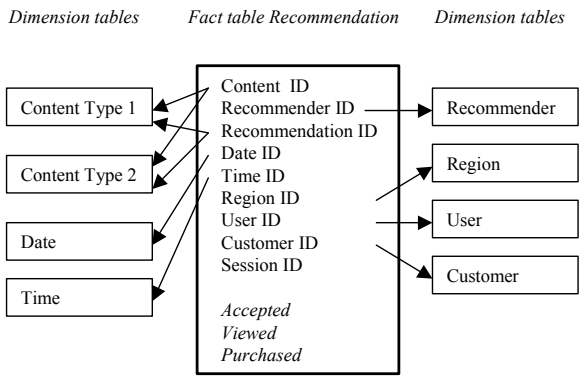

Figure 2: Schema subset for recommendations (simplified)

accesses. In addition, AWESOME eliminates all sessions containing special page views that can only be reached by following links invisible to humans. Finally, we analyze the navigation behavior and attributes like session length, average time between two requests and number of requests with blank referrer to distinguish between human user sessions and web crawler sessions [TK00].

The web data warehouse is a relational database with a "galaxy" schema consisting of several fact tables sharing several dimensions. Like in previous approaches on web usage analysis [KM00] we use separate fact tables for page views, sessions, and – for commercial sites – purchases. In addition we use a recommendation fact table as shown in Fig. 2. The details of the dimension and fact tables depend on the website, e.g. on how the content (e.g., products), users or customers are categorized. In the example of Fig. 2 there are two content dimensions for different hierarchical categorizations. Other dimensions such as user, customer, region and date are also hierarchically organized to allow evaluations at different levels of detail. The recommendation fact table represents the positive and negative user feedback on recommendations. Each record in this table refers to one presented recommendation. The ID attributes are foreign keys on the various dimension tables and identify the recommender algorithm, the recommended content, customer and details of the context (content, user, time, etc.) for which the recommendation was presented. Three Boolean measures are used to derive recommendation quality metrics (see Section 4). *Accepted* indicates whether or not the recommendation was directly accepted (clicked) by the user, while *Viewed* specifies whether the recommended content was viewed later during the respective session (i.e. the recommendation was a useful hint). *Purchased* is only used for e-commerce websites to indicate whether or not the recommended product was purchased during the current session.

The ETL process updates all affected warehouse dimension and fact tables. Moreover, the quality metrics for all recommenders as well as the rules for dynamic recommender selection are updated(see Sections 4 and 5).

3 Recommenders

A recommender generates for a given web page request, specified by a context, an ordered list of recommendations. Such recommendations link to current website content, e.g. pages describing a product or providing other information or services. Recommendations usually are presented as titled links with a short description or preview.

To calculate recommendations, recommenders can make use of the information available in the context as well as additional input, e.g. recorded purchase and web usage data. We distinguish between three types of context information relevant for determining recommendations:

- Current content, i.e. the currently viewed content (page view, product, …) and its related information such as content categories
- Current user, e.g. identified by a cookie, and associated information, e.g. her previous purchases, previous web usage, interest preferences, or current session
- Additional information available from the HTTP request (current date and time, user's referrer, …)

3.1 Recommender classification

Given the many possibilities to determine recommendations, there have been several attempts to classify recommenders ([Bu02], [KDA02], [SKR01], [TH01]). These classifications typically started from a given set of recommenders and tried to come up with a set of criteria covering all considered recommenders. This led to rather complex and specialized classifications with criteria that are only relevant for a subset of recommenders. Moreover, new recommenders can easily require additional criteria to keep the classification complete. For example, [SKR01] introduce a large number of specialized criteria for e-commerce recommenders such as input from target customers, community inputs, degree of personalization, etc.

To avoid these problems we propose a general top-level classification of website recommenders focusing on the usable input data, in particular the context information. This classification may be refined by taking additional aspects into account, but already leads to a distinction of major recommender types thus illustrating the design space. Moreover, the classification helps to compare different recommenders and guides us in the evaluation of different approaches.

Fig. 3 illustrates our recommender classification and indicates where sample approaches fit in. We classify recommenders based on three binary criteria, namely whether or not they use information on the current *content*, the current *user*, and recorded usage (or purchase) *history* of users. This leads to a distinction of eight types of recommenders (Fig. 3). We specify each recommender type by a three-character-code describing whether (+) or not (–) each of the three types of information is used. For instance, type [+,+,–] holds for recommenders that use

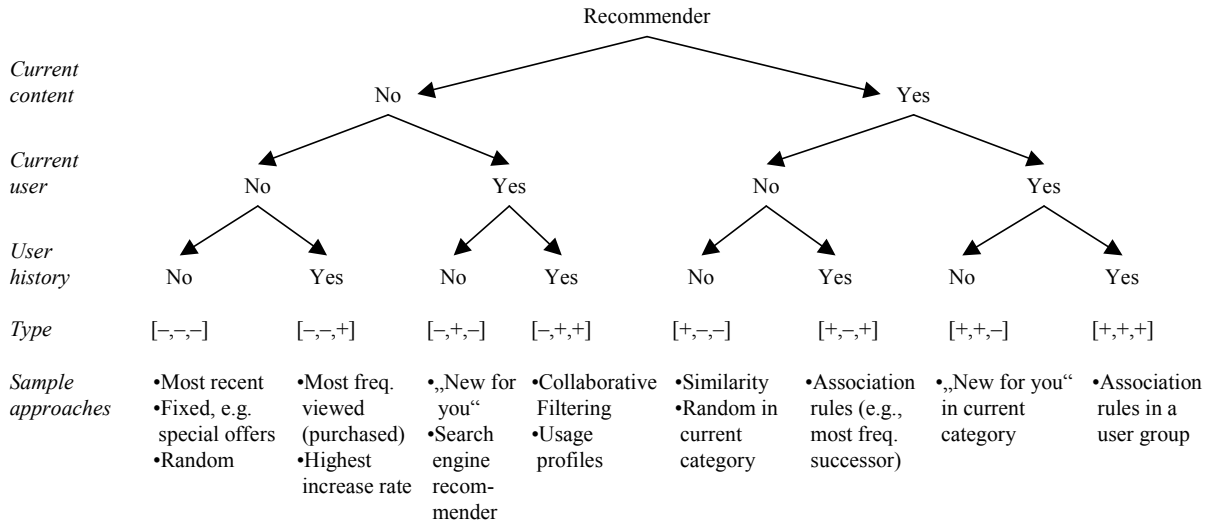

Figure 3: Top-level classification of recommenders

information on the current content and current user, but do not take into account user history.

The first classification criteria considers whether or not a recommender uses the current content, i.e. the currently requested page or product. A sample content-based approach (type [+,–,–]) is to recommend content that is most *similar* to the current content, e.g. based on text-based similarity metrics such as TF/IDF. Content-based recommenders may also use generalized information on the content category (e.g., to recommend products within the current content category). Sample content-insensitive recommenders (type [–,–,–]) are to recommend the *most recent* content, e.g. added within the last week, or to give a fixed recommendation at each page, e.g. for a special offer.

At the second level we consider whether or not a recommender utilizes information on the current user. User-based approaches could thus provide special recommendations for specific user subsets, e.g. returning users or customers, or based on personal interest profiles. Recommenders could also recommend content for individuals, e.g. new additions since a user's last visit (*"New for you"*). We developed a new recommender of type [–,+,–] for users coming from a search engine such as Google. This *search engine recommender (SER)* utilizes that the HTTP referrer information typically contains the search terms (keywords) of the user [KMT00]. SER recommends the website content (different from the current page that was reached from the search engine) that best matches these keywords. The SER implementation in AWESOME utilizes a predetermined search index of the website to quickly provide the recommendations at runtime.

With the third classification criteria we differentiate recommenders by their use of *user history* information. For commercial sites, recommenders can consider information on previous product purchases of customers. Another example is the evaluation of the previous navigation patterns of website users. Simple recommenders of type [–,–,+] recommend the *most frequently* purchased/viewed content (top-seller) or the content with the *highest* recent *increase* of interest.

While not made explicit in the classification, recommenders can utilize additional information than on current content, current user or history, e.g. the current date or time. Furthermore, additional classification criteria could be considered, such as metrics used for ranking recommendations (e.g. similarity metrics, relative or absolute access/purchase frequencies, recency, monetary metrics, etc.) or the type of analysis algorithm (simple statistics, association rules, clustering, etc.).

3.2 Additional approaches

Interesting recommenders often consider more than one of the three main types of user input. We briefly describe some examples to further illustrate the power and flexibility of our classification and to introduce approaches that are considered in our evaluation.

[+,–,+]: *Association rule* based recommenders such as "Users who bought this item also bought …", made famous by Amazon [LSY03], consider the current content (item) and purchase history but are independent of the current user (i.e. every user sees the same recommendations for an item). Association rules can also be applied on web usage history to recommend content which is frequently viewed together within a session.

[–,+,+] Information on navigation/purchase history can be used to determine *usage profiles* [MDL+02] or groups of similar users, e.g. by *collaborative filtering* approaches. Recommenders can assign the current user to a user group (either based on previous sessions or the current session) and recommend content most popular for this group.

In our evaluation we test a *personal interests* recommender, which is applicable to returning users. It deter-

mines the most frequently accessed content categories per user as an indication of her personal interests. When the user returns to the website, the most frequently accessed content of the respective categories is recommended.

[+,+,+] A recommender of this type could use both user groups (as discussed for [–,+,+]) and association rules to recommend the current user those items that were frequently accessed (purchased) by similar users in addition to the current content.

4 Recommender evaluation

The AWESOME prototype presented in Section 2 allows us to systematically evaluate recommenders for a given website. In Section 4.2, we demonstrate this for a sample non-commercial website. Before that, we introduce several metrics for measuring recommendation quality which are needed for our evaluation of recommenders and selection strategies.

4.1 Evaluation metrics

To evaluate the quality of presented recommendations we utilize the Accepted, Viewed, and Purchased measures recorded in the recommendation fact table (Section 2.2). The first two are always applicable, while the last one only applies for commercial websites. We further differentiate between metrics at two levels of granularity, namely with respect to page views and with respect to user sessions.

Acceptance rate is a straight-forward, domain-independent metric for recommendation quality. It indicates the share of page views for which at least one presented recommendation was accepted, i.e. clicked. The definition thus is

$$AcceptanceRate = |P_A| / |P|$$

where P is the set of all page views containing a recommendation and P_A the subset of page views with an accepted recommendation.

Analogously we define a session-oriented quality metric

$$SessionAcceptanceRate = |S_A| / |S|$$

where S is the set of all user sessions and S_A the set of sessions for which at least one of the presented recommendations was accepted.

Recommendations can also be considered of good quality if the user does not directly click them but reaches the associated content later in the session (hence, the recommendation was a correct prediction of user interests). Let P_V be the set of all page views for which any of the presented recommendations was reached later in the user session. We define

$$ViewRate = |P_V| / |P|$$

The corresponding metric at the session level is

$$SessionViewRate = |S_V| / |S|$$

where S_V is the set of all user sessions with at least one pageview in P_V. Obviously, every accepted recommendation is also a viewed recommendation, i.e. $P_A \subseteq P_V \subseteq P$ and $S_A \subseteq S_V \subseteq S$, so that view rates are always larger than or equal to the acceptance rates.

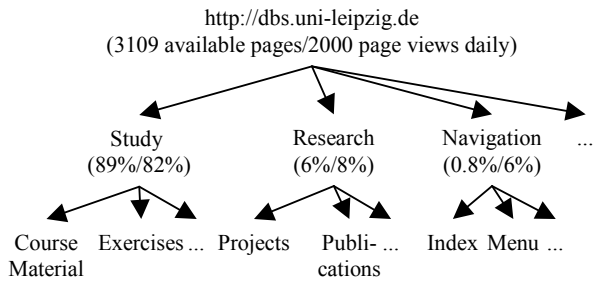

Figure 4: Example of content hierarchy

In commercial sites, product purchases are of primary interest. Note that purchase metrics should be session-oriented because the number of page views needed to finally purchase a product is of minor interest. A useful metric for recommendation quality is the share of sessions S_{AP} containing a purchase that followed an accepted recommendation of the product. Hence, we define the following metric:

$$ReommendedPurchaseRate = |S_{AP}| / |S|$$

Obviously, it holds $S_{AP} \subseteq S_A \subseteq S$.

4.2 Sample evaluation

We implemented and tested the AWESOME approach for recommender evaluation for a sample website, namely the website of our database group (http://dbs.uni-leipzig.de). We use two content hierarchies and Fig. 4 shows a fragment of one of them together with some numbers on the relative size and access frequencies. The website contains more than 3100 pages and receives about 2000 human page views per day (excluding accesses from members of our database group and from crawlers). As indicated in Fig. 4, about 89% of the content is educational study material, which receives about 82% of the page views.

We changed the existing website to show two recommendations on each page so that approx. 4000 recommendations are presented every day. For each page view AWESOME dynamically selects one recommender and presents its two top recommendations (see example in Fig. 5) for the respective context as described in Section 2. We implemented and included more than 100 recommenders in our recommender library. Many of them are variations of other approaches, e.g. considering different user categories or utilizing history data for different periods of time. Due to space constraints we only present results for the six representative recommenders of different types listed in Table 2, which were already introduced in Section 3. The presented results refer to the period from December 1st, 2003 until January 31st, 2004.

Recommender		User type		
Type	Name	New users	Returning users	Σ
[–,–,–]	Most recent	(0.42%)	(0.00%)	*(0.38%)*
[–,–,+]	Most frequent	1.00%	0.62%	*0.92%*
[–,+,–]	SER	2.84%	1.95%	*2.79%*
[–,+,+]	Personal Interests	–	1.54%	*1.54%*
[+,–,–]	Similarity	1.65%	0.82%	*1.56%*
[+,–,+]	Association Rules	1.16%	0.68%	*1.08%*
	Σ	*1.82%*	*1.09%*	*1.69%*

Table 2: Acceptance rate vs. user type

The AWESOME warehouse infrastructure allows us to aggregate and evaluate recommendation quality metrics for a huge variety of constellations (combination of dimension attributes), in particular for different recommenders. For our evaluation we primarily use (page view) acceptance rates as the most precise metric[1]. The average acceptance rate for all considered recommenders was 1.34%; the average view rate was 14.54%. For sessions containing more than one page view the average session acceptance rate was 8.16%, and the session view rate was 25.24%. These rather low acceptance rates are influenced by the fact that every single web page contains a full navigation menu with 78 links (partly nested in sub menus) and that we consciously do not highlight recommendations using big fonts or the like. Note however, that reported "click-tru" metrics are in a comparable range than our acceptance rates [CLP02]. Furthermore, the absolute values are less relevant for our evaluation than the relative differences between recommenders.

Table 2 shows the observed acceptance rates for the six recommenders differentiating between new and returning users. Fig. 6 compares the recommenders w.r.t. the current page type. As expected there are significant differences between recommenders. For our website the *search engine* recommender (SER) achieved the best average acceptance rates (2.79%), followed by the *similarity* and *personal interests* recommenders. On the other hand, simple approaches such as recommending the *most frequently* accessed or *most recent* content achieved only poor average results.

Figure 5: Recommendation screenshot

To more closely analyze the differences for different user and content types, one has to take into account that some recommenders are not always applicable. For instance, the *personal interests* recommender is only applicable for returning users (about 15% for our website). Similarly, SER can only be applied for users coming from a search engine, 95% of which turned out to be new users of the website. In Fig. 6 we only show results for recommenders with a minimum support of 5% i.e. they were applied for at least 5% of all page views of the respective page type. In Table 2 the results for the *most recent* recommender are shown in parentheses because the minimal support could not be achieved due to only few content additions during the considered time period.

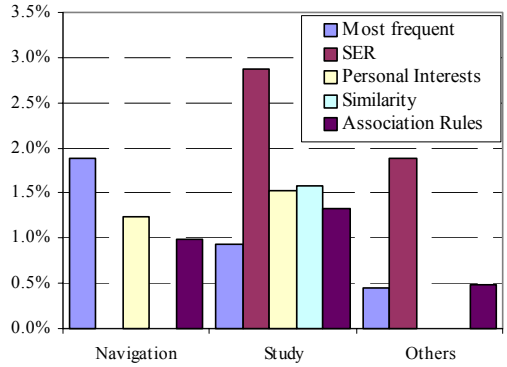

Figure 6 : Acceptance rate vs. page type

Table 2 shows that new users are more likely to accept recommendations than returning users. Except for the personal interests recommender, this holds for all recommenders. An obvious explanation is that returning users (e.g., students for our website) often know where to find relevant information on the website. We also observed that the first page view of a session has a much higher acceptance rate (4.82 %) than later page views in a session (1.28 %). In the latter value, the last page view of a session is not considered, because its acceptance rate obviously equals 0[2].

Fig. 6 illustrates that the relative quality of recommenders differs for different contexts. While the SER recommender achieved the best average results for Study and Others pages, the *most frequent* recommender received the best user feedback on navigation pages. For study pages and non search engine users (when SER is not applicable), either the *personal interests* or *similarity recommender* promise the best recommendation quality.

While these observations are site-specific they illustrate that the best recommender depends on context attributes such as the current content or current user. A careful OLAP analysis may help to determine manually which recommender should be selected in which situation. However, for larger and highly dynamic websites

[1] The recommendations presented during a session typically come from different recommenders making the session-oriented quality metrics unsuitable for evaluating recommenders. Session acceptance rates will be used in Section 5. The Recommended-PurchaseRate does not apply for non-commercial sites.

[2] Layout aspects and other factors also influence acceptance rates. For instance, from the two recommendations shown per page the acceptance rate of the first one was about 50% higher compared to the second recommendation.

{ Usertype='new user' AND ContentCategory1='Navigation' } ➠ *'Most frequent' [0.6]*
{ Referrer='search engine' } ➠ *'SER' [0.8]*
{ Clienttype='university' AND Usertype='returning user' } ➠ *'Personal interest' [0.4]*

Figure 7: Examples of selection rules

this is difficult and labor-intensive so that recommender selection should be automatically optimized.

5 Adaptive recommender selection

AWESOME supports a dynamic selection of recommenders for every website access. This selection is based on selection rules. Rules may either be manually defined or automatically generated. We first present the structure and use of selection rules. We then propose two approaches to automatically generate recommendation rules which utilize recommendation feedback to adapt to changing conditions. Finally we present a short evaluation to compare the different approaches for recommender selection.

5.1 Rule-based recommender selection

Recommender selection entails the dynamic selection of the most promising recommenders for a given context. Therefore selection rules have the following structure:

ContextPattern ➠ *recommender* [*weight*]

Here context pattern is a sequence of values from different context attributes (which are represented as dimension attributes in our warehouse). Typically, only a subset of attributes is specified implying that there is no value restriction for the unspecified attributes. On the right hand side of selection rules, *recommender* uniquely identifies an algorithm of the recommender library and *weight* is a real number specifying the importance of the rule. Fig. 7 shows some examples of such selection rules. In AWESOME, we maintain all rules in a single *SelectionRules* table.

Selection rules allow a straight-forward and efficient implementation of recommender selection. It entails a *match* step to find all rules with a context pattern matching the current context. The rules with the highest weights then indicate the recommenders to be applied. The number of recommenders to choose is typically fixed. In this paper, we focus on the selection of only one recommender, i.e. we choose the rule with the highest weight. The SQL query of Fig. 8 can be used to perform the sketched selection process. Since there may be several matching rules per recommender, the ranking could also be based on the average instead of the maximal weight per recommender.

Example: Consider a new user who reaches the website from a search engine. If her current page belongs to the navigation category, only the first two rules in Fig. 7 match. We select the recommender with the highest weight – SER.

The rule-based recommender selection is highly flexible. Selection rules allow the dynamic consideration of different parts of the current context, and the weights can be used to indicate different degrees of certainty. Rules can easily be added, deleted or modified independently from other rules. Moreover, rules can be specified manually, e.g. by website editors, or be generated automatically. Another option is a hybrid strategy with automatically generated rules that are subsequently modified or extended manually, e.g. to enforce specific considerations.

5.2 Generating selection rules

We present two approaches to automatically generate selection rules, which have been implemented in AWESOME. Both approaches use the positive and negative feedback on previously presented recommendations. The first approach uses the aggregation and query functionality of the data warehouse to determine selection rules. The second approach is more complex and uses a machine learning algorithm to learn the most promising recommender for different context constellations.

5.2.1 Query-based top recommender

This approach takes advantage of the data warehouse query functionality. It generates selection rules as follows:

1. Find all relevant context patterns in the recommendation fact table, i.e. context patterns exceeding a minimal support
2. For every such context pattern *P* do
   a) Find recommender *R* with highest acceptance rate *A*
   b) Add selection rule *P -> R* [*A*]
3. Delete inapplicable rules

The first step ensures that only context constellations with a minimal number of occurrences are considered. This is important to avoid generalization of very rare and special situations (overfitting problem). Note that step 1 checks all possible context patterns, i.e. any of the content attributes may be unspecified, which is efficiently supported by the CUBE operator (SQL extension: GROUP BY CUBE) [GBL+95]. AWESOME is based on a commercial RDBMS providing this operator. For every such context pattern, we run a query to determine the recommender with the highest acceptance rate and produce a corresponding selection rule.

Finally, we perform a rule pruning taking into account that we only want to determine the top recommender per context. We observe that for a rule A with a more general

context pattern and a higher weight than rule B, the latter will never be applied (every context that matches rule B also matches rule A, but A will be selected due to its higher weight). Hence, we eliminate all such inapplicable rules in step 3 to limit the total number of rules.

5.2.2 Machine-learning approach

Recommender selection can be interpreted as a classifier selecting one recommender from a predefined set of recommenders. Hence, machine learning (classification) algorithms can be applied to generate selection rules. Our approach utilizes a well-known classification algorithm constructing a decision tree based on training instances (Weka J48 algorithm [WF00]). To apply this approach, we thus have to transform recommendation feedback into training instances. An important requirement is that the generation of training data must be completely automatic so that the periodic re-calculation of selection rules to incorporate new recommendation feedback is not delayed by the need of human intervention.

The stored recommendation feedback indicates for each presented recommendation, its associated context attributes, and the used recommender whether or not the recommendation was accepted. A naïve approach to generate training instances would simply select a random sample from the recommendation fact table (Fig. 2), e.g. in the format (*context*, *recommender*, *accepted*). However, classifiers using such training instances would rarely predict a successful recommendation since the vast majority of the instances may represent negative feedback (> 98% for the sample website). Ignoring negative feedback is also no solution since the number of accepted recommendations is heavily influenced by the different applicability of recommenders and not only by their recommendation quality. Therefore, we propose a more sophisticated approach that determines the number of training instances according to the acceptance rates:

1. Find all relevant feedback combinations *(context, recommender)*
2. For every combination *c* do
   a) Determine acceptance rate for c. Scale and round it to compute integer weight n_c
   b) Add instance *(context, recommender)* n_c times to training data
3. Apply decision tree algorithm
4. Rewrite decision tree into selection rules

In Step 1, we do not evaluate context patterns (as in the previous approach), which may leave some context attributes unspecified. We only consider fully specified context attributes and select those combinations exceeding a minimal number of recommendation presentations. For each such relevant combination c *(context, recommender),* we use its acceptance rate to determine the number of training instances n_c. To determine n_c, we linearly scale the respective acceptance rate from the 0 to 1 range by multiplying it with a constant k and rounding to an integer value. For example, assume 50 page views for the combination of context ("returning user", "search engine", "Navigation", …) and recommender "Most frequent". If there are 7 accepted recommendations for this combination (i.e. acceptance rate 0,14) and k=100, we add n_c =14 identical instances of the combination to the training data. This procedure ensures that recommenders with a high acceptance rate produce more training instances than less effective recommenders and therefore have a higher chance to be predicted.

```
SELECT Recommender, MAX (Weight)
FROM SelectionRules
WHERE ((RuleContextAttribute1 = CurrentContextAttribute1)
        OR (RuleContextAttribute1 IS NULL))
AND    ((RuleContextAttribute2 = CurrentContextAttribute2)
        OR (RuleContextAttribute2 IS NULL))
AND …
GROUP BY Recommender
ORDER BY MAX(Weight) DESC
```

Figure 8: SQL query for selection strategy execution

The resulting set of training instances is the input for the classification algorithm producing a decision tree. With the help of cross-validation, all trainings instances are simultaneously used as test instances. The final decision tree can easily be rewritten into selection rules. Every path from the root to a leaf defines a context pattern where all unspecified context attributes are set to NULL. Each leaf specifies a recommender and the rule weight is set to the relative fraction of correctly classified instances provided by the classification algorithm.

5.3 Evaluation of selection rules

To evaluate the effectiveness of the two presented selection strategies we tested them with AWESOME on the sample website introduced in Section 4.2. For comparison purposes we also evaluated two sets of manually specified rules and a random recommender selection giving a total of five approaches (see Table 3). For every user session AWESOME uniformly selected one of the strategies; the chosen strategy is additionally recorded in the recommendation log file for evaluation purposes. We applied the selection strategies from January 1st until February 25th, 2004.

Table 4 shows the average number of rules per selection strategy as well as the average delay to dynamically select a recommender. Even for the two automatic approaches the number of rules is moderate (250 – 2000) and permitted very fast recommender selection of less than 20 ms on average for a Unix server (execution time for the query in Fig. 8).

Name	Description
Top-Rec	Automatic strategy of section 5.2.1 (query-based)
Decision Tree	Automatic strategy of section 5.2.2 (machine learning)
Manual 1	The most frequently viewed pages per content category are recommended. For returning users, this category is derived from previous sessions of the current user. For new users, the category of the current page is used.
Manual 2	For search engine users, the search engine recommender is applied. Otherwise the content similarity recommender (for course material pages) or association rule recommender (for other pages) is selected.
Random	Random selection of a recommender

Table 3: Tested selection strategies

Table 4 also shows the average (page view) acceptance rates and session acceptance rates for the five selection strategies. The two automatic feedback-based strategies for recommender selection (Top-Rec, Decision Tree) showed significantly better average quality than random and the first manual policy. The machine learning approach (Decision Tree) and the Manual2 policy were the best strategies. Note that the very effective strategy Manual2 utilizes background knowledge about the website structure and typical user groups (students, researchers) as well as evaluation results obtained after an extensive manual OLAP analysis (partially presented in Section 4.2), such as the effectiveness of the search engine recommender. The fact that the completely automatic machine learning algorithm achieves comparable effectiveness is thus a very positive result. It indicates the feasibility of the automatic closed-loop optimization for generating recommendations and the high value of using feedback to significantly improve recommendation quality without manual effort.

The comparison of the two automatic strategies shows that the machine learning approach performs much better than the query-based top recommender approach. The decision tree approach uses significantly fewer rules and was able to order the context attributes according to their relevance. The most significant attributes appear in the upper part of the decision tree and therefore have a big influence on the selection process. On the other hand, Top-Rec handles all context attributes equally and uses many more rules. So recommender selection was frequently based on less relevant attributes resulting in poorer acceptance rates.

The warehouse infrastructure of AWESOME allows us to analyze the recommendation quality of selection strategies for many conditions, similar to the evaluation of individual recommenders (Section 4.2). Figure 9 shows the session acceptance rates of the best two selection strategies w.r.t. user type, referrer, and entry page type, i.e. the page type of the first session page view. We observe that the manual strategy is more effective for search engine users by always applying the SER recommender to them. This helped to also get slightly better results for new users and sessions starting with an access to study material. On the other hand, the machine learning approach was significantly more effective for users not coming from search engines and returning users. These results indicate that the automatically generated selection rules help generate good recommendations in many cases without the need of extensive manual evaluations, e.g. using OLAP tools. Still, overall quality can likely be improved by adding a few manual rules (with high weight) to incorporate background knowledge. We will investigate such hybrid strategies in future work.

Strategy	Nr. of rules	Selection time	Acceptance rate	Session acceptance rate
Top-Rec	~ 2000	~ 14 ms	1.25 %	9.58 %
Decision Tree	~ 250	~ 12 ms	1.64 %	12.54 %
Manual 1	24	~ 13 ms	0.96 %	7.11 %
Manual 2	5	~ 13 ms	1.84 %	12.47 %
Random	137	~ 19 ms	0.89 %	6.51 %

Table 4: Comparison of selection strategies

6 Related work

An overview of previous recommendation systems and the applied techniques can be found in [JKR02], [KDA02] and [SCD+00]. [LSY03] describes the Amazon recommendation algorithms, which are primarily content (item)-based and also heavily use precomputation to achieve scalability to many users. [Bu02] surveys and classifies so-called hybrid recommendation systems which combine several recommenders. To improve hybrid recommendation systems, [SKR02] proposes to manually assign weights to recommenders to influence recommendations. [MN03] presents a hybrid recommendation system

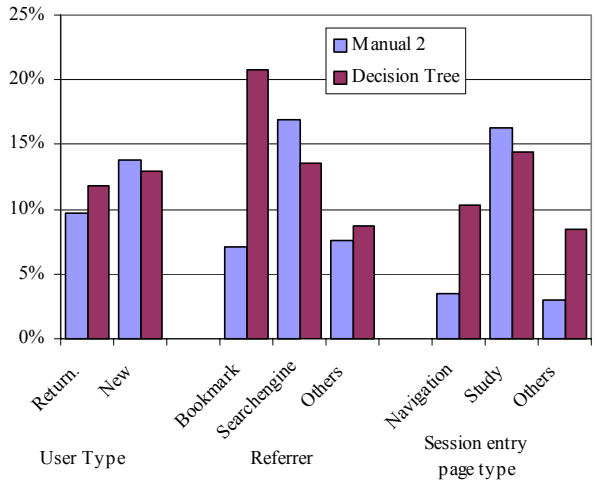

Figure 9: SessionAcceptanceRate w.r.t. user type, referrer, and session entry page type.

switching between different recommenders based on the current page's position within a website. The Yoda system [SC03] uses information on the current session of a user to dynamically select recommendations from several predefined recommendation lists. In contrast to AWESOME, these previous hybrid recommendation systems do not evaluate or use recommendation feedback.

[LK01] sketches a simple hybrid recommendation system using recommendation feedback to a limited extent. They measure which recommendations produced by three different recommenders are clicked to determine a weight per recommender (with a metric corresponding to our view rate). These weights are used to combine and rank recommendations from the individual recommenders. In contrast to AWESOME negative recommendation feedback and the current context are not considered for recommender evaluation. Moreover, there is no automatic closed-loop adaptation but the recommender weights are determined by an offline evaluation.

The evaluation of recommendation systems and quantitative comparison of recommenders has received little attention so far. [KCR02] monitored users that were told to solve certain tasks on a website, e.g. to find specific information. By splitting users in two groups (with recommendations vs. without) the influence of the recommendation system is measured. Other studies [GH02], [HC02] asked users to explicitly rate the quality of recommendations. This approach obviously is labor-intensive and cannot be applied to compare many different recommenders.

[GH02] and [SKK+00] discuss several metrics for recommendation quality, in particular the use of the information retrieval metrics precision and recall. The studies determine recommendations based on an offline evaluation of web log or purchase data; the precision metric, for instance, indicates how many of the recommendations were reached within the same session (thus corresponding to our view rate). In contrast to our evaluation, these studies are not based on really presented recommendations and measured recommendation feedback so that the predicted recommendation quality remains unverified.

In [HMA+02] a methodology is presented for evaluating two competing recommenders. It underlines the importance of such an online evaluation and discusses different evaluation aspects. Cosley et. al. developed the REFEREE framework to compare different recommenders for the CiteSeer website [CLP02]. Click metrics (e.g., how often a user followed a link or downloaded a paper), which are similar to the acceptance rates used in our study, are used to measure recommendation quality.

As an alternative to the two-level approach for selecting recommendations, we recently started to investigate how to directly determine suitable recommendations without prior selection of recommenders [GR04]. The approach requires that different recommenders produce comparable weights for individual recommendations. Reinforcement learning approaches can be used to consider user feedback for individual recommendations. A comparative evaluation of this approach with the presented two-level scheme is subject to future work.

7 Summary

We presented AWESOME, a new data warehouse-based website evaluation and recommendation system. It allows the coordinated use of a large number of recommenders to automatically generate website recommendations. Recommendations are dynamically determined by a flexible rule-based approach selecting the most promising recommender for the respective context. AWESOME supports a completely automatic generation and optimization of selection rules to minimize website administration overhead and quickly adapt to changing situations. This optimization is based on a continuous measurement of user feedback on presented recommendations. To our knowledge, AWESOME is the first system enabling such a completely automatic closed-loop website optimization. The use of data warehouse technology and precomputation of recommendations support scalability and fast web access times.

We presented a simple but general recommender classification. It distinguishes eight types of recommenders based on whether or not they consider input information on the current content, current user and users history. To evaluate the quality of recommendations and recommenders, we proposed the use of several acceptance rate metrics based on measured recommendation feedback. We used these metrics for a detailed comparative evaluation of different recommenders and different recommender selection strategies for a sample website. Our results so far indicate that the use of machine learning is most promising for an automatic feedback-based recommendation selection. The presented policy is able to automatically determine suitable training data so that its periodic re-execution to consider new feedback does not require human intervention.

In future work, we will adopt AWESOME to additional websites, in particular e-shops, to further verify and fine-tune the presented approach. We will investigate hybrid selection strategies where automatically generated selection rules are complemented by a limited number of manually specified rules utilizing site-specific optimization criteria or specific background knowledge. Finally, we will explore specific recommendation opportunities such as selecting the best recommender for product bundling (cross-selling).

Acknowledgements

We thank Nick Golovin and Robert Lokaiczyk for fruitful discussions and help with the implementation. The first author was funded by the German Research Foundation within the Graduiertenkolleg "Knowledge Representation".

References

[Bu02] Burke, R.: *Hybrid Recommender Systems: Survey and Experiments*. User Modeling and User-Adapted Interaction 12(4), 2002

[CMS99] Cooley, R., Mobasher, B., Srivastava, J.: *Data preparation for mining world wide web browsing patterns.* Knowledge and Information Systems. 1(1), 1999

[CLP02] Cosley, D., Lawrence, S., Pennock, D. M.: *REFEREE: An open framework for practical testing of recommender systems using ResearchIndex*. Proc. 28th VLDB conf., 2002

[GBL+95] J. Gray, A. Bosworth, A. Layman, H. Pirahesh. *Data cube: A relational aggregation operator generalizing groupby, cross-tab, and sub-total*. Proc. of the 12th EEE International Conference on Data Engineering (ICDE), 1995

[GH02] Geyer-Schulz, A., Hahsler, M.: *Evaluation of Recommender Algorithms for an Internet Information Broker based on Simple Association rules and on the Repeat-Buying Theory*. Proc. of ACM WebKDD Workshop, 2002

[GR04] Golovin, N., Rahm, E.: *Reinforcement Learning Architecture for Web Recommendations*. Int. Conf. on Information Technology (ITCC), 2004

[HC02] Heer, J., Chi, E. H.: *Separating the Swarm: Categorization Methods for User Sessions on the Web*. Prof. Conf. on Human Factors in Computing Systems, 2002

[HMA+02] Hayes, C., Massa, P., Avesani, P., Cunningham, P.: *An on-line evaluation framework for recommender systems*. Proc. of Workshop on Personalization and Recommendation in E-Commerce, 2002

[JKR02] Jameson, A., Konstan, J., Riedl, J.: *AI Techniques for Personalized Recommendation*. Tutorial at 18th National Conf. on Artificial Intelligence (AAAI), 2002

[KCR02] Kim, K., Carroll, J. M., Rosson, M. B.: *An Empirical Study of Web Personalization Assistants: Supporting End-Users in Web Information Systems*. Proc. IEEE 2002 Symp. on Human Centric Computing Languages and Environments, 2002

[KDA02] Koutri, M., Daskalaki, S., Avouris, N.: *Adaptive Interaction with Web Sites: an Overview of Methods and Techniques.* Proc. 4th Int. Workshop on Computer Science and Information Technologies (CSIT), 2002

[KM00] Kimball, R., Merz, R.: *The Data Webhouse Toolkit – Building Web-Enabled Data Warehouse*. Wiley Computer Publishing, New York, 2000

[KMT00] Kushmerick, N., McKee, J., Toolan, F.: *Toward zero-input personalization: Referrer-based page recommendation*. Proc. Int. Conf. on Adaptive Hypermedia and Adaptive Web-based Systems, 2000

[LK01] Lim, M., Kim, J.: *An Adaptive Recommendation System with a Coordinator Agent*. In Proc. 1st Asia-Pacific Conference on Web Intelligence: Research and Development, 2001

[LSY03] Linden, G., Smith, B., York, J.: *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*. IEEE Distributed Systems Online 4(1), 2003

[MDL+02] Mobasher, B., Dai, H., Luo, T., Nakagawa, M.: *Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization*. Data Mining and Knowledge Discovery, Kluwer, 6 (1), 2002

[MN03] Mobasher, B., Nakagawa, M.: *A Hybrid Web Personalization Model Based on Site Connectivity*. Proc. ACM WebKDD Workshop, 2003

[SBF01] Shahabi, C., Banaei-Kashani, F., Faruque, J.: *A Reliable, Efficient, and Scalable System for Web Usage Data Acquisition.* Proc. ACM WebKDD Workshop, 2001

[SC03] Shahabi, C., Chen, Y.: *An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback*. Distributed and Parallel Databases 14(2), 2003

[SCD+00] Srivastava, J., Cooley, R., Deshpande, M., Tan, P-T.: *Web Usage Mining: Discovery and Applications of Usage Patterns from Web Data*. SIGKDD Explorations, (1) 2, 2000

[SKK+00] Sarwar, B., Karypis, G., Konstan, J. , Riedl, J.: *Analysis of recommendation algorithms for e-commerce*. Proc. of ACM E-Commerce, 2000

[SKR01] Schafer, J.B., Konstan, J. A., Riedl, J.: *Electronic Commerce recommender applications.* Journal of Data Mining and Knowledge Discovery, 5 (1/2), 2001

[SKR02] Schafer, J. B., Konstan, J. A., Riedl, J.: *Meta-recommendation systems: user-controlled integration of diverse recommendations*. Proc. 11th Int. Conf. on Information and Knowledge Management (CIKM), 2002

[SMB+03] Spiliopoulou, M., Mobasher, B., Berendt, B., Nakagawa, M.: *A Framework for the Evaluation of Session Reconstruction Heuristics in Web Usage Analysis*. INFORMS Journal of Computing, Special Issue on Mining Web-Based Data for E-Business Applications, 15 (2), 2003

[TH01] Terveen, L., Hill, W.: *Human-Computer Collaboration in Recommender Systems*. In: Carroll, J. (ed.): Human Computer Interaction in the New Millenium. New York: Addison-Wesley, 2001

[TK00] Tan, P., Kumar, V.: *Modeling of Web Robot Navigational Patterns*. Proc. ACM WebKDD Workshop, 2000

[WF00] Witten, I.H., Frank, E.: *Data Mining. Practical Machine Learning Tools and techniques with Java implementations*. Morgan Kaufmann. 2000