A Collaborative Constraint-based Meta-level Recommender*

Markus Zanker
University Klagenfurt
Universitätsstrasse 65-67
9020 Klagenfurt, Austria
markus.zanker@uni-klu.ac.at

ABSTRACT

Recommender Systems (RS) have become popular for their ability to make useful suggestions to online shoppers. Knowledge-based RS represent one branch of these types of applications that employ means-end knowledge to map abstract user requirements to product characteristics. Before setting up such a system, the knowledge has to be acquired from domain experts and formalized using constraints or a comparable representation mechanism. However, the initial acquisition of the knowledge base and its maintenance are effort intensive tasks. Here, we propose a system that learns rule-based preferences from successful interactions in historic transaction data. It is realized as a meta-level hybrid that employs collaborative filtering to derive preferences from a user's nearest neighbors that are processed by a knowledge-based RS to derive recommendations. An evaluation using a commercial dataset showed that this approach outperforms the prediction accuracy of a knowledge base provided by domain experts. In addition, the approach is applicable for supporting domain experts in the maintenance and validation tasks associated with providing personalization knowledge bases.

Categories and Subject Descriptors

H.3.3 [Information Systems]: Information Search and Retrieval—*Relevance feedback, Retrieval models*; I.2.1 [Artificial Intelligence]: Applications and Expert Systems

General Terms

Algorithms

Keywords

Hybrid recommendation approaches, collaborative filtering, knowledge-based recommendation

*This work was partly supported by FFG grant MAPREC nr. 814294. Furthermore, it was conducted within the scope of the V-Know research project (FWF grant nr. 19996).

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

RecSys'08, October 23–25, 2008, Lausanne, Switzerland.
Copyright 2008 ACM 978-1-60558-093-7/08/10 ...\$5.00.

1. INTRODUCTION

Recommender systems are a fundamental technology for providing personalized user experiences on e-shopping platforms. They either explicitly request users to comment on items (e.g. by providing ratings or critique), ask them to state their requirements or exploit implicitly acquired preference information such as navigation actions or previous purchases. Knowledge-based [10, 7, 14] and case-based recommender systems [20, 16] typically engage in a conversational interaction with their users to collect explicit user preference and requirements information. This is in contrast with the prominent variants of recommender systems in the literature, namely collaborative and content-based filtering systems [5, 21], that typically pursue solely one-shot interaction strategies. Collaborative recommender systems compute item propositions by analyzing community data and similarities between users or items, while content-based filtering retrieves items that are similar to those the user liked in the past. Hybrid approaches combine at least two of the aforementioned techniques by, for instance, weighting and combining recommendations from two RS into a single list or employing one technique to augment the input data of a second one. Such systems have been developed to improve recommendation quality and to alleviate some of the specific problems of a single recommendation paradigm approach [8, 3]. For instance, a system based on the collaborative filtering paradigm suffers from the cold-start problem for new users or new items. Users that have not yet submitted ratings to the system cannot receive meaningful recommendations. Similarly, new items that have not been rated by several users will not be recommended. Knowledge-based RS can provide reasonable recommendations right from the start of the system at the expense of a considerable knowledge engineering effort during its inception. In addition, the system's knowledge needs to be continuously maintained in order to avoid wrong or outdated recommendations.

Therefore, we propose a collaborative approach that combines rule-based preferences¹ from users' nearest neighbors to provide a personalized knowledge base that can outperform a general domain knowledge base, provided by experts. According to the typology of hybrid recommender systems in [8], the proposed system is classified as a meta-level hybrid where a collaborative recommender contributes a model (i.e. a knowledge base) that is used by the knowledge-based recommender to generate recommendations. Although the pro-

¹Note, that we use the term *preferences* alternatively to the notion of *soft* constraints. By *rule-based* we mean implications in logic of the form $A \rightarrow B$.

Table 1: Product catalog

id	property	value
1	name	<i>Montecristo Nr. 3</i>
	brand	<i>Montecristo</i>
	format	<i>Corona</i>
	price	<i>47</i>
2	name	<i>Esplendidos</i>
	brand	<i>Cohiba</i>
	format	<i>Churchill</i>
	price	<i>22</i>

Table 2: Community transactions

User	for_whom	experience	cost	product
Sue	<i>gift</i>	<i>none</i>	<i>40</i>	1
Tom	<i>gift</i>	<i>low</i>	<i>50</i>	1
Bob	<i>myself</i>	<i>high</i>	<i>20</i>	2
Jim	<i>myself</i>	<i>high</i>	<i>50</i>	2
New user	<i>gift</i>	<i>none</i>	<i>20</i>	?

posed hybrid combines favorable properties from the worlds of collaborative and knowledge-based systems, the research area remains relatively unexplored [9].

In the following section we give a motivating example based on an application scenario from the domain of premium cigars. Section 3 presents our collaborative constraint-based recommendation approach that learns functional dependencies from successful past user transactions. In Section 4 we evaluate the collaborative constraint-based recommendation system by comparing it with a traditional knowledge-based RS and discuss results. Finally, we summarize related work and conclude with an outlook on open research tasks.

2. MOTIVATING EXAMPLE

Consider a RS in a trivial example domain, where users are required to choose between two different types of cigars. Table 1 encodes the product catalog. Besides its name, a cigar is characterized by its brand name, its format and the price. In order to be capable of making knowledgeable recommendations, a conversational recommender system must first interact with its users to elicit their explicit requirements. Typically, a series of questions are asked. For instance: *"who is it for?"* (short: for_whom), *"how experienced are you in smoking cigars?"* (experience) or *"how much are you willing to spend?"* (cost). Table 2 lists the answers users gave to these questions and denotes the id of the product they bought, effectively forming a matrix where each row represents a user session and columns hold the explicit requirements that users disclosed to the system and the ids of actually bought items. For instance, user *Sue* was looking for a gift for someone with no smoking experience, had a price expectation of 40 currency units and bought item number 1. Of particular interest is the last row of Table 2, which represents the *new user* entering the platform. The new user requires a cigar as a gift for someone with no smoking experience and her price expectation lies at 20 currency units per cigar. The goal of the RS is to predict which item she is most likely to buy.

While a traditional knowledge-based recommender exploits a knowledge base that was engineered by domain experts, the approach presented here extracts rule-based preference

Table 3: Derived preference knowledge

User	preference rule
Sue	If for_whom = <i>'gift'</i> then brand = <i>'Montecristo'</i> . If experience = <i>'none'</i> then format = <i>'Corona'</i> . If cost ≤ 40 then price $\leq 50^*$.
Tom	If for_whom = <i>'gift'</i> then brand = <i>'Montecristo'</i> . If experience = <i>'low'</i> then format = <i>'Corona'</i> . If cost ≤ 50 then price ≤ 50 .
Bob	If for_whom = <i>'myself'</i> then brand = <i>'Cohiba'</i> . If experience = <i>'high'</i> then format = <i>'Churchill'</i> . If cost ≤ 20 then price ≤ 30 .
Jim	If for_whom = <i>'myself'</i> then brand = <i>'Cohiba'</i> . If experience = <i>'high'</i> then format = <i>'Churchill'</i> . If cost ≤ 50 then price ≤ 30 .

*Note, that we rounded the price of a product and bucketed continuous values into equivalence classes.

knowledge from the community transactions. The goal is to identify functional dependencies between user requirements and product properties that can be represented as rule-based preference information in the form of **If** *<condition>* **then** *<consequent>* restrictions. Given some rule templates or patterns that describe presumably meaningful mappings from user requirements to product characteristics, the transformation step of generating preference knowledge can be done automatically. For instance, the smoking experience of a user might influence her choice of format, as less experienced smokers may feel sick when having smoked a large cigar. On the other hand, further dependencies such as between user's price expectations and the product name should not be automatically generated because they have not been deemed to be meaningful by a domain expert. Table 3 exemplifies the preference knowledge that has been automatically derived from the community transactions. For instance user *Sue* was looking for a gift and bought the prestigious *Montecristo* brand, which results in the preference rule: **If** for_whom = *'gift'* **then** brand = *'Montecristo'*.

The idea of the collaborative constraint-based recommendation approach is not to use all preference rules from Table 3 for making a prediction **but only those that derive from the most similar peers of a user**. Therefore, in order to compute a recommendation for the new user, a collaborative RS will first determine those users within the community that are most similar according to their user models. The k-nearest neighbor method using Pearson correlation or cosine as similarity measures is the standard way of neighborhood determination in user-based collaborative filtering [3]. Here, a simple matching coefficient based on

Figure 1: Computation of recommendations with a personalized KB

uniformly weighted user requirements is used for similarity computation. It indicates that *Sue* has a similarity degree of $\frac{2}{3}$ and *Tom* and *Bob* are similar by $\frac{1}{3}$ while *Jim* is completely dissimilar to the new user.

As a result the collaborative RS proposes a personalized knowledge base that is a subset of the automatically derived preference knowledge (see Figure 1), containing only the preference rules from the new user's most similar peers (i.e. *Sue*, *Tom* and *Bob*). Note, that preferences are weighted by the sum of computed similarities that can be interpreted as priority values when conflicts between constraints need to be resolved, e.g. the preference rule that recommends the *Montecristo* brand if users require a gift derives from users *Sue* and *Tom* and thus the sum of their similarity metrics results in a priority of 1. Note, that in this example all other preference rules only have the support of a single peer and therefore their priority is simply the similarity metric of the peer from whose transaction record the rule has been derived.

The knowledge-based RS has to evaluate the personalized knowledge base in order to derive a recommendation. First, the constraint solver determines which preference rules are applicable based on their condition part and the input from the current user model. For instance, the implication **If for_whom='gift' then brand='Montecristo'** is fulfilled for the first product record because for_whom='gift' is part of the current user model. Note that all applicable constraints are shaded grey in Figure 1. If none of the products satisfies

Figure 2: Recommendation process

all preference rules then some of these constraints must be relaxed in order to derive a recommendation. In this example the first product record satisfies all preference rules except the last one (weight $\frac{1}{3}$) while the second product record requires the relaxation of the first two constraints (the sum of weights of both is $\frac{5}{3}$). As a consequence, a RS using a *min-weight* relaxation strategy will propose the first product record, i.e. the *Montecristo Nr. 3* cigar. The following section will present a more formal description of the proposed approach.

3. COLLABORATIVE CONSTRAINT-BASED RS

The proposed collaborative constraint-based recommender system is a meta-level hybrid that combines collaborative filtering with the knowledge-based recommendation paradigm. As outlined in Figure 2, an offline preprocessing component generates preference rules (denoted KB_{gen}) by processing previous transaction. This task may be supported by rule templates that restrict the size of the problem space. For instance, only reasonable combinations of user requirements and product properties should be generated as constraint candidates or, in case of numeric properties, range restrictions are used instead of equality conditions.

The collaborative filtering step takes the original community data as well as the user model of the current user and the generated constraint candidates as input and selects a set of constraints that serves as a knowledge base (KB_{coll}) for the subsequent recommendation step.

The knowledge-based RS exploits the dynamically created KB_{coll} and the user model of the current user to retrieve a ranked list of recommendable items from the product catalog.

In [3] the recommendation problem is formalized by assuming a utility function rec that measures the usefulness of an item $i \in I$ to a user $u \in U$, i.e. $rec(i, u) \mapsto R$, where R is a totally ordered set of numbers within a certain range. Thus, the task of a recommender system is to identify for each user those items i_j that follow an abstract goal such as

maximizing the user's utility, increasing online conversion rates or even optimizing the user's lifetime value.

In order to formalize our proposed hybrid approach using the definition of a recommendation problem from [3], the following subsections first define the collaborative and knowledge-based recommendation paradigms.

3.1 User-based collaborative filtering

User-based collaborative filtering exploits similarities between users to make predictions. The approach taken here is memory-based user-to-user correlation, as the user neighborhood for a new user has to be computed while the system is operating. Collaborative filtering has the advantage of neither requiring the acquisition of domain knowledge prior to system deployment nor structured product descriptions. Therefore, product records can be solely represented by a unique identifier. However, collaborative filtering assumes the existence of recorded community transactions. The first step is neighborhood formation where users similar to the current user are identified. In our situation user models consist of attribute-value pairs that can be interpreted as nominal ratings. User similarities are computed from the set of u 's explicit requirements Rq_u . For instance, according to our introductory example $Rqs_{ue} = \{gift, none, 40\}$.

In our implementation we use the Dice coefficient [12] to determine the similarity between two users u and v , i.e.:

$$sim(u, v) = \frac{2 \times |Rq_u \cap Rq_v|}{|Rq_u| + |Rq_v|} \quad (1)$$

In order to derive recommendations the k-Nearest Neighbor method is used. Besides merely storing a user's requirements, the profile of each user v also contains a set of binary ratings on items R_v which were actually bought by v , e.g. $Rs_{ue} = \{1\}$. Thus $rec_{cf}(i, u)$ computes the most frequent successful transactions in u 's neighborhood of users N_u (i.e. the set of peers from the community), where the neighborhood size is limited by k , i.e. $|N_u| \leq k$.

$$rec_{cf}(i, u) = \frac{\sum_{v \in N_u} score_{i,v}}{|N_u|}, \text{ where} \quad (2)$$

$$score_{i,v} = \begin{cases} sim(u, v) & : i \in R_v \\ 0 & : \text{else} \end{cases} \quad (3)$$

For a more comprehensive presentation of the details of collaborative filtering the reader is referred to Schafer et al. [23].

3.2 Knowledge-based recommendation

This type of RS requires a knowledge base containing domain heuristics which are used to decide which items should be recommended. Preference rules that consist of a condition (*If ...*), a consequent (*then ...*) and an optional priority value are a straightforward and natural representation mechanism. These logical implications bridge the gap between abstract user requirements like looking for a gift or having no smoking experience and more technical product characteristics such as brand names or different formats. If none of the items in the product repository satisfies all applicable restrictions, then lower priority personalization filters can be relaxed until at least n items are part of the result set (see for instance [17, 15] for a discussion on relaxation strategies).

Note that knowledge-based recommenders in their pure form (as opposed to utility-based ones) do not compute utility scores or a ranking for recommended items, but solely decide if an item is part of their result set or not. Therefore, rec_{kb} is defined as follows:

$$rec_{kb}(i, u) = \begin{cases} 1 & : KB \cup UM_u \vdash_{rs} i \\ 0 & : \text{else} \end{cases} \quad (4)$$

The knowledge-based RS takes the knowledge base itself (KB) and the user model of the current user (UM_u) as input. The assumed utility of an item i for user u is 1 and therefore recommended if item i is inferred by the reasoning procedure of the recommender system ($\vdash_{rs}$) and 0 otherwise.

However, in case the knowledge base consists solely of soft preference constraints c that are annotated with a priority value p_c , it is possible to compute utility scores for each item as follows:

$$rec'_{kb}(i, u) = \frac{1}{\sum_{c \in KB} p_c} \sum_{\substack{c \in KB \wedge \\ c \cup UM_u \vdash_{rs} i}} p_c \quad (5)$$

Consequently, $rec'_{kb}(i, u)$ is the sum of priorities of those constraints that do not need to be relaxed in order to have item i in the result set. $\frac{1}{\sum p_c}$ normalizes the utility score to the range 0..1. In our evaluation we rank the knowledge-based recommendations according to their utility scores.

Typically, the knowledge base is engineered with the help of domain expertise. We denote a knowledge base that was acquired from domain experts KB_{exp} .

3.3 Collaborative constraint-based recommendation

According to Burke's taxonomy of hybrid algorithm designs, the proposed approach can be classified as a *meta-level* variant [8]. Its primary characteristics are that the results of one recommender system form the input of another. A meta-level collaborative constraint-based hybrid employs collaborative filtering to generate rule-based preferences for a knowledge-based RS. Hence the collaborative filtering algorithm does not recommend items from the product catalog but rather from the constraint base (KB_{gen}) that express functional dependencies between formulated user requirements and the respective product properties of bought items. These constraints can be pre-generated offline from community data and not at runtime when a user u requests a recommendation.

This preprocessing step utilizes rule templates T and takes the user models of a set of users U as input. Furthermore, a product catalog I that contains the product properties $prop_i$ for all items i has to be available. The user model for user u consists of a set of requirements Rq_u and actually bought items or unary ratings R_u . Consequently, in a preprocessing step a set of rule-based preferences C_u for each user model u is generated as follows:

$$C_u = \{rq \rightarrow prop_i | rq \in Rq_u \wedge i \in R_u \wedge prop_i \in I \wedge T\} \quad (6)$$

Note, that all generated constraints must be consistent with the rule templates T in order to avoid the automated generation of spurious functional dependencies. T is a logical expression on rq and $prop$ that restricts rule-based preferences to a limited set of patterns. For instance, in the motivating example only three different forms of preference

rules have been allowed. Thus, T was defined as follows: $T : rq = for_whom \wedge prop = brand \vee rq = experience \wedge prop = format \vee rq = cost \wedge prop = price$.

The collaborative filtering step consequently recommends all rule-based preferences from the neighborhood of the current user and thus treats these constraints as 'items'. The utility scores for each constraint c that is contained in the result set of the collaborative recommendation step for user u are summed up according to equation (2).

The result set and thus the recommended knowledge base consists of all rule-based preferences from users in the neighborhood of u .

$$rs_{cf}(u) = \{c | v \in N_u \wedge c \in C_v\} \quad (7)$$

Consequently, the resulting knowledge base of collaborative filtering forms the input model for the subsequent knowledge-based recommendation step. Note, that in the following equation we abbreviate $rs_{cf}(u)$ by KB_u for reasons of readability.

$$rec_{meta}(i, u) = \frac{1}{\sum_{c \in KB_u} rec_{cf}(c, u)} \sum_{\substack{c \in KB_u \wedge \\ c \cup UM_u \vdash_{rs} i}} rec_{cf}(c, u) \quad (8)$$

Thus, the meta-level recommender $rec_{meta}(i, u)$ returns in analogy to knowledge-based recommendation an utility score for each item with the difference that priorities are dynamically determined by the scores of the collaborative filtering step $rec_{cf}(c, u)$. In the following we evaluate the proposed method and research the hypothesis that the proposed meta-level hybrid is advantageous in terms of accuracy and serendipity of produced results compared to an existing knowledge-based RS.

4. EVALUATION

The evaluation is based on a dataset extracted from log information and profiling data of a conversational recommender system for premium cigars over a period of 12 months (Mar/07 - Feb/08). We identified 339 sessions of different anonymous users (U) that interacted with the conversational recommender system and actually added an item to their basket. The dataset comprises a total of 2791 explicit user requirements (Rq) and 449 bought items (R) that are interpreted as binary ratings. Furthermore, a total of 143 different items (I) are offered by the webshop. The sparsity level of the dataset is extremely high and it can be computed as the share of empty entries in the user-item matrix $1 - \frac{|R|}{|I| \cdot |U|}$ which results in a sparsity level of 99,07% ($|R| = 449$, $|I| = 143$ and $|U| = 339$). Note, that the number of explicit user requirements does not impact upon the sparsity level itself as only product items will have to be predicted.

We compared an existing knowledge-based algorithm RS_{exp} with our proposed collaborative knowledge-based meta-level hybrid RS_{meta} with respect to accuracy and serendipity. Furthermore, another knowledge-based variant (RS_{gen}) was developed that acts as a baseline method, applying all the constraints that have been generated from community transactions. The knowledge base of the RS_{exp} variant was developed with the help of domain experts in 2003. Since then it has been used to make knowledgeable recommendations

to users of the online e-commerce platform *www.premium-cigars.ch*. A total of 47 filter rules encode recommendation knowledge that takes the experience level, taste and format requirements as well as quantity and price preferences into account. More details on the knowledge base are given in previous work [25, 24]. The knowledge-base is enacted by the conversational *Advisor Suite* recommender [14]. The RS_{gen} and RS_{meta} algorithms do not start with any prior domain knowledge (except for some rule templates) but solely exploit the community transactions. However, RS_{gen} lacks the collaborative filtering step that personalizes the knowledge base according to the current user's neighborhood. For our experiments only binary constraints were generated, each combining a restriction on a user requirement and a product property.

The two knowledge-based recommenders RS_{exp} and RS_{gen} produce ranked lists as described in Subsection 3.2 and RS_{meta} uses the ranking scheme described in Subsection 3.3.

4.1 Methodology

Each of the 339 user sessions is represented by a user model containing a set of requirements (Rq) and a set of identifiers of those items that caught the user's attention (R), i.e. were added to the basket. The evaluated algorithms do not require the current user's ratings for determining her neighborhood but solely her Rq . Therefore a *Given 0* evaluation method is applied where all rated items R are used for testing (i.e. $testset_u = R_u$).

The RS_{meta} and RS_{gen} methods require community data in order to determine a knowledge base. Furthermore, the experiments followed a *leave-one-out* approach, where all user models except the one of the current user were exploited for learning the model that is then applied on the current user model for making predictions. The experiments were conducted exactly once for each user and due to the *Given 0* method the testset, i.e. R , was static for each user. At most ten recommendations were computed for each user, i.e. for a user u , $|recset_u| = 10$ for all experiments. It is assumed that recommendations contained in the testset are successful hits, i.e. $hits_u = recset_u \cap testset_u$. The accuracy of recommendations is computed using the Precision (Prec), Recall (Rec) and F1 metrics [22, 13]:

$$\begin{aligned} Prec &= \frac{|hits_u|}{|recset_u|} \\ Rec &= \frac{|hits_u|}{|testset_u|} \\ F1 &= \frac{2 \cdot Prec \cdot Rec}{Prec + Rec} \end{aligned}$$

The Precision metric gives the share of successful recommendations from the total number of computed recommendations, while the Recall metric computes the ratio of hits and the theoretical maximum number of hits due to the testing set size. The average test set size is 1.32 ($\frac{|R|}{|U|}$). Consequently, if it were possible to predict all items in the testing set, Recall would be 100% but Precision only around 13%! Another important metric in our evaluation is the Catalog Coverage metric (CC):

$$CC = \frac{|\bigcup_{u \in U} recset_u|}{|I|} \quad (9)$$

Table 4: Results on RS_{gen} , RS_{exp} and RS_{meta} with different neighborhood sizes k

Algorithm	<i>Prec</i>	<i>Rec</i>	<i>F1</i>	<i>CC</i>
RS_{gen}	0.97%	7.35%	1.72%	22.38%
RS_{exp}	2.03%	7.80%	3.22%	90.91%
RS_{meta} , $k = 1$	1.30%	9.8%	2.29%	98.60%
RS_{meta} , $k = 5$	1.74%	13.14%	3.07%	86.01%
RS_{meta} , $k = 10$	1.59%	12.03%	2.81%	70.63%
RS_{meta} , $k = 15$	1.48%	11.14%	2.60%	64.34%
RS_{meta} , $k = 20$	1.48%	11.14%	2.60%	57.34%
RS_{meta} , $k = 25$	1.36%	10.25%	2.40%	54.55%
RS_{meta} , $k = 30$	1.24%	9.35%	2.19%	51.75%
RS_{meta} , $k = 50$	1.24%	9.13%	2.14%	41.26%

This indicates the proportion of items that are ever recommended to a user [13] and it can be used for analyzing the diversity or serendipity of recommendations.

4.2 Results

Table 4 presents the performance of the RS_{exp} and RS_{gen} algorithms as well as the different RS_{meta} variants. For collaborative learning of the knowledge base the neighborhood size k was varied from 1 to 50, where $k = 1$ means that only constraint candidates from the most similar peer in the community became part of the knowledge base and $k = 50$ signifies that constraints from the fifty most similar user profiles were considered.

As can be seen in Table 4, all recommenders outperform the RS_{gen} that acts as a baseline. In terms of recall, the RS_{meta} algorithm variants clearly achieve better results than the original RS_{exp} approach (13.14% vs. 7.80%), indicating that the collaborative knowledge-based hybrid is able to make much more accurate predictions, i.e. hits, than the traditional RS_{exp} . Quite surprising, the RS_{gen} algorithm obtains a comparable Recall as RS_{exp} . However, RS_{exp} reaches best results in terms of precision as the RS_{exp} algorithm produces recommendation lists with on average 6.35 elements while the RS_{meta} and the RS_{gen} recommend exactly 10 items to each user. This is a side effect of the relaxation procedure implemented in the *Advisor Suite* system [14] that relaxes constraints in a stepwise manner if no single item fulfills all applicable constraints. We varied RS_{meta} with different neighborhood sizes k from 1 to 50. Interestingly, five users seems to be the ideal neighborhood size. This can be explained by the fact that explicit user requirements are used for determining user similarities. As only relatively few different combinations of requirements are possible compared to product ratings combinations in traditional collaborative filtering, larger neighborhoods seem to add more noisy constraints to the derived result sets.

With respect to the serendipity of results we can observe that only the RS_{meta} variant with a neighborhood size of one has a higher catalog coverage than RS_{exp} . Interestingly, the catalog coverage continuously decreases as the neighborhood size k increases. An explanation for this behavior might be that larger neighborhoods place more weight on common constraints. Consequently, the knowledge base and thus the recommendation lists become less diverse.

To summarize, the evaluation showed that the RS_{meta} hybrid clearly increases the number of successful predictions

in the given domain compared with the traditional RS_{exp} approach. The variation of the neighborhood size impacts upon the accuracy of the algorithm and significantly reduces the Catalog Coverage as a measure for the diversity of produced recommendations. However, these experiments should be seen as a first step, demonstrating some initial payoffs, however the exploration of further algorithm variants as well as different evaluation scenarios remains as a topic for future research.

5. RELATED WORK

Knowledge-based and collaborative recommendation paradigms were also integrated into the FindMe systems for assisted browsing [10]. They use a cascaded hybridization approach where collaborative filtering is utilized to discriminate between items that fulfill the same set of constraints [6].

Adomavicius and Tuzhilin present the 1:1Pro system that mines customer transaction data to produce personalization rules [2]. Their innovation lies in proposing an expert-driven validation process that is facilitated by a sophisticated template mechanism and groupings [1]. Adding a validation step after the knowledge base is generated could be an interesting future extension to our work.

Mining association rules in itemsets to derive recommendations has already been described in [21]. However, in our approach more abstract rules that map user requirements to product characteristics are identified. Furthermore, O’Sullivan et al. [18] explored data mining techniques to reduce the sparsity of rating matrices for collaborative filtering.

Felfernig et al. [11] presented an integrated environment for the development and maintenance of knowledge-based sales advisory applications that supports regression testing based on positive and negative test cases of the knowledge base after the completion of maintenance activities. Nevertheless, knowledge base development and maintenance tasks are still primarily manual activities that have to be performed by domain experts while our approach is based on automated knowledge acquisition.

The idea of reducing the knowledge acquisition bottleneck when constructing knowledge bases for expert system has always been an active research area. For instance Pham and Dimov proposed an efficient mining algorithm for deriving IF- THEN rules from positive and negative examples [19]. However, in our case we do not construct a single knowledge base for all users, but rather the collaborative filtering paradigm dynamically constructs a personalized knowledge base for each user.

Another line of related work lies in the manual construction of knowledge bases from domain expertise. Ainscough et al. for instance researched this issue from the viewpoint of Marketing Science [4]. They described the manual acquisition of selling expertise from the sales scripts of several sales experts, while we propose the automated acquisition of this knowledge from past transaction data.

6. CONCLUSION

This paper proposed a meta-level hybrid that utilizes collaborative filtering to learn personalization constraints. This dynamically determined knowledge base subsequently forms the input for a knowledge-based recommendation algorithm

for the computation of personalized recommendations. The approach mines functional dependencies between disclosed user requirements and product properties that are employed for generating recommendations. An evaluation indicated that the new algorithm produces more accurate results than an algorithm using a manually encoded knowledge base with domain expertise. Future work will develop the technique further to cope with n-ary constraints as well as explore additional application scenarios for this technique such as the validation of expert knowledge or the partial automation of knowledge base maintenance activities.

7. REFERENCES

- [1] G. Adomavicius and A. Tuzhilin. Expert-driven validation of rule-based user models in personalization applications. *Data Mining and Knowledge Discovery*, 5:33–58, 2001.
- [2] G. Adomavicius and A. Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, 2001.
- [3] G. Adomavicius and A. Tuzhilin. Towards the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 2005.
- [4] T. L. Ainscough, T. E. DeCarlo, and T. W. Leigh. Building expert systems from the selling scripts of multiple experts. *The Journal of Services Marketing*, 10(4):23–40, 1996.
- [5] M. Balabanovic and Y. Shoham. Fab: Content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [6] R. Burke. Integrating knowledge-based and collaborative-filtering recommender systems. In *AAAI-Workshop on Artificial Intelligence for E-Commerce*, pages 69–72, 1999.
- [7] R. Burke. Knowledge-based recommender systems. *Encyclopedia of Library and Information Systems*, 69(2), 2000.
- [8] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
- [9] R. Burke. Hybrid web recommender systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 377–408, Heidelberg, Germany, 2007. Springer.
- [10] R. D. Burke, K. J. Hammond, and B. C. Young. The findme approach to assisted browsing. *IEEE Expert*, July/Aug.:32–40, 1997.
- [11] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. An integrated environment for the development of knowledge-based recommender applications. *International Journal of Electronic Commerce*, 11(2):11–34, 2006.
- [12] W. B. Frakes and R. Baeza-Yates, editors. *Information Retrieval, Data Structure and Algorithms*. Prentice Hall, 1992.
- [13] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22(1):5–53, 2004.
- [14] D. Jannach. Advisor suite - a knowledge-based sales advisory system. In L. S. Lopez de Mantaras, editor, *16th European Conference on Artificial Intelligence - Prestigious Applications of AI (PAIS)*, pages 720–724. IOS Press, 2004.
- [15] D. Jannach. Finding preferred query relaxations in content-based recommenders. In *IEEE Intelligent Systems Conference (IS)*, pages 355–360, Westminster, UK, 2006. IEEE Press.
- [16] K. McCarthy, J. Reilly, L. McGinty, and B. Smyth. Experiments in Dynamic Critiquing. In *Proceedings of the Intelligent User Interfaces Conference (IUI)*, pages 175–182, 2005.
- [17] N. Mirzadeh, F. Ricci, and M. Bansal. Supporting user query relaxation in a recommender system. In *5th International Conference on E-Commerce and Web Technologies (EC-Web)*, pages 31–40, Zaragoza, Spain, 2004. Springer.
- [18] D. O’Sullivan, B. Smyth, and D. Wilson. Preserving recommenders accuracy and diversity in sparse datasets. *International Journal of Artificial Intelligence Tools*, 13(1):219–235, 2004.
- [19] D. Pham and S. S. Dimov. An efficient algorithm for automatic knowledge acquisition. *Pattern Recognition*, 30(7):1137–1143, 1997.
- [20] F. Ricci and H. Werthner. Case base querying for travel planning recommendation. *Information Technology and Tourism*, 3:215–266, 2002.
- [21] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *ACM Conference on e-Commerce (EC)*, pages 158–167, 2000.
- [22] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based Collaborative Filtering Recommendation Algorithms. In J. Carroll, editor, *10th International World Wide Web Conference*, 2001.
- [23] J. B. Schafer, D. Frankowski, and J. H. S. Sen. Collaborative filtering recommender systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*, pages 291–324, Heidelberg, Germany, 2007. Springer.
- [24] M. Zanker and M. Jessenitschnig. Case-studies on exploiting explicit customer requirements in recommender systems. *User Modeling and User-Adapted Interaction: The Journal of Personalization Research*, A. Tuzhilin and B. Mobasher (Eds.): *Special issue on Data Mining for Personalization*, (to appear 2008).
- [25] M. Zanker, M. Jessenitschnig, D. Jannach, and S. Gordea. Comparing recommendation strategies in a commercial context. *IEEE Intelligent Systems*, 22(May/Jun):69–73, 2007.