PROCEEDINGS

The 8th Workshop on

Intelligent Techniques for Web Personalization and Recommender Systems

ITWP 2010

Editors:

Bamshad Mobasher, Dietmar Jannach, Sarabjot Singh Anand

BIG ISLAND OF HAWAII, JUNE 20 2010

In Conjunction with the 2010 International Conference on
User Modeling, Adaptation and Personalization (UMAP 2010)

Copyright and Bibliographical Information

Proceedings of the 8th Workshop on Intelligent Techniques for Web Personalization and Recommender Systems (ITWP 2010), in conjunction with the International Conference on User Modeling, Adaptation, and Personalization (UMAP 2010). June 20-24, 2010. Bamshad Mobasher, Dietmar Jannach, and Sarabjot Singh Anand (editors).

Workshop Web Site

Additional information about the workshop, including the electronic versions of accepted papers and links to related workshops will be maintained at the workshop Web site:

http://ls13-www.cs.uni-dortmund.de/homepage/ITWP2010/

Workshop Co-Chairs

Bamshad Mobasher, DePaul University, USA
Dietmar Jannach, TU Dortmund, Germany
Sarabjot Singh Anand, University of Warwick, UK

Program Committee

Esma Aïmeur, Université de Montréal, Canada
Gediminas Adomavicius, CSOM, University of Minnesota
Liliana Ardissono, University of Torino, Italy
Bettina Berendt, K.U.Leuven, Belgium
Shlomo Berkovsky, CSIRO, Australia
José Luís Borges, University of Porto, Portugal
Derek Bridge, University College York, Ireland
Alexander Felfernig, Technical University Graz, Austria
Gerhard Friedrich, University Klagenfurt, Austria
Rayid Ghani, Accenture, USA
Andreas Hotho, University of Karlsruhe, Germany
Alipio Jorge, University of Porto, Portugal
Zan Huang, Pennsylvania State University, USA
Mark Levene, University College, London, UK
Frank Linton, The MITRE Corporation, USA
Stuart E. Middleton, University of Southampton, UK
Dunja Mladanic, Jožef Stefan Institute, Slovenia
Alexandros Nanopulos, Aristotle University of Thessaloniki, Greece
Olfa Nasraoui, University of Louisville, USA
Claire Nedellec, Université Paris Sud, Paris, France
George Paliouras, Demokritos National Centre, Greece
Naren Ramakrishnan, Virginia Tech, USA
Francesco Ricci, Free University of Bozen/Bolzano, Italy
Lars Schmidt-Thieme, University of Hildesheim, Germany
Spiros Sirmakessis, University of Patras, Greece
Barry Smyth, University College Dublin, Ireland
Markus G. Stolze, IBM Watson Research Center, NY
Alex Tuzhilin, Stern School of Business, New York University, USA
Suk-Chung Yoon, Widener University, Pennsylvania, USA
Markus Zanker, University Klagenfurt, Austria
Daniel Zeng, University of Arizona

Foreword

Web Personalization and recommendation systems have been steadily gaining ground as essential components of today's Web based applications, including in e-commerce and the delivery of business services, and in providing support for Web search and navigation in information rich domains. The proliferation of Web 2.0 applications has allowed users to go beyond simple consumers of information and instead actively participate in shaping collaborative environments in which users, resources, and user-provided content are all networked together. There is, therefore, an increased need for more intelligent and personalized tools that help users navigate these complex information spaces. These tools include a new generation of recommender systems that integrate multiple online channels, are more scalable and more adaptive, and can better handle user interactivity. To achieve this, such applications must rely on intelligent techniques from AI, machine learning, Web mining, statistics, and user modeling in order to leverage and mine all available data, including user profiles, the content and meta-data associated with resources, and underlying network structures.

The aim of this workshop is to bring together researchers and practitioners From Web Mining, Web Personalization, Recommender Systems, and User Modeling communities in order to foster an exchange of information and ideas and to facilitate a discussion of current and emerging topics related to the development of intelligent Web personalization and Recommender Systems. This workshop represents the 8th in a successful series of ITWP workshops that have been held at IJCAI and AAAI since 2001 and would be – after the successful events at AAAI'07, AAAI'08, and IJCAI'09 - the 3rd combined workshop on ITWP and Recommender Systems.

This year's workshop attracted a number of high-quality contributions of which six papers were accepted for presentation at the workshop. These accepted papers span a variety of issues and techniques related to personalization and recommender systems. Specifically, the papers deal with such topics as the integration of domain ontologies in collaborative recommendation; the use of data mining techniques such as association rule discovery in recommendation; the generation of user recommendations based by integrating and aggregating online data sources; the use of rating frequencies as the basis for generating prediction; the use of Bayesian networks to model preferences from customer feedback; and non-traditional personalization models such as reciprocal recommendation. This year's workshop also includes two invited addresses: "The user side of personalization: how personalization affects the users" by Professor Peter Brusilovsky from the University of Pittsburgh; and "Content-based Recommender Systems: problems, challenges and current research directions" by Professor Giovanni Semeraro from the University of Bari, Italy.

ITWP 2010 Organizing Committee
Bamshad Mobasher, Dietmar Jannach, and Sarabjot Singh Anand
June 2010

Table of Contents

Invited Talk

The user side of personalization: How personalization affects the users

Peter Brusilovsky
University of Pittsburgh

Abstract:

The personalization algorithms are polished on log data and ready to face real users. But what you can expect when real users hit the newly minted personalized Web site?

The talk will focus on user behavior in personalized Web systems and examines how user behavior is affected by personalized guidance and recommendation. It will review the results of several long-term studies of personalized systems and discuss both local impact (whether the users follow recommendations) and global impact of personalization.

Biography:

Peter Brusilovsky has been working in the field of adaptive educational systems, user modeling, and intelligent user interfaces for more than 20 years. He published numerous papers and edited several books on adaptive hypermedia and the adaptive Web. Peter is currently an Associate Professor of Information Science and Intelligent Systems at the University of Pittsburgh, where he directs Personalized Adaptive Web Systems (PAWS) lab. Peter is the Associate Editor-in-Chief of IEEE Transactions on Learning Technologies and a board member of several journals including User Modeling and User Adapted Interaction, ACM Transactions on the Web, and Web Intelligence and Agent Systems. He is also the current President of User Modeling Inc., a professional organization of user modeling researchers.

Invited Talk

Content-based Recommender Systems: problems, challenges and current research directions

Giovanni Semeraro
University of Bari

Abstract:

Content-based recommender systems (CBRS) analyze a set of objects, usually textual descriptions of items previously rated by a user, and build a model of user interests, called user profile, based on the features of the objects rated by that user. The user profile is then exploited to recommend new potentially relevant items. In spite of the growing importance of collaborative filtering algorithms over the last years, Web 2.0 and the huge amount of user generated content, such as tags, annotations, folksonomies, etc., are providing new opportunities and challenges for CBRS. The talk discusses the main problems which cause some limitations of CBRS, such as overspecialization and limited availability of content, and describes current research directions for overcoming them, including: defeating homophily in recommender systems: introducing serendipity for recommendation diversification; knowledge infusion into CBRS: exploiting open knowledge sources (Wikipedia, folksonomies) for improving recommendation algorithms; and cross-language recommender systems: algorithms for learning multilingual content-based profiles.

Biography:

Giovanni Semeraro is Associate Professor at the University of Bari "Aldo Moro" in Italy since November 1, 1998, where he leads the research group SWAP (Semantic Web Access & Personalization, http://www.di.uniba.it/~swap/) at the Department of Computer Science. His main research interests fall into the following areas: Intelligent Information Access, Recommender Systems, Information Mining, Machine Learning, Personalization and User Modelling, Natural Language Processing. He has been responsible for 10 international and national projects and 16 research contracts. He is editor of 8 international books and author of more than 300 scientific papers published in international journals, books, conference and workshop proceedings.

Neighborhood-restricted mining and weighted application of association rules for recommenders

Fatih Gedikli and Dietmar Jannach

Technische Universität Dortmund,
44227 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Association rule mining algorithms such as Apriori were originally developed to automatically detect patterns in sales transactions and were later on also successfully applied to build collaborative filtering recommender systems (RS). Such rule mining-based RS not only share the advantages of other model-based systems such as scalability or robustness against different attack models, but also have the advantages that their recommendations are based on a set of comprehensible rules. In recent years, several improvements to the original Apriori rule mining scheme have been proposed that for example address the problem of finding rules for rare items. In this paper, we first evaluate the accuracy of predictions when using the recent IMSApriori algorithm that relies on multiple minimum-support values instead of one global threshold. In addition, we propose a new recommendation method that determines personalized rule sets for each user based on his neighborhood using IMSApriori and at recommendation time combines these personalized rule sets with the neighbors' rule sets to generate item proposals. The evaluation of the new method on common collaborative filtering data sets shows that our method outperforms both a standard IMSApriori recommender and a nearest-neighbor baseline method. The observed improvements in predictive accuracy are particularly strong for sparse data sets.

1 Introduction

Association rule mining is a popular knowledge discovery technique which was designed as a method to automatically identify buying patterns in sales transactions, or, in a more broader view, to detect relations between variables in databases. One of the earliest efficient techniques to find such rules is the Apriori algorithm proposed by Agrawal and Srikant in [AS94]. A common example of an association rule that could be found in the sales transactions in a supermarket could be [LHM99]:

$$cheese \Rightarrow beer \quad [support = 10\%,\ confidence = 80\%]$$

which can be interpreted that in 10% of all transactions beer and cheese were bought together (support of the rule) and that in 80% of the transactions, in which cheese was bought, also beer was in the shopping basket (confidence). Confidence and support are thus statistical measures that indicate the "strength" of the pattern or rule.

Quite obviously, the knowledge encoded in such automatically detected association rules (frequent itemsets) can be exploited to build recommender systems (RS)[1]. A simple recommendation algorithm capable of producing a "top-N" list of items could include the following steps: (1) Use Apriori to detect all association rules that surpass a minimum support threshold. These rules can be mined from real purchase data or from "like" statements in a rating database. (2) Take those rules that are "supported" by the target user (i.e., where the user has purchased all items on the rule's left-hand-side) and compute the set of items that are predicted by those rules and which the user has not yet purchased. (3) Sort the recommendation list by the confidence values of the involved rules. Early successful experiments using such a method for recommendation purposes with a slight variation of the prediction scheme are for example reported in [SKKR00].

Since the rules can be mined in an offline model-learning phase, rule mining-based approaches do not suffer from scalability problems like memory-based algorithms [SMB07]. A further advantage of these approaches lies in the fact that the underlying model, i.e., the set of rules, is explicit and comprehensible to users. Thus, not only interesting consumer behavior phenomena can be learned from it, the rules can also be used to explain the recommendations to end users as to increase the users' confidence in the system's proposals. Finally, from a business perspective, the explicit rule bases can also be easily extended manually with additional domain knowledge, which is not easily possible with other learning-based recommendation methods.

While the accuracy of rule mining-based recommenders is comparable to nearest-neighbor (kNN) collaborative filtering approaches, using the original Apriori algorithm can lead to the problem of reduced coverage as shown in [SMB07]. This phenomenon can be caused by the usage of a global minimum support threshold in the mining process, which leads to the effect that no rules for rare items can be found. Lin et al. [LAR02] therefore propose an "adaptive-support" method, in which the minimum support value is determined individually for each user or item (depending on whether item associations or user associations are used). Their experiments show a slight increase in accuracy when compared with the baseline kNN-method.

More recently, Kiran and Reddy [KR09] proposed a new method called IMSApriori that uses a particular metric to determine appropriate minimum support values per item (see also [LHM99]) in order to mine rare itemsets; their experiments indicate that this method is better suited to mine rare itemsets than previous methods. An evaluation of the approach for recommendation purposes has however not been done so far.

In this work, we evaluate the predictive accuracy of a recommender system based on the IMSApriori algorithm and describe our extension to the *Frequent Itemset Graph* used in [NM03b] for enabling a fast recommendation process. In addition, we propose a new scheme for association rule-based recommendation called NRR (*Neighborhood-restricted Rule-based Recommender*), which is based on the idea to learn a personalized set of rules for each user based on his nearest

[1] See [AT05] for an overview.

neighbors and not on the whole database of transactions, see also [Zan08]. Similar to kNN-approaches, the underlying idea of this is that close neighbors will be better predictors than others. The user's personalized knowledge base is then combined with the rule sets of his nearest neighbors to generate recommendation lists.

The paper is organized as follows. After an example and the introduction of the basics of the used rule mining methods, we describe our NRR method to learn personalized rule learning and prediction generation in detail. Afterwards, the results of an evaluation on typical CF data sets with different density levels and parameter settings are discussed. The paper ends with a conclusion and a short discussion of further works.

2 Example

Let us illustrate the different ideas in this paper with a simplified example. Consider the rating database in Figure 1 in which a "1" indicates that a user liked or purchased an item, a "0" corresponds to a dislike statement. Empty cells mean that no information is available. Let us assume that our goal is to make a recommendation for *User1*.

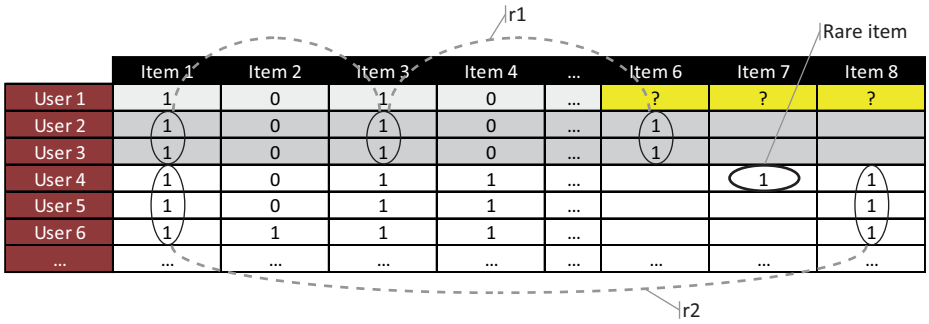


Fig. 1. Example setting for personalized rule bases.

Among others, a rule mining algorithm could detect rules such as

r1: $Item1, Item3 \Rightarrow Item6$ [$support = 33\%, confidence = 33\%$] and

r2: $Item1 \Rightarrow Item8$ [$support = 50\%, confidence = 50\%$]

where the second rule is "stronger" as it is supported by more evidence. Note that *Item7* is a "rare" item, i.e., only few ratings (in that simplified case only one) are available. When using the standard Apriori algorithm, a global minimum support threshold value is used in order to avoid the effect of "rule explosion". This however leads to the effect, that no rules can be learned for rare items because they never reach the global threshold value. Thus, as mentioned above, variations to the Apriori scheme like IMSApriori have been developed that employ multiple minimal support values to take the relative frequency of the items into account.

When making a prediction based on the standard scheme described in the introduction, both rule $r1$ and $r2$ apply as *User1* has rated both *Item1* and *Item3* positively. Since the confidence value of $r2$ is the higher one, the recommender would recommend *Item8* (place *Item8* before *Item6* in the recommendation list). The idea in our approach however is that closer neighbors are better predictors than other users. In the example, it could therefore be a good idea to recommend *Item6* because rule $r1$ is supported by the ratings of *User2* and *User3* who are very similar to the target user *User1* in their rating behavior (in that case even identical).

The NRR method proposed in this paper works as follows. First, we learn a personalized set of rules for each user by taking only the n closest neighbors into account. In an extreme setting, we could only use the ratings of *User2* and *User3* for learning the rules for *User1*. In that case, rule *r1* would be learned. The rule base for *User4*, on the other hand, would probably also include $r2$. When again applying the standard prediction scheme, *Item6* would now be recommended for *User1* as intended. However, the coverage of that approach could be very limited. Thus, we propose a neighborhood-based prediction scheme, in which also the rules of the neighbors are taken into account. When recommending items for *User1* we would therefore use the rules of *User2* and *User3*, but probably also those of *User4* (which include the rule $r2$). In that case, coverage is increased again. At the same time - if we give limited weights to rules of farther-away neighbors - $r1$ can remain the dominating rule and cause *Item6* to be at the top of the recommendation list without losing the other rules.

3 Algorithms

In the following we will shortly summarize the rough ideas of the used rule mining approaches Apriori and IMSApriori in order to give the reader a quick overview of the algorithm parameters that were varied in the experimental evaluation. In addition, we will describe how the *Frequent Itemset Graph* proposed, e.g., in [NM03b], has to be extended for a recommender based on IMSApriori.

Apriori. The original Apriori algorithm [AS94] works by iteratively generating a set of *candidate itemsets* in multiple phases. In our example above, it will start by constructing one-element itemsets, such as $\{Item1\}$ and $\{Item2\}$, and then check if these itemsets have minimum support, where the support of an itemset $X \Rightarrow Y$ is defined as the ratio of the number of transactions containing $X \cup Y$ to the number of all transactions. Itemsets that have not enough support are pruned. In the next phase, the remaining itemsets are combined with one more (frequent) element and checked against the database. This process is repeated until no more candidates can be generated. Overall, one of the ideas of the algorithm's implementation is that "any subset of a large itemset must be large"[2] [AS94].

[2] Frequent itemsets were originally called *large* itemsets.

Once all frequent itemsets are detected, association rule mining approaches use further quality metrics to measure the significance of the detected rules. The *confidence* of a rule $X \Rightarrow Y$ is defined as $\frac{support(X \cup Y)}{support(X)}$, which is a common metric to prune uninteresting rules. The threshold values for minimum *confidence* and *support* are often empirically determined and have to be specified by the user.

IMSApriori. In order to deal with the problem of "missing rules" for rare, but interesting itemsets, different proposals have been made. IMSApriori [KR09], which is used in this work, is a very recent one that builds on the idea of having several minimum support thresholds, an idea also proposed earlier as MSapriori in [LHM99]. The general idea is to calculate a minimum item support (MIS) value for each item with the goal to use a lower support threshold for rare itemsets. In [LHM99] a user-specified value β (between 0 and 1) is used to calculate a MIS value based on the item's support and a lower support threshold value LS as $MIS(item) = max(\beta \times support(item), LS)$. In order to be counted as a *frequent* itemset, itemsets containing only frequent items have to pass a higher minimum support threshold than itemsets consisting of frequent and rare or only rare items. Thus, rare itemsets are found when using a low value for LS while at the same time not too many uninteresting, but more frequent rules are accepted.

Recently, in [KR09], a different approach to calculate the MIS values was proposed because MSapriori fails to detect rare itemsets in situations with largely varying item support values. This phenomenon can be attributed to the fact that due to the constant proportional factor β the difference between the item support and the MIS value decreases when we move from frequent to rare items. The main idea of the *improved* MSapriori (IMSApriori) is therefore the use of the concept of "support difference" (SD) to calculate MIS values as $MIS(item) = max(support(item) - SD, LS)$. SD is calculated as $SD = \lambda(1-\alpha)$, where λ is a parameter "like mean, median, mode, maximum support of the item supports" and α is a parameter between 0 and 1. The net effect of the support difference concept is that the difference between item support values and the MIS values remains constant so that rare items can also be found in data sets with strongly varying item supports. Finally, an itemset is considered to be frequent if its support is higher than the minimum of the MIS values of its components. Regarding the generation of candidates, it has to be noted that the Apriori assumption that all subsets of frequent itemsets are also frequent does not hold and that a different algorithm for finding frequent itemsets has to be used.

Neighborhood-restricted Rule-based Recommender (NRR). As shown in the example, the idea of the herein proposed NRR algorithm is to learn personalized rule sets for each user in an offline phase and to exploit these rule sets in combination with the neighbor's rule sets to generate more accurate predictions. The algorithm is summarized in Algorithm 1. The parameters of the algorithm include – beside the IMSApriori parameters – two neighborhood sizes (for rule learning and for the prediction phase). In the online phase, the calculated user-specific frequent itemsets (UserFISs) of the target user and of the

neighbors of the target user are used to calculate predictions using the *Extended Frequent Itemset Graph* (EFIG) which is introduced in the next section. The resulting confidence scores are weighted according to the similarity of the target user and the neighbor (using Pearson correlation as a metric). These user-specific predictions are finally combined and sorted by the weighted confidence scores.

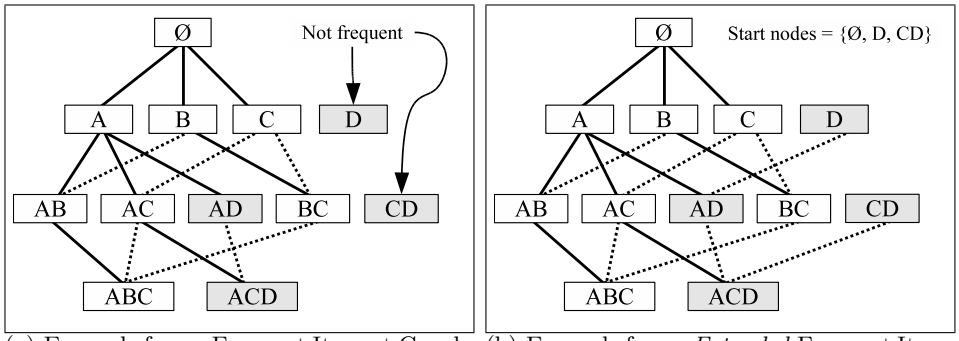
Algorithm 1 NRR algorithm (sketch).

```
In: user, ratingDB, learnNeighborSize, predictNeighborSize, λ, α
Out: recommendedItems
(Offline:) UserFISs = CalcUserFISsIMSApriori(ratingDB, learnNeighborSize, λ, α)
neighborhood = user ∪ findNeighbors(user, predictNeighborSize, ratingDB)
recommendedItems = ∅
for all u ∈ neighborhood do
   userRecs = Recommend(u,buildEFIG(UserFISs(u)))
   weightedUserRecs = adjustConfidenceScoresBySimilarity(userRecs, user, u)
   recommendedItems = recommendedItems ∪ weightedUserRecs
end for
recommendedItems = sortItemsByAdjustedScores(recommendedItems)
```

The Extended Frequent Itemset Graph. The *Frequent Itemset Graph* (FIG) as proposed in [MDLN01] is a data structure to organize the frequent itemsets in a way that allows us to generate recommendations directly from the frequent itemsets (i.e., without the need to derive all association rules first). Figure 3(a) shows such a graph in which the elements of the frequent itemsets are lexicographically sorted and organized in a tree structure where the size of the itemsets are increased on each level. Given, for example, a set of past transactions $T = \{A, D\}$ of user u, recommendations can be produced by traversing the tree in depth-first order and looking for supersets of $\{A, D\}$ in the next level of the graph. In the example, given the superset $\{A, D\}$, C could be recommended to u. The solid arrows in the figure indicate how the graph would be traversed in depth-first order.

Since, however, the assumption that "any subset of a frequent itemset must be frequent" does not hold when using multiple minimum-support values, the standard FIG-based method has to be extended. Let us assume that in the example Figure 3(a) the itemsets $\{D\}$ and $\{C, D\}$ are not frequent, although they are subsets of the frequent itemset $\{A, D\}$ and $\{A, C, D\}$ respectively. We could therefore not recommend $\{A\}$ to users who purchased $\{C, D\}$ or $\{D\}$ alone although this would be plausible.

In our work, we solve this problem by extending the FIG in a way that it also contains all subsets of the frequent itemsets and connect these additional nodes with their supersets as shown in Figure 3(b). In order to find frequent itemsets like $\{A, C, D\}$ from $\{C, D\}$ we re-start the depth-first search on the not-yet-visited parts of the subgraph beginning from the additional nodes. Note



(a) Example for an Frequent Itemset Graph. (b) Example for an *Extended* Frequent Itemset Graph.

Fig. 2. Extended Frequent Itemset Graph approach.

that the negative effects on the scalability of the approach itself are very limited, because only small portions of the graph have to be analyzed in these additional traversals.

4 Experimental Evaluation

The proposed NRR algorithm has been evaluated in an experimental study on different data sets. In particular, the predictive accuracy was measured using different sparsity levels and compared to (a) a recommender based on IMSApriori and a classical prediction scheme and (b) the standard correlation-based kNN-method. In the following, we will summarize the findings of this evaluation.

4.1 Experimental Setup and Evaluation Metrics

Data sets. As data sets for the evaluation, we used the 100k-MovieLens rating database consisting of 100,000 ratings provided by 943 users on 1,682 items and a snapshot of the Yahoo!Movies data set containing 211,231 ratings provided by 7,642 users on 11,915 items[3]. Regarding the user characteristics, the MovieLens data set only contains users who have rated at least 20 items; the minimum number of rated items per user in the Yahoo! data set is 10. In addition, in the Yahoo! data set, each item was at least rated by one user.

In order to test our NRR scheme also in settings with low data density, we varied the density level of the original data sets by using subsamples of different sizes of the original data set as described in [SKKR01]. The smallest subsample contained 10% of the original data. In this subsample, the average number of ratings per user was around 10 as opposed to 100 for the original MovieLens data set. Further measurements were taken in steps of 10% up to the 90% data

[3] http://www.grouplens.org/node/73, http://webscope.sandbox.yahoo.com

set. Four-fold cross-validation was performed for each data set; in each round, the data sets were split into a 75% training set and a 25% test set.

Accuracy metrics. In the study, we aim to compare the predictive accuracy of two rule mining-based methods and the kNN-method. We follow the evaluation procedure proposed in [NM03a] and proceed as follows. First, we determine the set of existing "like" statements (ELS) in the 25% test set and retrieve a top-N recommendation list of length $|ELS|$ with each method based on the data in the training set[4]. In the kNN-case, the rating predictions are converted into "like" statements as described in [SMB07], where ratings above the user's mean rating are interpreted as "like" statements. The set of predicted like statements returned by a recommender shall be denoted as *Predicted Like Statements (PLS)*, where $|PLS| \leq |ELS|$.

We use standard information retrieval accuracy metrics in our evaluation. *Precision* is defined as $\frac{|PLS \cap ELS|}{|PLS|}$ and measures the number of correct predictions in PLS. *Recall*[5] is measured as $\frac{|PLS \cap ELS|}{|ELS|}$ and describes how many of the existing "like" statements were found by the recommender.

In the evaluation procedure, recommendations and the corresponding precision and recall values were calculated for all users in the data set and then averaged. These averaged precision and recall values are then combined in the usual F-score, where $F = 2 * \frac{precision * recall}{precision + recall}$.

Algorithm details and parameters. Regarding the algorithms, note that we used Pearson correlation as a similarity metric both for the kNN-baseline method and for determining the neighborhood in the NRR algorithm. For the kNN-method, we additionally applied *default voting* and used a neighborhood-size of 30, which was determined as an optimal choice in [SKKR01].

The IMSApriori implementation used in the experiments corresponds to above-described algorithm and learns the rules from the whole database of transactions. Recommendations are generated by using the Extended Frequent Itemset Graph structure.

For the NRR method, two further parameters can be varied: *neighborhood-size-learn* is the number of neighbors used to learn association rules; *neighborhood-size-predict* determines on how many neighbors the predictions should be based. The sensitivity of these parameters were analyzed by conducting multiple experiments on the MovieLens data set with a fixed density level of 70%. The value of the parameter *neighborhood-size-predict* was empirically determined to be 100, see Figure 3 (a). To analyze the sensitivity of this parameter, we performed experiments in which we varied the number of neighbors used for making predictions and fixed the parameter *neighborhood-size-learn* at 30 as suggested as an optimal value for this data set in literature.

[4] The top-N recommendation lists are created either based on the confidence of the producing rule or based on the prediction score of the kNN-method.

[5] In [NM03a], this metric is called *coverage*.

We can observe from the figure that the recall value increases from 46% to 61% when moving from a prediction neighborhood size of 10 to 100. At the same time, the precision value stays rather constant at 65% and does not decrease. Afterwards, we fixed the parameter *neighborhood-size-predict* at 100 and analyzed the sensitivity of *neighborhood-size-learn*, see Figure 3 (b). It can be seen that the initial value of 30 was actually a good choice for this parameter.

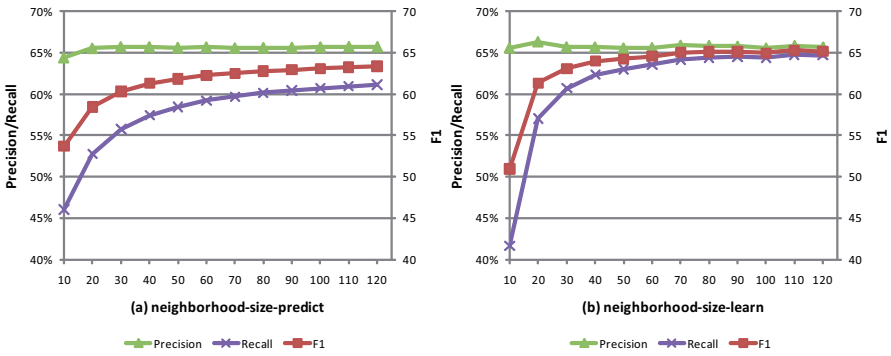


Fig. 3. Sensitivity of the parameters *neighborhood-size-predict* (a) and *-learn* (b).

Finding a suitable lower support threshold value (LS) for the rule learning methods is challenging. In [MDLN01], the authors argue that higher LS values are in general more desirable because they lead to a smaller model size, as the number of frequent itemsets decreases, which in turn leads to good scalability. Higher LS values however may lead to the effect that no rules for rare items can be found.

In addition, remember that in IMSApriori-based algorithms the minimum item support MIS for each item has to be at least LS, i.e., $MIS(i) >= LS$, for each item i. Therefore, when using a high global LS value, the problem can arise that each item gets the same MIS value, i.e., the value of the global minimum support threshold LS. As a consequence, the IMSApriori algorithm would behave like the standard Apriori algorithm which uses one global lower support threshold value. Note that this phenomenon can also appear when the density level of the data set is very low. A low density level implies low item support values, which can result in MIS values that are below the LS value.

In order to establish fair conditions in our study, we have used individual, empirically-determined LS values for each rule learning algorithm (IMSApriori: 3%; NRR: 19%), which we have then used for all density levels and data sets[6].

Regarding computational complexity, the proposed NRR algorithm runs the IMSApriori algorithm once for each user, which can be done offline during the model building phase. Each IMSApriori execution however only has to consider a

[6] Our current work includes a more detailed analysis of optimal parameter values for specific density levels.

relatively small fraction of the whole database of transactions, or, more precisely, at most *neighborhood-size-learn* transactions, when learning rules. On a standard desktop computer (Intel Core 2 Duo CPU, 2.4 GHz, 3GB RAM), learning the rules for all users takes about 11 minutes for the MovieLens data set at a density level of 70%. The size of the resulting model is determined by the number of existing frequent itemsets. In the baseline IMSApriori recommender, 316 frequent itemsets were found. When using the NRR algorithm, the average model size is about 45 frequent itemsets for each user (MovieLens data set, density-level of 70%). The model size of course strongly depends on the selected LS values.

Results. Figure 4 summarizes the evaluation results for the three algorithms kNN, IMSApriori and NRR. The table shows the average values of the F-score as well as the precision and recall values for the different density levels for both the MovieLens and the Yahoo! data set.

		Density ➔	10%	20%	30%	40%	50%	60%	70%	80%	90%
MovieLens	F1	kNN	30,76	41,48	48,06	51,25	55,39	57,13	58,89	59,99	61,32
		IMSApriori	3,75	32,80	50,00	55,94	59,34	60,97	62,66	62,84	62,82
		NRR	37,14	44,09	50,80	56,54	59,10	61,38	63,10	63,51	63,57
	Precision	kNN	39,85%	53,62%	59,76%	61,73%	64,44%	65,29%	66,18%	66,61%	67,07%
		IMSApriori	5,92%	48,43%	63,03%	64,91%	65,31%	65,35%	65,70%	65,24%	64,48%
		NRR	47,25%	57,73%	63,62%	65,74%	65,50%	65,49%	65,69%	65,30%	64,75%
	Recall	kNN	25,05%	33,83%	40,21%	43,81%	48,57%	50,78%	53,05%	54,57%	56,47%
		IMSApriori	2,76%	24,81%	41,43%	49,15%	54,38%	57,15%	59,90%	60,61%	61,24%
		NRR	30,60%	35,67%	42,28%	49,60%	53,84%	57,75%	60,71%	61,81%	62,42%
YahooMovies	F1	kNN	9,24	15,93	21,91	27,95	33,30	37,53	41,32	44,02	46,29
		IMSApriori	6,95	17,06	24,96	31,95	37,86	41,76	43,84	45,61	47,05
		NRR	15,10	20,70	26,81	31,50	36,53	40,30	43,05	45,00	47,01
	Precision	kNN	10,93%	19,75%	27,31%	35,28%	41,95%	47,35%	52,08%	55,47%	58,23%
		IMSApriori	7,65%	20,20%	30,11%	39,37%	46,07%	51,11%	53,81%	56,28%	57,70%
		NRR	17,38%	24,76%	32,37%	38,71%	44,65%	49,46%	52,73%	54,94%	57,36%
	Recall	kNN	8,01%	13,35%	18,30%	23,15%	27,61%	31,09%	34,24%	36,49%	38,41%
		IMSApriori	6,37%	14,77%	21,32%	26,89%	32,13%	35,31%	36,99%	38,35%	39,73%
		NRR	13,36%	17,79%	22,88%	26,55%	30,90%	34,00%	36,37%	38,11%	39,83%

Fig. 4. Overall average F1, precision and recall values for different density levels.

The results show that our NRR algorithm consistently outperforms the kNN algorithm on the F1-measure and is better than the IMSApriori method in nearly all settings for both data sets. The observed accuracy improvements are particularly high for low density levels, i.e., for sparse data sets. With higher density levels, the relative improvements become smaller for both data sets. As a side-observation, we can see that in settings with medium and higher density levels, also the pure IMSApriori version outperforms the kNN-method, which was not analyzed in previous research. Note that the accuracy gains are stronger for

the MovieLens data set, which can be partially attributed to the fact that the optimal parameters were empirically determined based on this data set.

A closer look at the precision and recall values shows that NRR has particular advantages with respect to the recall measure, i.e., NRR is capable of retrieving more relevant items than the other algorithms while at the same time precision values remain at a comparably high level. Again, this effect is particularly strong when the data sets are very sparse, which is a common situation in most real-world settings.

5 Summary

Association rule mining is a powerful method that has been successfully used for various personalization and recommendation tasks in the past; see for example its recent application for social tag prediction ([HRGM08], [WHD09]).

In this paper we have shown how the personalization of the learned model in rule mining-based approaches to recommendation can help to increase the accuracy of the system's prediction while at the same time the advantages of model-based approaches such as robustness against attacks and the possibility to generate explanations can be preserved.

Additional computational costs for the personalization task arise only in the offline phase in which multiple smaller frequent itemset collections are computed instead of one large one. At run-time, data structures such as the Extended Frequent Itemset Graph can be used to efficiently generate recommendations online. Furthermore, given the explicit and comprehensible nature of the frequent itemsets, these (personalized) frequent itemsets can be easily manually extended with additional manually-engineered domain rules.

Our future work includes the evaluation of our approach on further data sets and a comparison with further algorithms. In addition, in our current work, we conduct experiments in which we first perform probabilistic clustering on the user base and then mine the frequent itemsets for each cluster. While we might not expect significant accuracy gains, this approach will lead to a substantially reduced model size which further improves the scalability of our approach. In addition, the neighborhood-size parameter will not be required in the training phase when a method like *AutoClass* [CKS+93] is used to determine the optimal number of clusters automatically.

References

[AS94] Rakesh Agrawal and Ramakrishnan Srikant, *Fast algorithms for mining association rules in large databases*, Proceedings of the 20th International Conference on Very Large Data Bases (VLDB'94) (Santiago de Chile, Chile), 1994, pp. 487–499.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin, *Toward 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** (2005), no. 6, 734–749.

[CKS+93] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman, *Autoclass: A bayesian classification system*, Readings in knowledge acquisition and learning: automating the construction and improvement of expert systems, Morgan Kaufmann, 1993, pp. 431–441.

[HRGM08] Paul Heymann, Daniel Ramage, and Hector Garcia-Molina, *Social tag prediction*, Proceedings 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'08) (Singapore, Singapore), 2008, pp. 531–538.

[KR09] R. Uday Kiran and P. Krishna Reddy, *An improved multiple minimum support based approach to mine rare association rules*, Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2009 (Nashville, TN, USA), 2009, pp. 340–347.

[LAR02] W. Lin, S. Alvarez, and C. Ruiz, *Efficient adaptive-support association rule mining for recommender systems*, Data Mining and Knowledge Discovery **6** (2002), 83–105.

[LHM99] Bing Liu, Wynne Hsu, and Yiming Ma, *Mining association rules with multiple minimum supports*, Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'99) (San Diego, CA, United States), 1999, pp. 337–341.

[MDLN01] Bamshad Mobasher, Honghua Dai, Tao Luo, and Miki Nakagawa, *Effective personalization based on association rule discovery from web usage data*, Proceedings of the 3rd International Workshop on Web Information and Data Management (WIDM'01) (Atlanta, Georgia, USA), 2001, pp. 9–15.

[NM03a] Miki Nakagawa and Bamshad Mobasher, *A hybrid web personalization model based on site connectivity*, Proceedings of the 2003 WebKDD Workshop (Washington, DC, USA), 2003, pp. 59–70.

[NM03b] ———, *Impact of site characteristics on recommendation models based on association rules and sequential patterns*, Proceedings of the IJCAI'03 Workshop on Intelligent Techniques for Web Personalization (Acapulco, Mexico), 2003.

[SKKR00] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl, *Analysis of recommendation algorithms for e-commerce*, Proceedings of the 2nd ACM Conference on Electronic Commerce (EC'00) (Minneapolis, MN, USA), 2000, pp. 158–167.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Reidl, *Item-based collaborative filtering recommendation algorithms*, Proceedings of the 10th International Conference on World Wide Web (WWW'01) (Hong Kong), 2001, pp. 285–295.

[SMB07] J. J. Sandvig, Bamshad Mobasher, and Robin Burke, *Robustness of collaborative recommendation based on association rule mining*, Proceedings of the 2007 ACM Conference on Recommender Systems (RecSys'07) (Minneapolis, MN, USA), 2007, pp. 105–112.

[WHD09] Jian Wang, Liangjie Hong, and Brian D. Davison, *Tag recommendation using keywords and association rules (RSDC'09)*, ECML PKDD Discovery Challenge 2009 (DC09) (Bled, Slovenia), vol. 497, CEUR Workshop Proceedings, 2009, pp. 261–274.

[Zan08] Markus Zanker, *A collaborative constraint-based meta-level recommender*, Proceedings of the 2008 ACM Conference on Recommender Systems (RecSys'08) (Lausanne, Switzerland), 2008, pp. 139–146.

Ontology-Based Collaborative Recommendation

Ahu Sieg, Bamshad Mobasher, Robin Burke

Center for Web Intelligence
DePaul University, Chicago, Illinois, USA
{asieg, mobasher, rburke}@cdm.depaul.edu

Abstract. Recommender systems have emerged as critical tools that help alleviate the burden of information overload for users. Since these systems have to deal with a variety of modes of user interactions, collaborative recommendation must be sensitive to a user's specific context and changing interests over time. Our approach to building context-sensitive collaborative recommendation is a hybrid one that incorporates semantic knowledge in the form of a domain ontology. User profiles are defined relative to the ontology, giving rise to an *ontological user profile*. In this paper, we describe how ontological user profiles are learned, incrementally updated, and used for collaborative recommendation. We empirically show that the ontological approach significantly improves the accuracy and coverage of recommendations.

1 Introduction

Recommender systems [1] have become essential tools in assisting users to find what they want in increasingly complex information spaces. Collaborative recommender systems typically generate recommendations by identifying neighborhoods for the target user consisting of other users with similar interests or preferences [2].

Typical collaborative recommenders rely on profiles of users represented as flat vectors of ratings or preference scores. Thus, the same collection of user preferences across all items or resources is used as the basis for generating recommendations regardless of the user's current information context or task-related needs. Consider the following example. Suppose Steve buys and rates mystery-detective fiction novels for his own entertainment ("Da Vinci Code"), books on computer science topics ("Python Programming") for work-related purposes, children's books ("Green Eggs and Ham") for his daughter. It makes little sense to represent Steve's *interest in books* in a single representation that aggregates all of these disparate interests without some acknowledgment that they represent different sorts of needs and contexts. The system needs to know the difference between computer books and children's books, as well as Steve's current context (buying a book for his personal reading or for his work), in order to make the most useful recommendation. Furthermore, a system that is aware of this difference may also have the capability of recognizing similarities among syntactically disparate items, and be able to recommend a book on Perl scripting to Steve because he has shown an interest in Python programming.

This scenario exemplifies why it is desirable for intelligent and personalized information systems to be capable of seamlessly integrating knowledge from three sources: the short-term user activity, representing immediate user interests; long-term user profiles, representing established preferences; and existing ontologies that provide an explicit representation of the domain of interest. Such systems will be able to leverage a variety of sources of evidence to provide the best personalized experience for the user, including both the semantic evidence associated with the user's individual interaction, as well as social knowledge derived collaboratively from peer users.

In this paper, we present an approach to collaborative recommendation that effectively incorporates semantic knowledge from ontologies with collaborative user preference information. The salient feature of our framework is the notion of *ontological user profiles* which are instances of a pre-existing domain ontology with numerical annotations associated with concepts derived from users' past behavior and preferences. The ontology represents concepts and relationships in a particular domain of interest, books for example. In this paper, we use the term ontology to refer to a hierarchical concept structure and instances within the knowledge base. Rather than being associated with single atomic entities like individual books, users' choices and preferences are associated with relevant concepts in the ontology. So, the fact that Steve buys computer books such as "Python Programming" can be readily distinguished from his interest in children's books because they occupy disparate places in the book ontology.

We present an algorithm based on spreading activation to incrementally update these user profiles, as a result of ongoing user interaction, in a way that takes into account relationships among concepts in the ontology as well as the collaborative evidence derived from the ontological profiles of similar users. Our approach to recommendation generation is an extension of standard user-based collaborative framework in which user similarities are computed based on their interest scores across ontology concepts, instead of their ratings on individual items. Our experimental results for collaborative recommendation, based on real ratings in the book domain, show significant improvement in prediction accuracy as well as coverage when compared to standard collaborative filtering.

2 Related Work

Widely used collaborative filtering methods can be divided into two main categories including Memory-based (user-based) and Model-based (item-based) algorithms [3, 4]. User-based techniques [5] generally model the user as a vector of item ratings and compare these vectors using a correlation or similarity measurement. Item-based algorithms [6] explore the relationships among items first, rather than the relationships between users, thus avoiding the bottleneck of having to search for neighbors among a large user population of potential neighbors.

Content-based filtering methods [7] have also been used in the context of recommending books and Web pages, where content descriptors are available. Rather than using simple feature vector models, our work differs from existing

approaches by taking advantage of the deeper semantic knowledge in an existing ontology for generating recommendations.

Many recommender systems suffer from the cold-start problem of handling new items or new users. Hybrid recommenders [8] combine semantic or content-knowledge with collaborative filtering to deal with this problem. Knowledge-based recommender systems use knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements [9]. Our work can be described as a knowledge-based collaborative hybrid.

The availability of large product taxonomies such as *Amazon.com* and *Open Directory Project* has allowed researchers to incorporate semantic information into recommender systems [10]. In order to address rating sparsity, Ziegler et al. [11] classify products by topics based on taxonomic information. Cho and Kim [12] have utilized a product taxonomy to overcome scalability issues. In [13], spreading activation techniques are used to find related concepts in the ontology given an initial set of concepts and corresponding initial activation values.

In our approach, the hierarchical structure of an underlying ontology is used explicitly and automatically in the learning and incremental updating of user profiles. There has been little work in the area of ontological user modeling and even less in the application of such models to Web personalization [14]. Our research follows the lead of other systems [15] that use ontologies to mediate information access, but these systems have generally not incorporated user modeling.

3 Augmenting Collaborative Recommendation

We take the goal of the recommender system to be the presentation of personalized recommendations for a particular target user. To accomplish this task, there are three broad categories of knowledge that may come into play: social, individual, and content knowledge [16]. Social knowledge covers what we know about the large community of users other than the target user, whereas individual knowledge refers to what we know about the target user. Content knowledge encapsulates domain knowledge about the items being recommended.

Recommender systems based on collaborative filtering utilize explicit or implicit ratings collected from a population of users. The *standard k-Nearest Neighbor (kNN)* algorithm operates by selecting the k most similar users to the target user, and formulates a prediction by combining the preferences of these users. Without the advantage of deeper domain knowledge, collaborative filtering models are limited in their ability to reason about the relationships between item features and about the underlying factors contributing to the final recommendations.

Our goal is to augment collaborative filtering by incorporating domain knowledge in the form of an ontology to enhance personalized recommendations. The ability to learn from user interaction is a critical factor for a good recommender system. In our ontology-based user model, the user behavior is represented not

as entries in a uniform vector, but as annotations to an ontology. We refer to this structure as the ontological user profile. In our previous work [17], ontological user profiles are utilized for Web search personalization based on individual users' interests. In this paper, we focus on a collaborative approach for ontology-based recommendation.

We maintain and update the ontological user profiles based on the user behavior and on-going interaction. For example, when Steve buys a book on programming in Python, the user profile associates this fact with the Python programming language concept via an annotation, and may activate other nearby concepts such as the Perl programming language. The system would not, for example, activate nodes associated with snakes or with British comedy troupes, although these have a syntactic relationship to the word "python". We utilize profile normalization so that the relative importance of concepts in the profile reflect the changing interests and varied information contexts of the user.

An ontological approach to user profiling has proven to be successful in addressing the *cold-start problem* in recommender systems where no initial information is available early on upon which to base recommendations [18]. Using ontologies as the basis of the profile allows the initial user behavior to be matched with existing concepts in the domain ontology and relationships between these concepts. Therefore, our approach strengthens the knowledge sources discussed above by providing an enriched representation of social and individual knowledge. Rather than developing the domain ontology ourselves, we rely on existing hierarchical taxonomies such as *Amazon.com's Book Taxonomy*.

Since collaborative filtering is based on the ratings of the neighbors who have similar preferences, it is very important to select the neighbors properly to improve the quality of the recommendations. Rather than computing user similarity on the whole set of items, we use a completely novel approach where the similarity among users is computed based on the users' level of interest for each concept. We compare the ontological user profiles for each user to form semantic neighborhoods. Because the number of items is often very large and so is the diversity among items, users who have similar preferences in one category may have totally different judgments on items of another kind [19]. Our approach allows us to take advantage of the deeper semantic knowledge in the domain ontology when selecting neighbors based on the interest level for each concept in the user profiles.

4 Ontology-Based Personalized Recommendation

For our purposes, an ontology is simply a hierarchy of topics, where the topics can be used to classify items being recommended. There is one main ontology on which all user profiles are based – we call this the *reference ontology*. An ontological user profile is a set of nodes from the reference ontology, each annotated with an *interest score*, which represent the degree of interest that the user has expressed in that topic or concept. Each node in the ontological user profile is a pair, $\langle C_j, IS(C_j)\rangle$, where C_j is a concept in the ontology and $IS(C_j)$ is the

interest score annotation for that concept. Whenever the system acquires new evidence about user interests, such as purchases, page views, or explicit ratings, the user profile is updated with new interest scores.

The hierarchical relationship among the concepts is taken into consideration for maintaining the ontological user profiles as we update the annotations for existing concepts. Each concept in the user profile is annotated with an *interest score* which has an initial value of one. As the user interacts with the system (i.e. rating a new book), the ontological user profile is updated and the annotations for existing concepts are modified. As a result, the profiles are maintained and updated incrementally based on the user's ongoing behavior.

4.1 Learning Profiles by Spreading Activation

We use *Spreading Activation* to incrementally update the *interest score* of the concepts in the user profiles. In our current implementation, the users' item based ratings are utilized to propagate interest scores in the user profiles. The process of learning an ontological user profile is depicted in Figure 1 using a portion of the ontology as an example.

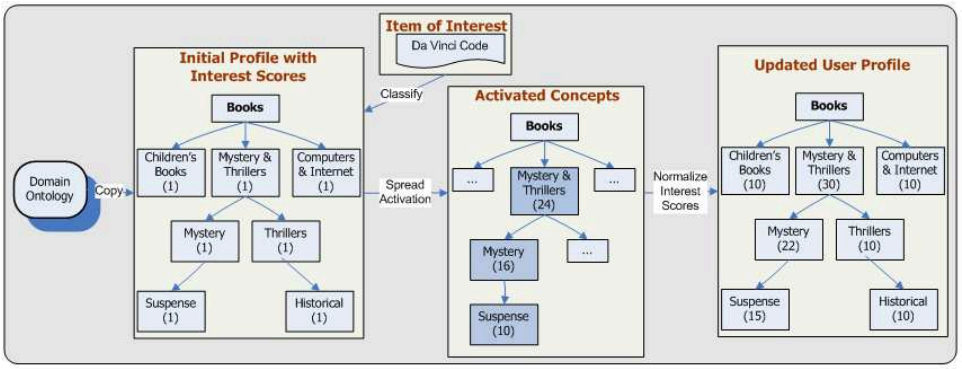


Fig. 1. Updating an Ontological User Profile

We use a very specific configuration of spreading activation, depicted in Algorithm 1, for the sole purpose of maintaining *interest scores* within a user profile. The ontological user profile is treated as the semantic network and the interest scores are updated based on activation values. The algorithm has an initial set of concepts from the ontological user profile. The main idea is to activate other concepts following a set of weighted relations during propagation and at the end obtain a set of concepts and their respective activations.

As any given concept propagates its activation to its neighbors, the weight of the relation between the origin concept and the destination concept plays an important role in the amount of activation that is passed through the network.

Algorithm 1: Spreading Activation Algorithm

Input: Ontological user profile with interest scores and an item of interest to the user, i
Output: Ontological user profile with updated interest scores

$CON = \{C_1, ..., C_n\}$, *user profile concepts with interest scores*
$IS(C_j)$ *and* $Activation(C_j)$, *interest score and activation value for concept* C_j

```
// Step 1: Spreading Activation
Initialize priorityQueue;
Set initial Activation of all concepts to 0;
foreach C_j ∈ CON do
    begin
        if (i ∈ C_j) then
            Activation(C_j) = IS(C_j);
            priorityQueue.Add(C_j);
        end
    end
end
while priorityQueue.Count > 0 do
    Sort priorityQueue; // activation values(descending)
     C_j = priorityQueue[0]; // first item(spreading concept)
     priorityQueue.Dequeue(C_j); // remove item
     if passRestrictions(C_j) then
        linkedConcepts = GetLinkedConcepts(C_j);
        foreach C_l in linkedConcepts do
            Activation(C_l)+ = Activation(C_j) * Weight(C_j, C_l);
            priorityQueue.Add(C_l);
        end
    end
end
// Step 2: Profile Normalization
foreach C_j ∈ CON do
    IS(C_j) = IS(C_j) + Activation(C_j);
    n = sqrt(n + (IS(C_j))^2); // square root of sum of squared interest scores
end
foreach C_j ∈ CON do
    IS(C_j) = (IS(C_j) * k)/n; // normalize to constant length, k
end
```

Thus, a one-time computation of the weights for the relations in the network is needed. Since the nodes are organized into a concept hierarchy derived from the domain ontology, we compute the weights for the relations between each concept and all of its subconcepts using a measure of containment. The weight, $Weight(C_j, C_s)$, of the relation for concept C_j and one of its subconcepts C_s is computed based on the number of items that are categorized under each concept. Once the weights are computed, we normalize the weights to ensure that the total sum of the weights of the relations between a concept and all of its subconcepts equals to one.

The algorithm is executed for each item of interest, such as a book. For each iteration of the algorithm, the initial activation value for each concept in the user profile is reset to zero. The concepts which contain the specific item are activated and the activation value, $Activation(C_j)$, for each activated concept C_j is set to the existing interest score, $IS(C_j)$, for that specific concept. If there is no interest information available for a given concept, then $IS(C_j)$ equals to one. The concept with the highest activation value gets removed from the queue after propagating its activation to its neighbors. The amount of activation that

is propagated to each neighbor is proportional to the weight of the relation. The neighboring concepts which are activated and are not currently in the priority queue are added to queue, which is then reordered. The process repeats itself until there are no further concepts to be processed. For a given spreading concept, we can ensure the algorithm processes each edge only once by iterating over the linked concepts only one time. The order of the iteration over the linked concepts does not affect the results of activation. The linked concepts that are activated are added to the existing priority queue, which is then sorted with respect to activation values.

After spreading activation, the interest scores in the profile are normalized. First the resulting activation values are added to the existing interest scores. The interest scores for all concepts are then treated as a vector, which is normalized to a unit length using a pre-defined constant, k, as the length of the vector. The effect of normalization is to prevent the interest scores from continuously escalating throughout the network. As the user expresses interests in one set of concepts, the scores for other concepts may decrease. For the long-term maintenance, the concepts in the ontological user profile are updated with the normalized interest scores.

4.2 Semantic Neighborhoods and Prediction Computation

In standard collaborative filtering, the similarity between the target user, u, and a neighbor, v, is calculated by the Pearson's correlation coefficient. Our alternative similarity metric uses the interest scores of these users' corresponding ontological profiles. First, we turn the ontological user profiles into flat vectors of interest scores over the space of concepts. We then compare the user profiles to figure out how distant each user's profile is from all other users' profiles. The distance between the target user, u, and a neighbor, v, is calculated by the Euclidean distance formula defined below:

$$distance_{u,v} = \sqrt{\sum_{j \in C} (IS(C_{j,u}) - IS(C_{j,v}))^2}$$

where C is the set of all concepts in the reference ontology, $IS(C_{j,u})$ and $IS(C_{j,v})$ are the interest scores for concept C_j for the target user u and neighbor v, respectively. Once all distances have been computed, we normalize the distance between the target user u and a neighbor v, then calculate a similarity value based on the inverse of the normalized distance.

The most similar k users are selected to generate the semantic neighborhoods. To further improve the quality of the neighborhoods, we use a concept-based filtering for the neighbors where a neighbor is included in the final prediction algorithm only if that neighbor's interest score for the specific concept is greater than their mean interest scores in their user profile. Our resulting semantic neighborhoods are not only based on similar users' explicit ratings for an item, but also based on the degree of interest those users have shown for the topic of a given item.

The ability to generate good recommendations relies heavily on the accurate prediction of a user's rating for an item they have not seen before. Our prediction algorithm uses a variation of Resnick's standard prediction formula [4] defined below:

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in V} sim_{u,v} * (r_{v,i} - \bar{r}_v)}{\sum_{v \in V} sim_{u,v}},$$

where $\bar{r}_u$ is the mean rating for the target user, V is the set of k similar users, $\bar{r}_v$ is the mean rating for a neighbor, $sim_{u,v}$ is the similarity described above.

We utilize the semantic evidence in the ontology for computing the mean rating for a user. For the mean rating of a target user or one of its neighbors, $\bar{r}_u$ and $\bar{r}_v$ respectively, we maintain two different values including the user's overall mean rating and user's concept-based mean rating. If an item belongs to only one concept, the user's concept-based mean rating is the user's average rating for all books that belong to that specific concept. In the case where a book belongs to multiple concepts, the concept-based mean rating becomes the user's average rating for all books that belong to these concepts. If the user's concept-based mean rating does not exist, the prediction formula uses the user's overall mean rating. Otherwise, the user's concept-based mean rating is used.

5 Experimental Evaluation

In the research community, the performance of a recommender system is mainly measured based on its accuracy with respect to predicting whether a user will like a certain item or not [20]. Our experimental evaluation focuses on comparing the quality of the recommendations based on our ontological approach versus standard collaborative filtering.

5.1 Experimental Data Sets and Metrics

Our data set consists of a subset of the book ratings that were collected by Ziegler in a 4-week crawl from the Book-Crossing community[11]. For each distinct ISBN, a unique identifier for the books in the dataset, we mined *Amazon.com's Book Taxonomy* and collected the category, title, URL, and editorial reviews for the specific book. Our resulting reference ontology includes 4,093 concepts and a total of 75,646 distinct books that are categorized under various concepts.

Only the explicit ratings, expressed on a scale from 1-10, are taken into account in our experiments. Our data set includes 72,582 book ratings belonging to those users with 20 or more ratings. The data set was converted into a user-item matrix that had 1,110 rows (i.e. users) and 27,489 columns (i.e. books). For evaluation purposes, we used 5-Fold cross-validation. For each fold, 80% of the book ratings were included in the *training set*, which was utilized to compute similarity among users. The remaining 20% of the book ratings were included

in the *test set*, which was used for predicting ratings. The advantage of *K-Fold cross-validation* is that all the examples in the dataset are eventually used for both training and testing.

To measure prediction accuracy, we rely on a commonly used metric *Mean Absolute Error (MAE)*, which measures the average absolute deviation between a predicted rating and the user's true rating: $MAE = \frac{\sum |p_{u,i} - r_{u,i}|}{N}$, where N is the total number of ratings over all users, $p_{u,i}$ is the predicted rating for user u on item i, and $r_{u,i}$ is the actual rating. The lower the *MAE*, the more accurately a recommender systems predicts user ratings. One main advantage of *MAE* is that it is a statistical metric which allows for testing the significance of a difference between the mean absolute errors of two systems. For our second type of evaluation, we generate a list of *Top-N recommendations* for each user. We compare the recommendations to the user's actual preferences and consider each match a hit. We use a *Hit Ratio* metric to compare our approach to standard collaborative filtering.

5.2 Experimental Methodology and Results

The first step in our experimental evaluation was to compute user-to-user similarity for the *standard kNN* algorithm using the Pearson's correlation based on the training data. Next, we used the books in the training set to generate ontological user profiles. Each user started out with an ontological user profile where all interest scores were initialized to one, this simulates a situation where no initial user interest information is available. For each book that was rated by a user, we performed our spreading activation algorithm to update interest scores in the ontological user profile for that specific user. In order to ensure the interest in the profiles is propagated based on strong positive evidence, only the items with ratings that were equal to or greater than the user's overall average rating were utilized for spreading activation. After an ontological user profile was created for each user based on their ratings, we utilized our semantic neighborhood generation approach explained above to compute the similarity among user profiles.

We calculated the *MAE* across the predicted ratings produced by each algorithm. For both the *standard kNN* and our ontological approach, the most similar k users were selected to compute the prediction for each item in the test set.

To generate a recommendation list for a specific user, we computed a predicted rating for all items that were rated by that user's neighbors, excluding the items that were rated by the user. With this type of an evaluation, the goal is to generate recommendations that the user has not seen before. The recommendation list was sorted in descending order with respect to the predicted rating for each item. Therefore, items with higher predicted ratings are included in the *Top-N recommendations*. We compared the recommendation list to the user's actual ratings for items in the test set.

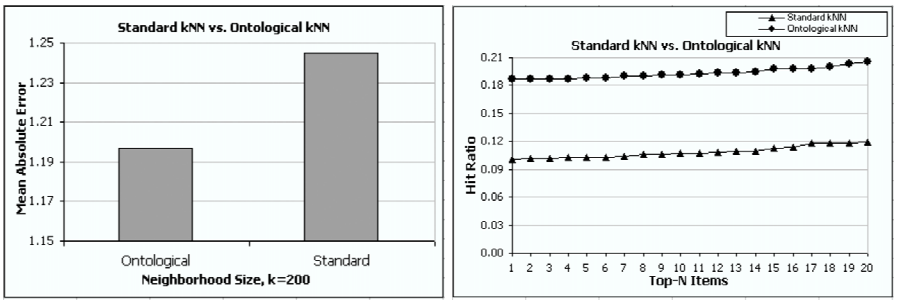
We ran our experiments for different values for the neighborhood size k, ranging from 20 to 200. For each value of k, the *MAE* was lower for predictions using

our ontological approach than the *MAE* across the predictions generated with the *standard kNN* algorithm. Our ontological approach also provides much better coverage, which is a measure for the percentage of items that a recommender system can provide predictions for. As depicted in Table 1, we computed three

Algorithm	Overall Ratings	Actual Ratings	Default Ratings	Coverage
Standard kNN	1.139	1.245	1.049	45.9%
Ontological kNN	1.112	1.197	1.025	50.6%

Table 1. Mean Absolute Error, k = 200 - Standard kNN vs. Ontological Approach

different *MAE* values for each algorithm using overall ratings, actual ratings, and default ratings. The *MAE* across actual ratings takes into account only those ratings where an actual predicted rating can be made based on the ratings of neighbors as opposed to the predicted ratings based on the user's default rating due to lack of ratings from neighbors.



(a) Mean Absolute Error (b) Hit Ratio

Fig. 2. Standard kNN vs. Ontological Approach, k = 200

Inspecting the *MAE* across the actual predicted ratings separately than the default ratings is important in order to effectively compare the different algorithms presented in this paper. In [21], the authors explore the importance of the influence of neighbors in collaborative filtering and present their finding on three commonly used, large-scale, real-world datasets including MovieLens, NetFlix, and BookCrossing. Due to the high sparsity of the Book-Crossing dataset, user-based collaborative filtering performs particularly poorly, with only 53% of neighborhood estimates actually contributing to better quality predictions than chance [21]. One additional advantage of our approach is that we are able to improve the predicted ratings based on the user's default rating since we use a concept-based mean as opposed to taking the user's average across all of the items rated by that user.

The comparative results with the *MAE* values across actual predicted ratings for $k = 200$ are depicted in Figure 2(a). The *MAE* values were confirmed to be significantly different using the *ANOVA* significance test with a 99% confidence interval, *p-Value = 6.9E-11*. Thus, we can confidently conclude that the prediction accuracy using our ontological approach is higher than the prediction accuracy of the *standard kNN* algorithm.

Next, we present our *Hit Ratio* results to compare *standard kNN* with our ontological approach in terms of *Top-N Recommendation*. The *Hit Ratio* is computed by determining whether a hit exists within the top N items in the list for each value of N, where $N = 1$ through $N = 20$. With this approach, the *Hit Ratio* is either 0 or 1 for each value of N for each user. We then take an average across all users in our data set. The recommendation lists for each user were sorted in descending order with respect to the predicted rating for each item. For each algorithm, the *Hit Ratio* results were based on the predicted ratings using a neighborhood size of $k = 200$. As depicted in Figure 2(b), the *Hit Ratio* for *ontological kNN* is significantly improved over *standard kNN*. These results further validate that our ontological approach performs better as a recommender system.

6 Conclusions and Outlook

We have presented our approach to collaborative recommendation that effectively incorporates semantic knowledge from ontologies with collaborative user preference information. Our approach not only outperforms traditional collaborative filtering in prediction accuracy but also offers improvements in coverage. Although accuracy metrics are important, in order to fully satisfy a user's recommendation needs, other measures such as diversity of recommendation lists and uniqueness of recommended items must be considered. In our future work, we plan to further evaluate the advantages of our ontological approach in terms of coverage, diversity, personalization, and cold-start performance.

References

1. Adomavicius, G., Tuzhilin, A.: Toward 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) 734–749
2. Schafer, J.B., Frankowski, D., Herlocker, J.L., Sen, S.: Collaborative filtering recommender systems. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York (2007)
3. Breese, J.S., Heckerman, D., Kadie, C.: Empirical analysis of predictive algorithms for collaborative filtering. In: Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence, San Francisco, CA (1998) 43–52
4. Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: Grouplens: An open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work, CSCW 1994, New York, NY (2004) 175–186

5. Herlocker, J.L., Konstan, J., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In: Proceedings of the 22nd ACM Conference on Research and Development in Information Retrieval, SIGIR 1999, Berkeley, CA (August 1999)
6. Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web, WWW '01, New York, NY (2001) 285–295
7. Mooney, R., Roy, L.: Content-based book recommending using learning for text categorization. In: Proceedings of the fifth ACM conference on Digital libraries, DL 2000, New York, NY (2000) 195–204
8. Burke, R.: Hybrid web recommender systems. In Brusilovsky, P., Kobsa, A., Nejdl, W., eds.: The Adaptive Web: Methods and Strategies of Web Personalization. Volume 4321 of Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg New York (2007)
9. Burke, R.: Knowledge-based recommender systems. In Kent, A., ed.: Encyclopedia of Library and Information Systems. Volume 69. Marcel Dekker (2000)
10. Middleton, S., Shadbolt, N., Roure, D.C.D.: Ontological user profiling in recommender systems. ACM Transactions on Information Systems **22**(1) (2004) 54–88
11. Ziegler, C., McNee, S., Konstan, J., Lausen, G.: Improving recommendation lists through topic diversification. In: Proceedings of the 14th international conference on World Wide Web, WWW 2005, Chiba, Japan (May 2005)
12. Cho, Y., Kim, J.: Application of web usage mining and product taxonomy to collaborative recommendations in e-commerce. Expert Systems with Applications **26**(2) (2004) 233–246
13. Rocha, C., Schwabe, D., Aragao, M.P.: A hybrid approach for searching in the semantic web. In: Proceedings of the 13th international conference on World Wide Web, WWW 2004, New York, NY, USA (2004) 374–383
14. Gauch, S., Speretta, M., Chandramouli, A., Micarelli, A.: User profiles for personalized information access. In: The adaptive web, LNCS 4321. (2007) 54–89
15. Gauch, S., Chaffee, J., Pretschner, A.: Ontology-based personalized search and browsing. Web Intelligence and Agent Systems **1**(3-4) (2003)
16. Burke, R., Ramezani, M.: Matching recommendation domains and technologies. In Kantor, P., Ricci, F., Rokach, L., Shapira, B., eds.: Handbook of Recommender Systems, to appear. Springer (2010)
17. Sieg, A., Mobasher, B., Burke, R.: Web search personalization with ontological user profiles. In: ACM Sixteenth Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal (November 2007)
18. Middleton, S., Shadbolt, N., Roure, D.D.: Capturing interest through inference and visualization: Ontological user profiling in recommender systems. In: Proceedings of the International Conference on Knowledge Capture, K-CAP 2003, Sanibel Island, Florida (October 2003) 62–69
19. Truong, K., Ishikawa, F., Honiden, S.: Improving accuracy of recommender system by item clustering. IEICE - Transactions on Information and Systems **E90-D**(9) (2007) 1363–1373
20. Herlocker, J.L., Konstan, J., Terveen, L., Riedl, J.: Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems **22**(1) (2004) 5–53
21. Rafter, R., O'Mahony, M., Hurley, N.J., Smyth, B.: What have the neighbours ever done for us? a collaborative filtering perspective. In: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization, UMAP 2009, Trento, Italy (2009) 355–360

Using Bayesian Networks To Infer Product Rankings From User Needs

Sven Radde and Burkhard Freitag

Institute for Information Systems and Software Technology
University of Passau, 94030 Passau, Germany
{sven.radde,burkhard.freitag}@uni-passau.de
http://www.ifis.uni-passau.de/

Abstract. Customers are commonly not able to provide preferences that are technical enough to be used in the internal algorithms of knowledge-based recommender systems. In this paper, we present an approach to use a Bayesian network to infer technical preferences from customer answers obtained through a conversational elicitation process. The inferred preferences can be used in conjunction with a variety of common database ranking technologies to generate product recommendations.

Key words: Bayesian inference, ranking databases

1 Introduction

When recommending complex products to customers, a knowledge-based recommender system has to base its reasoning on the mostly technical product properties that can be obtained using datasheets or similar sources of information. To this end, several well-known techniques exist, such as database ranking based on multi-attribute utility-theory or preference-based databases. However, as a prerequisite for applying these techniques, customers would need to be able to specify their wishes in technical terms. Experience reveals that this is far more than what can be expected from the average customer, in particular for complex technical product damains.

In this paper, we present a way to elicit preferences from customers by asking them "soft" questions about their needs and expectations. From their answers, preferences that serve as inputs to the afore-mentioned recommendation algorithms can be inferred by using a Bayesian network. We describe a utility-based approach to obtain product recommendations and provide a method to use the inferred knowledge as pareto-preferences.

The rest of this paper is organized as follows:

In section 2, we present a use case and describe the requirements arising from it. The process of inferring utility values for technical product attributes from customer answers is detailed in section 3. In section 4, we elaborate on how to make use of the inferred knowledge by employing a MAUT-based approach as well as other techniques such as pareto-based preferences. We review some related work in section 5 before concluding in section 6.

2 Use Case

Today's cellphones are complex technical devices, fulfilling a wide range of functions beyond the classic "make a phone call away from home" scenario, such as navigation aid, MP3 player, digital camera, web / messaging client and many more. Very few customers are able to make completely well-informed buying-decisions in this field without a significant need for consultation. The situation is made even worse by the frequently changing product domain. Retailers are forced to permanently release new products to expand their share in a rather saturated market. Also, the domain itself sees frequent technical innovations, creating new products and sometimes even new business models (think of, e.g., location-based services made possible by the recent integration of GPS receivers into many mobile phones). Therefore, even once-acquired domain knowledge ages rapidly, for customers and salespersons alike.

Cellphones differentiate themselves primarily through their *technical* properties, whereas customers' whishes commonly consist of "softer" *needs* that the desired phone is supposed to satisfy. Salespersons bridge this discrepancy in a natural way, translating the customers' wishes into appropriate technical requirements before recommending cellphones based on these requirements.

However, many current online-recommenders for mobile phones, despite being aimed at end-users, e.g., in web stores, cannot handle this abstraction. Commonly, many online applications are merely product configurators that require their users to specify technical constraints which are then used to restrict the set of available products accordingly. To bridge the gap between a customer's wishes and technically evaluable preferences, an electronic recommender system must be prepared to accept fuzzy user input and use an internal model to infer the technical criteria. At the same time, the inference mechanism must be flexible enough to be used in a dynamic dialogue environment, i.e., it cannot rely on a particular order answers are given in. Some questions may well remain unanswered throughout the dialogue, and for some questions the customers may change their opinion at a later point in the dialogue.

3 Eliciting Preferences

Our approach to infer technically useful information from customer answers is based on a Bayesian network (cf., e.g., chapter 14 in [12]) which is automatically generated from a metamodel-based description of the targeted market domain. Bayesian networks can be applied to model causal relationships between observations and their possible effects based on conditional probabilities. Once having evidence of (part of) the observations, a Bayes engine can infer the probabilities of the possible outcomes by probability propagation through the network. In the work reported on here, we use a Bayesian network to model the relationships of user needs (observations) and technical properties of products (outcomes).

Fig. 1 shows a UML diagram of a portion of the domain metamodel which is relevant to the process of eliciting preferences. The *Articles* in the targetted

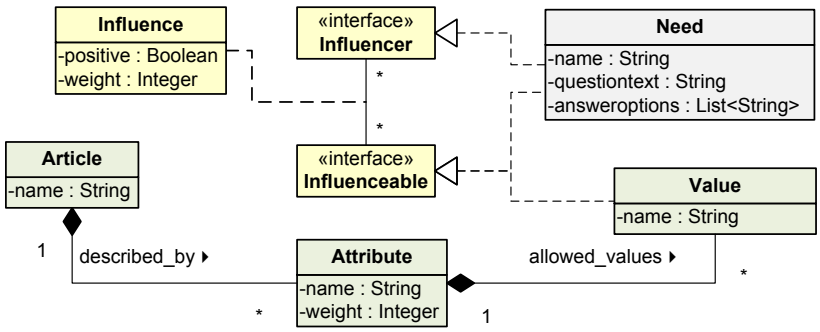


Fig. 1. UML diagram of the domain metamodel

domain are described by an explicit aggregation of their technical *Attributes*. To be able to distinguish the relative importances of *Attributes* during the recommendation process, they are assigned a numerical weight (cf. definition 4 for a detailed explanation). Also, each *Attribute* explicitly lists its allowed *Values*, defining the possible characteristics of the products in the target domain. Recommendations will then be obtained based on the modeled description of the technical properties of the target domain. The implied restriction to finite, discrete value ranges for *Attributes* has proven to be not a significant limitation, since, for our purpose, the value ranges of typically continuous *Attributes* (e.g., prices or sizes) can easily be classified into discrete ranges.

The dialogue used to obtain information from the customer is specified by modelling a number of *Needs*. The *Need* entity contains properties that are used to formulate a specific question that can be asked during the course of the elicitation process, including the possible answer options.

Connecting the answers about *Needs* with the desired technical product properties that are required to obtain recommendations is accomplished by modelling a number of *Influences* that represent the causal interdependencies between *Needs* and other *Needs*, as well as between *Needs* and attribute *Values*. *Influences* can be positive, i.e., have a strengthening effect, or not, i.e., have an attenuating effect. The numerical weight of an *Influence* denotes the "strength" of the *Influence* as we will detail further below.

Linking *Needs* with each other by means of an *Influence* is based on the notion that answers to questions regarding one *Need* may influence answers that the customer is likely to give when asked questions about other *Needs* in the model, as can be observed in example 1. *Influences* between *Needs* and *Values* model the influence an answer to a question has on the perceived utility of a product, as we will show in definition 1.

Note that the diagram in Fig. 1 displays a *metamodel*, which must be instantiated to a domain model with concrete *Needs*, *Attributes*, and *Influences*. Examples 1, 2, and 3 will illustrate this process for a simplified mobile communications domain.

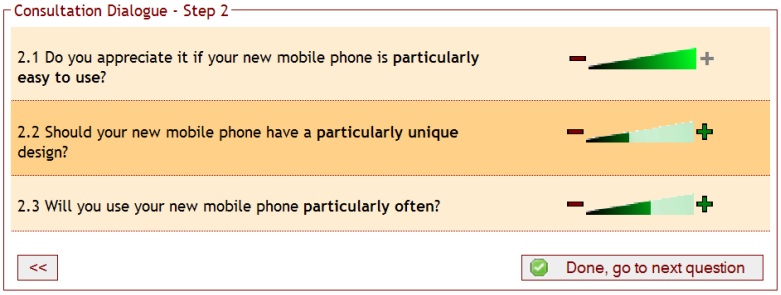


Fig. 2. Screenshot of the dialogue in our web application

Example 1 (Needs). Let us consider a simplified example from the mobile communications domain where the metamodel could have been instantiated by specifying the following exemplary *Needs*:

Need name	**Question text**
	"Will you use your new phone to...
`multimedia`	...play multimedia content?"
`business`	...use business functions?"
`music`	...listen to music?"
`internet`	...access the internet?"

It is immediately apparent that causal influences exist between some of these *Needs*, which can be represented using *Influences* (cf. Fig. 1), as we show below.

For many questions it turns out in practice that they should be written in a way so that they can be answered by customers on a Likert scale [9] (i.e., on a scale between "strongly agree" and "strongly disagree"). See Fig. 2 for a screenshot that illustrates how questions are displayed in a web application that implements the approach described in this paper.

Example 2 (Attributes and Values). A current real-life model instantiation for the mobile telecommunications domain comprises about 70 technical *Attributes* with about 500 possible attribute *Values*. In this paper, we focus on three exemplary attributes and their possible values: Does the cellphone have an integrated MP3 player, does it have broadband internet connectivity via UMTS, and what size has its internal memory?

Attribute	**Allowed values**	**weight**
`mp3`	`yes` / `no`	2
`umts`	`yes` / `no`	1
`memory`	`small` / `medium` / `large`	1

While `mp3` and `umts` are examples of boolean value ranges, `memory` illustrates a discretized value range where the actual amount of memory has been mapped to discrete values (using, e.g., <1GB / 1-4GB / >4GB as a classification rule).

Table 1. Exemplary Influences

from	**to**	**type**
business	internet	positive
business	Memory_small	negative
business	Memory_medium	positive
internet	umts_yes	positive
internet	umts_no	negative
multimedia	music	positive
multimedia	Memory_medium	negative
multimedia	Memory_large	positive
music	mp3_yes	positive
music	mp3_no	negative

Example 3 (Influences). Let our sample domain model contain *Influences* as shown in table 1 connecting the model elements of examples 1 and 2. For simplicity, assume further that all of them have an equal weight of 1.

Using *Influences*, arbitrary hierarchies of *Needs* may be specified, although a real-life evaluation revealed that the hierarchy remains rather flat in the considered use case. In fact, there are merely three tiers, i.e., one tier of *Needs* that influence other *Needs* on a middle tier which, in turn, influence the technical *Attributes* that form the bottom tier.

From an instance of the domain metamodel, a corresponding Bayesian network can be automatically generated. *Needs* and attribute *Values* are represented as random variables (i.e., nodes of the graph representing the network). *Need*-variables have outcomes that correspond to the possible answers of the corresponding questions in the elicitations dialogue, so that a customer's answers can easily be represented in the Bayes net by introducing evidence for the appropriate outcomes. On the other hand, each *Value* is represented by a random variable with the outcomes of "*true*" and "*false*" that model the likelihood that a certain *Value* fulfills the customer's *Needs*. Modelling *Values* in this way (and not, e.g., by representing one *Attribute* as a single variable with all *Values* as possible outcomes) is necessary to model the fact that *Values* may be satisfying independently of each other.

The *Influences* form the edges of the Bayesian network. Modeled causal dependencies between source nodes (*Influencers*) and a target node (*Influenceable*) are expressed by the entries in the conditional probability table (CPT) of the target node. The CPTs are constructed in such a way that, for positive *Influences*, if the customer agreed to the question corresponding to the source's *Need*, the likelihood that he/she will also agree to the target *Need* is increased, weighted relatively against other *Influences* by using the specified weight. If the target node corresponds to a *Value*, the probability for "*true*" of that particular *Value* is increased (i.e., it is more likely that products with this attribute value will be acceptable for the customer). Negative *Influences* have the opposite effects.

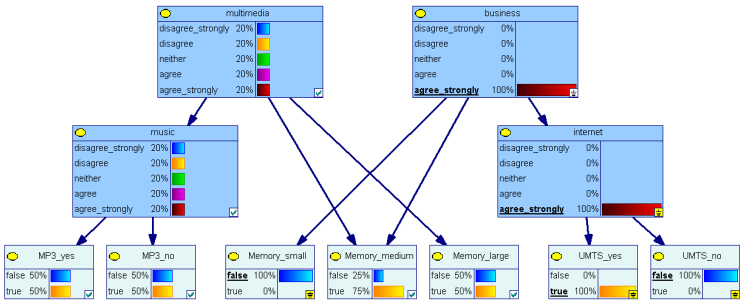


Fig. 3. Inference snapshot (modeled with GeNIe – http://dsl.sis.pitt.edu/)

For a detailed description of the overall generation method see [10]. Fig. 3 shows the Bayes net for the domain model as introduced in examples 1, 2, and 3. The main purpose of the Bayesian network is to derive utility estimations for attribute *Values* from which preferences will be derived. To this end, the notion of utility introduced below is based on the assumption that *Values* which are more likely to satisfy the customer's needs are also more useful (i.e., should have a greater utility).

Definition 1 (Utility of a Value). *We define the* utility u_{av} *of an attribute Value av as the posteriori-probability of the outcome "true" of the random variable* r_{av} *in the Bayesian network that corresponds to av:*

$$u_{av} := p(r_{av} = true \mid \ldots)$$

It immediately follows that all utility values are normalized between 0 *and* 1*. Also, the advantage of representing attribute Values as separate booleans becomes apparent: It is reasonable that more than one Value for an Attribute can get a* utility *of 1.0 (i.e., is useful with 100% probability). This situation can be taken into account in a Bayes net as described above.*

Example 4 (Utility). Consider an (early) point in the preference elicitation dialogue where the customer has given the answer "I strongly agree." to the question for the *Need* `business`. The answer is entered as evidence into the Bayesian network, leading to the posteriori probability-distributions shown in Fig. 3. Evaluating the *Influences* as described above, the elicitation algorithm infers that the customer will need to access the internet using the cellphone which makes UMTS connectivity a necessity, i.e., we have $u_{umts\_yes} = 1.0$ and $u_{umts\_no} = 0.0$

On the other hand, needing business functionality does not allow any conclusions towards multimedia support, which leads to ambivalent utilities for an MP3 player:

$$u_{mp3\_yes} = u_{mp3\_no} = 0.5$$

As for the memory capacity, the system reasons that a too small capacity is inacceptable in a business context, while a medium capacity is probably sufficient (depending on whether additional storage is required for multimedia content).

As we have seen, the primary goal of the inference engine in our context is to infer technical values from user needs according to the influence relations and given conditional probabilities. However, it can also process direct user input to allow for short-cuts by simply inserting a submitted value as evidence for the node of the Bayesian network that corresponds to the appropriate *Value*-entity. Thus, the preference elicitation dialogue is inherently adaptable to different levels of customer expertise. Further uses of the described inference approach, e.g., to control the dialogue flow by estimating the importance of unanswered Needs and to predict a customer's answers are presented in a broader context in [11].

4 Using Preferences

As shown above, the Bayesian network provides us with estimations about the utility of every possible attribute value in the product domain. To use these estimations to elicit preferences, we also need a notion of the relative importance of the attributes relative to each other. In our approach, the importance of an attribute depends on three factors:

(1) Those attributes customers show significant interest in should be regarded as more important than others. In our model, "significant interest" is derived from the fact that more *distinctive* predictions for the attribute values exist.

(2) The dialogue *situation* has influence on the importance of an attribute for the elicitation process. Attributes that are not connected to any already answered question, should not have any influence at all.

(3) A domain expert may assign a static numerical *weight* to each attribute (cf. Fig. 1). Marketing research shows that some attributes are inherently more important than others in a buying decision. Experiments indicate that it is sufficient to classify attributes into a small number of weight "classes", which is considered to be rather simple for suitably knowledgeable experts.

Definition 2 (Distinctiveness of an Attribute). *The* Distinctiveness d_a *of Attribute a is defined as the average of the distances of the utilities of all possible attribute values of a from the "indifferent" utility of* 0.5*, normalized to* [0..1]*:*

$$d_a := 2 * \frac{\sum_{v \in dom(a)}(|u_v - 0.5|)}{|dom(a)|}$$

Example 5 (Distinctiveness). To calculate the distinctiveness d_{memory} of the attribute "memory", assume that the following utilities have been inferred (cf. Fig. 3):

$$\begin{aligned} u_{memory\_small} &= 0.0 \\ u_{memory\_medium} &= 0.75 \\ u_{memory\_large} &= 0.5 \end{aligned}$$

$$d_{memory} = 2 * \frac{|0.0-0.5|+|0.75-0.5|+|0.5-0.5|}{3} = 0.5$$

The result fits our intuition: We are not yet sure about the customer's opinion regarding memory size at this point of the dialogue. Therefore, values derived for this attribute should not be taken too seriously.

Definition 3 (Situation Factor of an Attribute). *Let $Q_{answered}$ be the set of all questions already answered in the current dialogue. For an Attribute a let $P(a)$ denote the set of all Influence-ancestors of a, i.e., the Needs connected directly or transitively with a Value of a via an Influence-path in the network. The* Situation factor s_a *of a is defined as follows:*

$$s_a := \begin{cases} 0 \text{ if } \forall q \in Q_{answered} : q \notin P(a) \\ 1 \text{ otherwise} \end{cases}$$

Definition 4 (Importance of an Attribute). *Let a be an Attribute and w_a be the weight of a as specified in the particular model under consideration. Let d_a be the distinctiveness of a according to definition 2 and s_a be the situation factor of a (definition 3). The* Importance i_a *of a is defined as*

$$i_a := d_a * s_a * w_a$$

Example 6 (Importance). Extending example 5, we determine i_{memory} based on the following parameters:

$d_{memory} = 0.5$ (*example 5*)
$s_{memory} = 1.0$ (*memory is connected to the answered question, Fig. 3*)
$w_{memory} = 2.0$ (*taken from the domainmodel*)
$i_{memory} = 0.5 * 1.0 * 2.0 = 1.0$

Note that distinctiveness and situation factor depend on the preferences learned during the course of the elicitation dialogue. Therefore, they are updated after each dialogue step to account for newly acquired knowledge.

The approach to preference elicitation described here is based on Multi-Attribute Utility-Theory (MAUT – cf., e.g., [13]) which is used to combine the utility estimations of the various technical attributes into one global ordering of the whole product catalogue. To achieve this, we formulate a utility function for products which essentially computes a weighted sum over the utilities of all technical attributes.

The utility function can be formulated as a standard SQL query. For the sake of simplicity, we assume that all relevant data for an article is available in a single table with one column for each technical attribute (cf. Fig. 1) and an additional column containing the product's name. Every tuple in this table represents one concrete product (e.g., in the sample domain, one concrete cellphone). The following `SELECT` query computes a numerical `UTILITY` value for each tuple and orders the answer set accordingly:

```
SELECT   *, ($utilityfunction) AS UTILITY
FROM     cellphones
ORDER BY UTILITY DESC
```

`$utilityfunction` calculates the overall utility of a given cellphone by summing up the utility of each attribute value u_{value} (cf. definition 1), weighted by that attribute's importance $i_{attribute}$ (cf. definition 4). To this end, our implementation uses a set of `CASE-WHEN` clauses for each attribute.

Table 2. Exemplary Product Catalogue

Name	**mp3**	**memory**	**umts**
MobileA	yes	medium	no
MobileB	yes	large	no
MobileC	no	medium	yes
MobileD	no	small	no

Example 7 (Preference Order by MAUT). Assume that the current product catalogue contains the entries as shown in table 2. The utility and distinctiveness values are derived from the dialogue situation shown in Fig. 3:

$u_{umts\_yes} = 1.0$
$u_{umts\_no} = 0.0 \quad d_{umts} = 1.0$
$u_{memory\_small} = 0.0$
$u_{memory\_medium} = 0.75$
$u_{memory\_large} = 0.5 \quad d_{memory} = 0.5$
$u_{mp3\_yes} = 0.5$
$u_{mp3\_no} = 0.5 \quad d_{mp3} = 0.0$

For simplicity, assume situation factors of 1.0 for all attributes. For this constellation, the following ranking of catalogue entries according to the utility values computed as shown above results:
1) MobileC (Utility: `1.375`)
2) MobileA (Utility: `0.375`)
3) MobileB (Utility: `0.25`)
4) MobileD (Utility: `0.0`)

Since UMTS capability is the most important feature for our sample business customer, it is decisive in producing the utility-based rank ("MobileC" is the only UMTS-capable device in table 2). In contrast, support for MP3 does not play a role since the course of the dialogue did not yet allow any conclusions about the customer's wishes in this respect.

The structure of the query is known at the time of domain model design. Therefore, the query can be implemented as a stored procedure that can be called with the current importances and utility values as parameters. In our experiments, the actual query execution times were only a few millliseconds. It should be noted, however, that there are less than 1,000 different cellphones on the market today, leading to a rather small product catalogue and thus a small domain size. It is worth noting that, in general, the calculated utility for a product does not have an absolute meaning (i.e., a statement about *how* useful the product is, cannot be made). The value can only be interpreted in a relative way, i.e., a higher utility means greater usefulness and hence a higher rank.

The method described so far can also be used to provide the necessary inputs for more general approaches to rank query answers using boolean predicates, such as the one presented in [2]. To take full advantage of the more elaborate possibilites offered there, it would be necessary to extend our product modelling

by ways to express the potentially hierarchical structure of complex boolean conditions. Also, preferences for approaches based on pareto-optimality such as PreferenceSQL [7, 8] can be derived using a Bayesian network as described in this paper. In effect, the Bayesian network provides an ordering for all values of a technical attribute by interpreting the calculated utility values as the pareto-preferences serving as an input for PreferenceSQL.

PreferenceSQL supports "`LAYERED`" preferences (cf. [7]) for situations where a domain $dom(A)$ of an attribute can be partitioned into subsets that are ordered according to a "better than" relation. In our approach, all attribute values that have the same utility are grouped together in the same "layer", leading to a straight-forward application of the `LAYERED` preference constructor. Clustering techniques may be used to limit the number of subsets that are to be considered by interpreting some values as equally preferred, despite minimal differences in their numerial utility values.

Since the semantics of our approach relies on the notion that the customer is generally indifferent about attribute values with the same utility (i.e., all values with the same value are mutually substitutable), we annotate each `LAYERED` preference with the additional "`regular`" keyword (cf. section 4 in [7]). These preferences are combined using the pareto "`AND`" operator to form the complete PreferenceSQL query.

Example 8 (PreferenceSQL). We re-use the dialogue situation of example 7 to formulate a query in PreferenceSQL. Using the pattern described above leads to the following statement:

```
SELECT * FROM cellphones PREFERRING
umts   LAYERED (('yes'), ('no'), others) regular                  AND
memory LAYERED (('medium'), ('large'), ('small'), others) regular AND
mp3    LAYERED (('yes', 'no'), others) regular
```

Executing this statement against the product catalogue of table 2 yields "*MobileC*" as the query result, which is the pareto-optimal tuple of the relation and therefore the only result according to the "Best-Matches-Only" semantics of PreferenceSQL. Successively re-executing the query with added `WHERE`-clauses to exclude the already-retrieved tuples yields the following results which are ordered exactly as those obtained using our MAUT-approach in example 7 (NB: in general, the orderings defined by both approaches are not equivalent):

1) MobileC
2) MobileA (`WHERE name <> 'MobileC'`)
3) MobileB (`AND name <> 'MobileA'`)
4) MobileD (`AND name <> 'MobileB'`)

Generated in this straight-forward way, the PreferenceSQL query would consist of a very large number of pareto-preferences. To reduce the number of preferences, the importance of an attribute (cf. definition 4) may be exploited to limit the preferences to a fixed number or to consider only preferences for attributes having an inportance that exceeds a certain threshold.

5 Related Work

The knowledge-based elicitation approach described here is notably different from collaborative filtering methods (cf., e.g., [4, 6]) since it does not require item ratings. Assuming that a user will not interact with the system very frequently, we cannot rely on buying histories to build our model of the user. While eliciting explicit ratings may be acceptable in an online context, it seems not to be an adequate form of interaction between salespersons and customers. Hence, our user-model builds on preferences that can be elicited during the course of a natural sales dialogue.

Ardissono et al. [1] present a personalized recommender system for configurable products. Their approach involves preference elicitation techniques that employ reasoning about customer profiles to tailor the dialogue to a particular customer by providing explanations and smartly chosen default values wherever possible. The customer preferences learned this way are then used as constraints in the configuration problem at hand to generate the recommended product configuration, which might result in empty recommendations (i.e., the specified constraints are not satisfyable), requiring repair actions. Our approach does not directly exploit the elicited preferences as constraints but rather uses them as an input to ranking database queries which return a list of products ordered according to the customer's preferences.

In [3], the dialogue-oriented recommender suite CWAdvisor is presented. Their knowledge-base is similar to ours but it includes an explicit representation of the "recommender process definition", i.e. all possible dialogue paths in a tree-like structure. While obviously able to specify a fine-grained dialogue, the achievable level of detail is limited by the complexity of the dialogue specification. Our approach generates the (equally complex) dialogue specification from a much more compact model and is more flexible by incorporating mixed-initiative selection of questions, easy belief revision and adaptive personalization.

An approach similar to ours is presented in [5]. However, the utility estimations (the "value tree") of Jameson et al. do not seem to be built on an explicit model of the currently served customer but rather on the assumed properties of an average user of their system. Hence, the derived preferences are not personalized as strongly as in our approach. Also, as the value tree is a strictly hierarchical structure, it cannot capture the fact that a technical attribute may be influenced by more than one single need.

6 Conclusion

We presented an approach to inferring utility estimations usable for preference-based or ranking-based database queries, which is accomplished by using a Bayesian network that can be generated from a domain model. The approach described in this paper has been implemented in a joint project together with an industry partner. Their business experts designed a domain model comprising about 25 needs and more than 100 technical attributes. The model is considered

to adequately represent the marketing-relevant aspects of the mobile communications domain from the industry point of view. Although the effort for this first real-life instantiation of the model was significant, the task was regarded as feasible and the routine model maintenance has turned out to be inexpensive under operational conditions. Evaluations of the overall recommendation concept by marketing research experts have shown convincing results concerning the adequacy of the model and the acceptance of the overall approach.

Our current work focuses on conducting a thorough field evaluation of the system with a larger number of volunteers to obtain more statistical evidence about the quality of the derived preferences and related product rankings. Another field of ongoing research is the design of an explanation component which is able to explain the reasoning that led to the generated recommendations and to exploit user feedback about the recommendations to automatically adjust some parts of the domain model.

References

1. Ardissono, L., Felfernig, A., Friedrich, G., Goy, A., Jannach, D., Petrone, G., Schaefer, R., Zanker, M.: A Framework for the Development of personalized, distributed Web-Based Configuration Systems. AI Magazine 24, 93–110 (2003)
2. Beck, M., Freitag, B.: Weighted Boolean Conditions for Ranking. In: Proc. of the ICDE-08 Workshop on Ranking in Databases (DBRank'08) (2008)
3. Felfernig, A., Friedrich, G., Jannach, D., Zanker, M.: An Integrated environment for the Development of Knowledge-Based Recommender Applications. Intl. Journal of Electronic Commerce 11(2), 11–34 (2006)
4. Herlocker, J., Konstan, J., Terveen, L., Riedl, J.: Evaluating Collaborative Filtering Recommender Systems. ACM Transactions on Information Systems (TOIS) 22(1), 5–53 (2004)
5. Jameson, A., Schaefer, R., Simons, J., Weis, T.: Adaptive Provision of Evaluation-Oriented Information: Tasks and Techniques. In: Proc. of the 14th International Joint Conference on Artificial Intelligence (IJCAI) (1995)
6. Jin, R., Si, L., Zhang, C.: A Study of Mixture Models for Collaborative Filtering. Information Retrieval 9(3), 357–382 (2006)
7. Kießling, W.: Preference Queries with SV-Semantics. In: Proc. of the 11th International Conference on Management of Data (COMMAD). pp. 15–26. Computer Society of India (2005)
8. Kießling, W., Köstler, G.: Preference SQL – Design, Implementation, Experiences. In: Proc. of the 28th Intl. Conference on Very Large Data Bases (VLDB) (2002)
9. Likert, R.: A Technique for the Measurement of Attitudes. Archives of Psychology 22(140), 55ff (1932)
10. Radde, S., Kaiser, A., Freitag, B.: A Model-Based Customer Inference Engine. In: Proc. of the ECAI-08 Workshop on Recommender Systems (2008)
11. Radde, S., Zach, B., Freitag, B.: Designing a Metamodel-Based Recommender System. In: Proc. of the 10th International Conference on Electronic Commerce and the Web (EC-WEB) (2009)
12. Russel, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Prentice Hall International Editions (1995)
13. Schaefer, R.: Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests. In: Workshop on Adaptivity and User Modelling (2001)

Contact Recommendations from Aggegrated On-Line Activity

Abigail Gertner, Justin Richer, and Thomas Bartee

The MITRE Corporation
202 Burlington Road, Bedford, MA 01730
{gertner,jricher,tbartee}@mitre.org

Abstract. We describe a system for recommending people to contact based on similar interests and activities as part of a company-wide social networking site. Our contact recommendation service aggregates input from multiple on-line data sources and combines them using a Bayesian noisy-MAX function to generate a rating of the overall match between two users. The system is running as part of an experimental social networking site at MITRE. We present the results of a preliminary evaluation in which we compare the recommendations to existing friend relationships.

1 Introduction

The use of social networking platforms in Enterprise settings, including industry and government, is growing rapidly. One of the primary stated uses for these tools is to help workers connect with each other within their organizations. This is particularly attractive in the case of large corporations and government agencies whose organizational structure as well as geographic separation can make it very difficult to know who you should be talking to.

Modeled on the popular internet social networking sites, such as Facebook and LinkedIn, Enterprise social networking tools generally include contact lists, allowing users to connect with the people they know. These tools often also include a suite of social utilities such as blogs, wikis, bookmarks, tags and file sharing. By connecting with people via the contact list, a user can then keep track of their contacts' activities on the site, providing a *social filter* on the available information.

Social networking sites become more useful and more attractive the more connections their users make with each other. Contact recommendation is a feature that is intended to make it easier for users of a social network to create their online network. Facebook and LinkedIn both provide suggestions of people to connect to, primarily based on existing network connections in order to help users find people they know to connect to on the site.

In this paper we present a contact recommendation tool that is designed to help workers in a large distributed enterprise environment make connections with others who share similar interests or work activities. We generate contact recommendations by aggregating information about users from diverse data sources

within the company. The contact recommendations are implemented as a standalone web-service which is designed to be integrated with a social networking user interface. We show how the recommendations appear in our company's internal social networking tool and then discuss two preliminary evaluations of the recommendations.

2 Related Work

The literature on recommender systems has primarily focused on recommending items to users. Much of this work is based on *collaborative filtering* [1] – a technique that clusters people according to their item preferences and then recommends items that other similar users have liked.

There are several research projects that have looked at recommending people to each other. ReferralWeb [2] had the goal of finding existing chains of relationships between people by mining on-line documents such as co-authored papers and organizational charts. The *Do You Know?* system from IBM [4] also attempts to find people who are already known to the user in order to suggest that they be added to their social network. *Do You Know?* is implemented using SONAR [5], a social network aggregation tool that is probably the most similar to our own contact recommendation tool, in that it brings together evidence from multiple data sources to form its recommendations. The primary differences are the way we do the aggregation, and the fact that the IBM tool is attempting to identify existing social relationships, while we are primarily concerned with recommending people who are *not* known but possibly should be.

Terveen and McDonald [3] coined the term "social matching" to refer to systems that try to recommend and connect people each other. They outline the problem space in a series of claims, such as the need for explicit user models and the application to on-line social networks.

Another related area of research is the field of expertise finding [6]. This typically involves keyword searches for an expert who can answer a question or help solve a problem. In contrast, our contact recommender is looking for people who may be appropriate to form a longer term connection with based on common interests, not necessarily based on their expertise on a single topic.

3 Contact Recommendation Implementation

Our contact recommendation service is implemented as part of MITREverse, an experimental social networking system that is deployed inside the MITRE firewall. MITREverse is built on the Elgg open source social networking platform [7], which includes the basic social network features of friend lists, activity streams and message boards, as well as providing additional social tools such as groups, blogs, bookmarks and file sharing. MITREverse was deployed as part of a research project two years ago and has about 400 members in spite of never having been officially publicized or advertised within the company. Figure 1 shows a screenshot of a profile page on MITREverse.

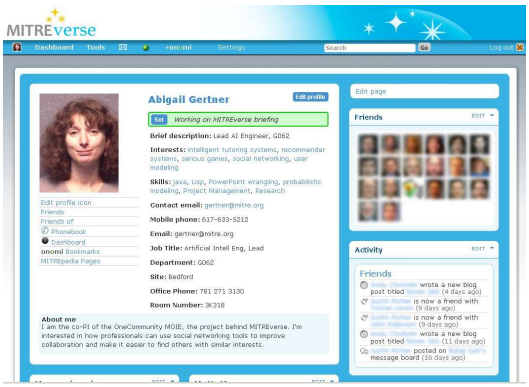


Fig. 1. A profile page on MITREverse

The original insight behind the MITREverse contact recommendation service was that by bringing together information about people from multiple places on the network, we could form a more accurate picture of what their interests are and what activities they are engaging in. As with many large companies, over the past five years or so MITRE has been making a number of social media tools available to its employees on our intranet. These include blogs, wikis, email lists, microblogging and social bookmarking tools. Some of these tools are official corporate supported offerings, and others are grassroots efforts started by individual employees. Many of these tools have built-in APIs that allow other programs to easily access and re-use their data. By aggregating the data from these different services together, the contact recommender creates a multi-dimensional view of what users have in common with each other.

Currently MITREverse uses seven data sources to compute the similarity scores: use of the same tags in onomi, our social bookmarking site, shared bookmarks in onomi, shared membership in internal email lists, co-editing of pages on our corporate-wide wiki, membership in groups on MITREverse, friend of a friend relationships, and use of the same tags on MITREverse. All of the data sources we use currently are inside the MITRE firewall and are accessed via public (to all MITRE users) APIs. We chose to avoid any privacy concerns by basing our recommendations only on data that would be accessible to anyone looking at the recommendations.

The contact recommender works by first generating a similarity score between each pair of users for each data source being considered. Since each data source may have a different type of user data, the data sources may use different algorithms for computing the similarity score. For instance, in the case of social bookmarking tags and MITREverse tags we use the cosine similarity of tag frequency vectors to compare two users' collections of tags. In the case of data sources in which there is a simple binary association between users and items, such as mailing list memberships, we use the Jaccard similarity coefficient (the size of the intersection divided by the size of the union) as the measure of

similarity between two users. If there is not enough information about a user for one of the data sources, no scores will be computed for that data source for user pairs involving that user.

After the similarity scores are generated for the individual data sources, an overall score is generated for the match between each pair of users based on the combination of those scores. The overall match is represented as a rating from zero to five, which is displayed on the user interface as a set of zero to five stars, as shown in Figure 2. The icons beneath the stars represent the data sources that were used to generate the recommendation, and clicking on the recommendation will take the user to a detailed explanation page for the recommendation.

Fig. 2. The display of a single recommendation

There are several possibile approaches to combining the individual data source scores into an aggregated rating. The most straightforward solution is to use the average score of the data sources, possibly weighted according to which data sources are considered more important. This is the approach taken by the *Do You Know?* system [4]. However, there are often cases where there is a strong match between two users on one or two of the data sources and a weak match on the rest. Using an average over all data sources would cause the overall score in these cases to be low, whereas we believed that people with a strong match in even one area would be likely to benefit from knowing each other. Therefore we decided to use a model that works more like an OR relationship – if *any* of the data sources scores is high, the resulting aggregated score will be high.

The model we are using is a causal probabalistic model called the Noisy-MAX [8], which is used in Bayesian networks to model a multi-valued variable whose value depends on the maximum value of its causal influences. Figure 3 shows a graphical representation of the Noisy-MAX model for aggregating similarity scores. The definition of the Noisy-MAX says that the inferred value of the node representing the outcome variable (in this case, the strength of the match in question, from zero to five) is determined by the maximum value produced independently by that node's inputs. The "noise" built into the probabilistic relationship means that the more inputs there are with a high similarity score, the more likely the overall similarity is to have a high value.

The Noisy-MAX takes advantage of the fact that the causal influences on the effect node are considered to be independent of each other. In this case, the causal influences are the individual data sources and the effect is the aggregated rating. Using this independence assumption, it is possible to define the relationship between the causes and the effect with much fewer parameters than

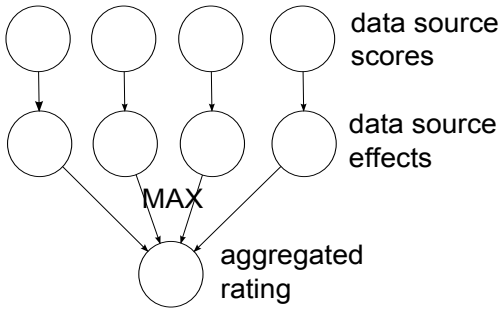


Fig. 3. The Noisy-MAX model for aggregating scores

a full conditional probability table. For each link from a data source to the combined rating, the parameters needed specify the probability that the combined rating will be equal to each possible value (0-5 stars), given the value of the data source and assuming that all other data sources are absent. Currently the parameters for the noisy-MAX are estimated subjectively but we are working to derive values from actual user judgments of matches between users.

We expect that it is the case that some data sources are more influential than others in determining a good match between users. Furthermore, each user may prioritize the various data sources differently. Therefore it is important to be able to weight the inputs in order to adjust their effect on the aggregated rating, as was done in [4] with the weighted average of the input scores. We have modified our implementation of the Noisy-MAX to include a weight parameter for each input data source, so that the influence of the individual data sources on the aggregated score can be adjusted according to the corresponding weights. Since different users may have different priorities for the data sources, we plan to allow them to adjust these weights via the user interface, although this feature is not yet implemented.

The contact recommender runs nightly to update its recommendations. With 396 users on MITREverse, the update takes about twelve minutes. However much of this time is spent downloading the full user data from the external (outside of MITREverse) data sources and so that time will not increase as MITREverse gains additional users.[1] The noisy-MAX computation to combine the data source scores currently takes about 1 minute and forty seconds. Extrapolating that value to apply it to all possible pairs of MITRE's approximately 7000 employees, the update would take 8.35 hours, so it will still be possible to update the recommendations once a day.

[1] Several of the web services we connect to include API calls that allow us to retrieve all user data in a single call. We do this even though many of the users are not current MITREverse users because we include non-MITREverse users in our recommendations in order to encourage current users to invite their recommended contacts onto the site.

4 Evaluation

As a preliminary evaluation of the accuracy of the contact recommendation ratings we have looked at the correspondence between the ratings and the actual friend relationships that currently exist in MITREverse. We hypothesized that for pairs of users who are connected to each other in the social network, the recommendation rating should be higher than for pairs of people who are not connected. This is not a perfect measure because, first, there may be people who know each other who have not yet connected on the site and, second, people may have friends on the site who they don't have much in common with. However, both of these disadvantages actually make it less likely that a difference would be detected. If we can see a difference in the ratings between the friend pairs and the non-friend pairs, it would give us an initial confirmation that our recommendations are doing the right thing.

There are 396 total user accounts on the MITREverse site, making 156,420 possible friend relationships (friend relationships in MITREverse are unidirectional, so a relationship of person A to person B is treated separately from a relationship of person B to person A). Out of these possible friend relationships there are 1,752 actual friend relationships in the site. The contact recommender found enough information in at least one data source to generate recommendations for 29,609 of the possible friend relationships. 1316 of these recommendations were for pairs of users who are already connected as friends, and the remaining 28293 are for pairs who are not yet connected. Table 1 summarizes the number of recommendations generated for existing friend and non-friend pairs.

Table 1. Total number of recommendations generated for friend and non-friend pairs

	Friends	Not Friends
Recommendation	1316	28293
No Recommendation	436	126374

We then compared the recommendations that were generated for the friend pairs with the recommendations that were generated for the non-friend pairs. The mean recommendation rating (taken from the 1-5 star ratings) for the friend relationships is 1.94, while the mean for the non-friends is 1.38. While this is a small difference, it is encouraging given our caveats about the friend relationships not being a completely reliable correlate of match strength. Figure 4 shows the percentage of the friend and non-friend pairs that were given each of the five possible ratings by the contact recommender. The non-friend pairs are 1.7 times as likely to be given a rating of one star as compared to the friend pairs. Conversely, the friend pairs are 5.15 times as likely to get a four star rating and 3.73 times as likely to get a five star rating.

To evaluate the use of the noisy-MAX function for combining the data source inputs, we generated recommendations using an alternative method of simply

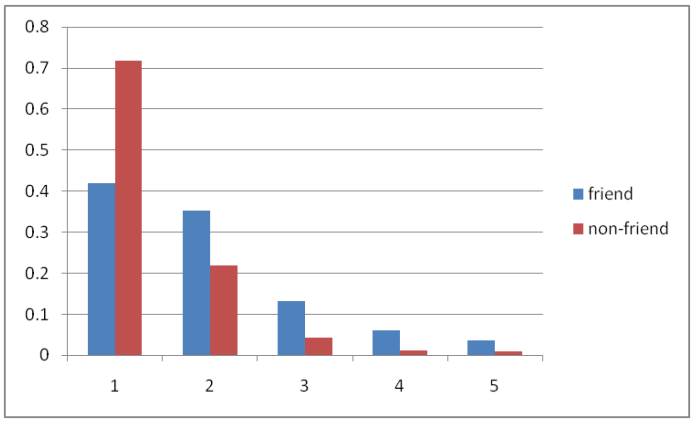


Fig. 4. Percentage of friend and non-friend user pairs assigned each rating, from one to five stars, using the noisy-MAX function

taking the average of the input scores, which is the approach taken in SONAR [5]. Figure 5 shows the percentage of pairs associated with one to five star ratings using this method. In this case, since we are not using the max function, both the friend and non-friend pairs are more likely to be given a rating of one star, and there are very few four or five star ratings, even for the friend relationships. Based on these results, it appears that the Noisy-MAX model does a better job of assigning a high rating to people who are actually friends than the straight average does.

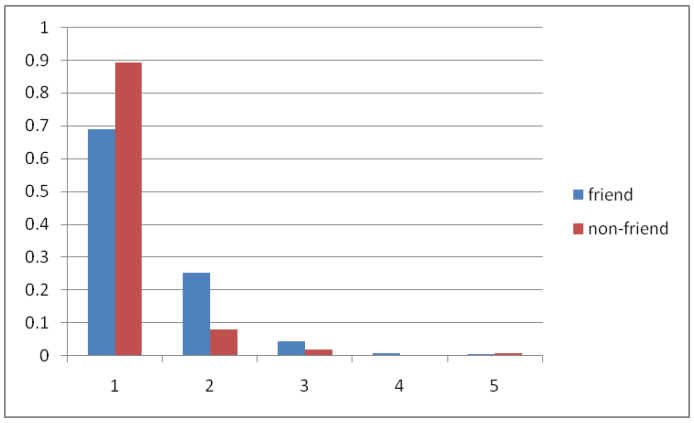


Fig. 5. Percentage of friend and non-friend user pairs assigned each rating, from one to five stars, using the average of data source inputs

4.1 Human Evaluation

An additional evaluation of the contact recommendations is underway, in which we are asking people to judge the similarity of pairs of people based on the same types of inputs that the recommender uses, to get a better understanding of what algorithms people use to make those decisions. This will help us to evaluate the use of the Noisy-MAX model as well as to determine the optimal parameter values. Initial results indicate that many people do consider the maximum of the individual data source inputs to be important in determining the final rating.

In this study we are also asking subjects to weight the individual data sources according to their importance to the overall recommendation. Eight out of the ten subjects so far have said that they consider the differences between data sources to be important or very important when computing the overall rating, but their ordered rankings of the data sources are all very different from each other. This supports our belief that it will be important to allow users to adjust the data source weights themselves to deliver optimal recommendations.

5 Discussion and Future Work

We have described a contact recommendation tool that looks at data available about users based on their on-line activities and uses that information to generate recommendations for other people with similar interests. This capability will soon be deployed company wide on a new social networking site called Handshake.

Our initial evaluation of the contact recommendations based on existing friend relationships shows that the ratings are at least able to distinguish between people who are connected to each other in the social network and those who aren't. It also suggests that the noisy-MAX model provides an advantage over taking the average of the inputs because it leads to higher ratings in general, especially for the existing friend pairs. However, as we noted earlier, we are primarily interested in the ability of the recommender to identify relevant contacts that are *not* already known to the user. Therefore we are planning a followup evaluation in which we will ask users to judge the actual recommendations that the system generates for them personally. In that case, we will be able to see whether the system is able to recommend novel connections that users would be likely to follow up on.

References

1. Herlocker, J. L., Konstan, J. A., Borchers, A., Riedl, J.: An Algorithmic Framework for Performing Collaborative Filtering. In: Proceedings of Research and Development in Information Retrieval, ACM, New York (1999)
2. Kautz, H., Selman, B., Shah, M.: Referral Web: combining social networks and collaborative filtering, Communications of the ACM 40(3), pp 63–65 (1997)

3. Terveen, L. and McDonald, D. W.: Social Matching: A Framework and Research Agenda. ACM Transactions on Computer-Human Interaction, 12(3), pp. 401–434, (2007)
4. Guy, I., Ronen, I., Wilcox, E.: Do You Know? Recommending People to Invite into Your Social Network, In: Proceedings of the 13th international conference on Intelligent user interfaces, pp. 77–86. ACM, New York (2009)
5. Guy, I., Jacovi, M., Shahar, E., Meshulam, N., Soroka, V., Farrell, S.: Harvesting with SONAR: the value of aggregating social network information. In: Proceedings of the twenty-sixth annual SIGCHI conference on human factors in computing systems (CHI '08), pp. 1017–1026. ACM, New York (2008)
6. Maybury, M., D'Amore, R., House, D.: Automating Expert Finding. International Journal of Technology Research Management. 43(6): 12-15 (2000)
7. Elgg open source social networking platform, `http://www.elgg.org/`
8. Díez, F., Druzdzel, M.: Canonical probabilistic models for knowledge engineering, Technical Report CISIAD-06-01. UNED, Madrid, (2006)

Reciprocal Recommenders

Luiz Pizzato, Tomek Rej, Thomas Chung,
Kalina Yacef, Irena Koprinska, and Judy Kay

School of Information Technologies
University of Sydney, NSW 2006, Australia
{forename.surname}@sydney.edu.au

Abstract. This paper introduces *Reciprocal Recommenders,* an important class of personalised recommender systems that has received little attention until now. The applications of Reciprocal Recommenders include online systems that help users to find a job, a mentor, a business partner or even a date. The contributions of this paper are the definition of this class of recommendation system, the identification of the particular personalisation challenges for them, the proposition of some promising techniques to address these challenges. We illustrate these concepts with a case study in online dating.

1 Introduction

There has been considerable research on recommenders and many deployed systems, in domains such as books (Amazon.com), pharmacy products (Drugstore.com), online auctions and seller reputation modelling (eBay) [11]. The dominant model for such recommenders is to provide a *user* with recommendations of *items* likely to be of interest to the user.

In social matching, where people are recommended to each other, the quality of a match is determined by all parties involved in the match. A match is successful only if everyone's preferences are satisfied. In terms of recommender systems, this motivates the concept of the *reciprocal recommender*, where the *user* and the *item* both have preferences that are considered when making a recommendation. This is in contrast with traditional recommenders which consider only the preferences of the user.

The traditional recommendation process is based on four important classes of models:

1. *explicit* user models or profile information about the user, such as their personal attributes like age, gender and educational level;
2. *explicit* models of the items, such as genre, director and actors for a movie recommender;
3. *explicit* user models of preferences in the domain, such as movie preferences in a movie recommender;
4. *implicit* user models based on their actions, for example purchases and time spent gaining more information about particular items.

Table 1. Difference between traditional recommenders and reciprocal recommenders

Traditional recommender	Reciprocal recommender
User receives recommendations and is sole decider about their use/purchase.	User is aware that success depends on the agreement of the other party involved.
Items are typically abundant, and even if not, there is no need to limit the number of users recommended an *item*.	Items have very limited availability. Items are represented by other users, or, as in online dating, *item*s are other users.
A successful transaction is defined by the user who was given the recommendation.	A successful transaction is defined by both the user given the recommendations and the recommendation *item* itself.
Users and products might constantly re-occur in the system, making easier to track preferences.	Users and products might only occur once, and might never appear again after a successful transaction. Therefore, the cold-start problem is significant in this domain.

Notably, in this large body of recommender research work, there is some symmetry between the explicit models of the *users* and the *items* (classes 1 and 2 above). However, there is no such symmetry for the explicit model of the user's preferences (Class 3) and the implicit model of preferences (Class 4).

In this paper, we will use the term *item* to refer to object being recommended, even when both the *user* and the *item* models may represent people. This allows us to compare the reciprocal recommender with conventional recommenders. For reciprocal recommender systems, we can make use of additional models:

5. *explicit* models of the *item*'s preferences in the domain;
6. *implicit* models of the *item*s based upon their activity.

Many domains will benefit from a reciprocal recommender. Consider the workflow of the expert recommender system in [10] where expert-seekers contact experts after receiving recommendations. The experts are passive in this system, and may only choose to reject a seeker after being contacted. It is clearly desirable to reduce the number of rejections from experts, as it costs time and effort for the seeker to browse for and contact experts. A reciprocal recommender would help by not only finding the right expert for the job, but also an expert who is likely to accept the job. In contrast, improving the immediate satisfaction of the users by providing them with people they like, might not reflect the final satisfaction of the user if these relationships are not mutual.

The reciprocal recommender is also useful in areas apart from expert recommendation. A job recommender needs to match the qualification of a candidate to the requirements of a position, but should also consider the likelihood of a candidate accepting a job [6]. A student/tutor recommendation needs to consider both the student and tutor's needs, skills and previous experiences [14]. For an online dating website, successful recommendations require that both users be interested in one another. Table 1 highlights some other differences between traditional and reciprocal recommenders.

Because reciprocal recommenders rely on a two-sided expression of interest between two different types of users, it presents some issues raised by Terveen and McDonald [13], such as privacy, trust, relation and interpersonal attraction. However, the authors have defined a research agenda centred on the computer

human interaction issues for social matching, which may or may not not have a reciprocal facet. In this paper, we define the reciprocal recommendation which, although mostly consisting of interactions between people, does not necessarily imply social matching.

In Section 3, we present a definition of reciprocal recommendation which can be used to guide future research in the area. Section 2 presents a review of existing systems and techniques that are used in the areas that require reciprocal recommenders. Although a considerable amount of work has been done in related areas, reciprocal recommenders are still much underdeveloped. Section 4 highlights some approaches that can be taken to address the problem of recommendation in these domains. In Section 5 we present a case study using online dating as one domain that requires this type of recommendation. Section 6 presents the concluding remarks.

2 Related work

The field of recommender systems is well established, with a large volume of literature describing different techniques to predict those items that are appealing to the users. A common criteria used to distinguish between recommenders is the technique used to generate the recommendations itself (i.e. content-based techniques, collaborative-filtering, or hybrid techniques). Although much work has been done in the areas of recommender system that could benefit from reciprocation, very few highlight the need for reciprocity.

The work of Malinowski et al. [6] builds two recommender systems for an employment website: one recommender that finds the best jobs for a person seeking a job, and one recommender that suggests the best people for a certain job. Malinowski et al. point out that it is important to combine both approaches to match the interests of both job seekers and employers. The authors describe different ways of integrating both recommenders, highlighting the fact that a one-to-one (job-seeker to job) Pareto-optimal solution would be desired.[1] Addressing the computational cost of working with a large dataset, the authors also proposes a quicker solution that takes into account a single stakeholder for which the recommendations are maximized..

Another work that shows some level of reciprocity is described by Vassileva et al. [14] in the mentoring systems iHelp. iHelp uses a multi-agent architecture to facilitate the search for mentors by students. In iHelp, agents have a model of the knowledge of each user and when students ask for assistance, they are capable of finding other users who are willing to help and whose knowledge is comprehensive in the required topic of learning. The reciprocity of iHelp comes from the fact that the user model of the student and all possible mentors are analysed before a match is selected. The knowledge deficiencies of the student

[1] A Pareto-optimal solution in the employment website domain is a combination of matches between jobs and job seekers such that no single swap between a pair of job seekers improves or maintains the satisfaction of every single person involved.

are found and paired with the knowledge strengths of possible mentors, as well as with the willingness of users to become mentors.

In Brožovský and Petříček [2], collaborative filtering is used to predict the ratings that users will give to other users when presented with their photo. This task is not necessarily reciprocal as it is mostly used by users who want to know how other people rate their appearance. Despite this, Brožovský and Petříček discuss the need for reciprocal matching algorithms as finding that "A likes B" does not imply that "B likes A".

Although reciprocity is an important issue for intrinsically reciprocal tasks such as friend recommendations on social networks and date recommendations on online dating systems, many works such as [16, 5] do not mention the need for reciprocity. These works seem to focus on the task of satisfying the immediate need of the user at hand. However, improving the immediate satisfaction of the users by providing them with people they like or believe to be their friends, might not reflect the final satisfaction of the user if these relationships are not mutual.

Work on referral systems in social networks such as [17] could also benefit from reciprocity; in particular when agreement between users (or their agents) is required. One such task is the business partner identification in social networks [15].

Little is know about the users' preference when they are new to the system. The lack of knowledge about the user restricts the power of the system to create recommendations. Several solutions were proposed to solve this problem, which is commonly referred to as the cold-start problem. Park and Chu [9] define four groups of recommendations: (1) existing items to existing users; (2) existing items to new user; (3) new items to existing users; (4) new items to new users. The cold-start problem is relevant for all groups except (1). For group (2) recommending popular items is a good baseline, while for group (3) an approach that takes into account the content of the items is needed. According to the authors, group (4) is a hard case that needs a "random" strategy. Park and Chu compared several strategies to generate the recommendations for the cold start cases and found that their method, pairwise preference regression, outperforms other known methods such as random, most popular, segmented most popular, and Vibes Affinity [7].

In contrast to the cold-start problem, controlling overspecialisation is particularly important for reciprocal recommenders in domains such as online dating, where allowing variability and a wide spread of recommendations is important. In online dating some users receive lots of attention, while at the same time many other users are being neglected. It is important for a recommender in this domain not to overload the popular users and to allow neglected users to be present in the other people's recommendations.

One strategy to ensure a balanced distribution of recommendation among users of different popularity is to recommend users among these groups. For instance popular users would be recommended to other popular users, while less popular users will be recommended to users with similar popularity scores. This

could minimize the effects described in [4], which demonstrated the tendency of collaborative filtering to recommend popular items even when the starting items are not popular.

The work of Abbassi et al. [1] deals with overspecialisation by finding regions in the item space that are relatively unexplored by a user. Similarly, Onuma et al. [8] applies a graph-based method to recommend items that are not centred in the user's current interest area, but are borderline between distinct areas of interest. This ensures novelty, and provides certain variety in the recommendations, that is seen favourably by users [12].

3 Reciprocal Recommender

A recommender $R1$ is a system that, when given a user u, recommends a list of items I such that the degree of preference between a user and every item in I is larger than the degree of preference between the same user and every items not in I. This is shown in Equation (1).

$$R1(u) = \{i : P1(u,i) > P1(u,j), \forall i \in I, \forall j \notin I\} \tag{1}$$

where $P1$ represents how much a user prefers an item.

Because a reciprocal recommender needs to consider the degree of preference for the items in I we can build a recommender $R2$ that gives the best users U for an item i, such that the degree of preference of an item i for every user in U is larger than every user not in U (see Equation 2).

$$R2(i) = \{u : P2(i,u) > P2(i,v), \forall u \in U, \forall v \notin U\} \tag{2}$$

where $P2$ represents how much a user is preferred by an item (normally represented by another user).

Therefore, the reciprocal recommender RR for a user u is a set of items I (subset of $R1(u)$) such that u is in the list of recommendations $R2(i)$ for all items i in I (see Equation 3).

$$RR(u) = \{i : i \in R1(u) \text{ and } u \in R2(i)\} \tag{3}$$

3.1 Combining recommenders

To obtain a single list of recommendations from a reciprocal recommender, we need to combine $R1$ and $R2$. Such a combination is necessary if the recommendations are to take into consideration the preferences of both user and item. Depending on the domain, we may choose different methods of combination that assign different weights or meanings to each of $R1$ and $R2$. Using the terminology of Burke [3], we apply the Cascade, Weighted and Switched methods of combination in our discussion below.

If both $R1$ and $R2$ produce unranked sets of users/items as described above, we may combining their output by filtering out the items in $R1$ that are not

reciprocated in $R2$ (Equation 3). This method treats $R1$ and $R2$ equally and is simple to compute.

If we assign a numeric score to the recommendations in $R1$ and $R2$ using $P1$ and $P2$, we may produce a combined ranked output PRR by calculating a weighted sum of the scores for each user/item pair (Equation 4). Using the flexibility in the choice of weights, it is possible to give focus to either the user's preference or the item's preferences. This customisability is useful in many cases. For example, if the user only wants items that are highly tailored to his/her needs and does not mind having a few unsuccessful interactions, we can give a higher weight to the user's recommendations.

$$PRR(u,i) = w_1 P1(u,i) + w_2 P2(i,u) \tag{4}$$

On the other extreme, if we need to recommend for a (possibly new) user for which we have no preference information about, we may choose to entirely rely on the item's preference for users to give the user items which will like the user. This can be used to mitigate the cold start problem.

4 Approaches to reciprocal recommendation

Recommender systems approaches are normally divided into two main classes: content-based and collaborative filtering. Content-based recommender systems take into account the content of the items that have been used by a user in order to find the likes and dislikes of the user and by using these preferences the system can find new and unknown items to present to the user. Conversely, collaborative filtering does not take into account the content of the items, but instead analyses their usage patterns. For instance, if a group of items used by a user A was also used by other users B and C, an item used by B and C and not used by A is a potential good recommendation to present to A.

When both users and items are actively engaged in the search of each other, then it would be possible to create a reciprocal recommender using any technique and finding the overlapping pair of users and items. For instance, in a job search scenario where one side is looking for job positions and the employee is looking for suitable candidates, a system can be built that finds all positions which will consider a particular candidate suitable. The same logic can be applied for the employer: a system can find all candidates who consider the position attractive to them. A system that is based on the overlapping recommendation is described by [6].

However, when one side of the reciprocal recommender is not actively engaged with the search, group generalisations may become necessary. For instance, if a job post is advertised but the advertiser company does not actively search for employees in the website database, then a system can generalise the job preferences by using previous similar positions by the same advertiser or company, or even previous similar positions by any company. This is one way of dealing with the cold start problem. Generalisations can create more data and therefore

minimise the problems associated with lack of data. However, it is unclear when these types of generalisations can be made.

Because similar people[2] might not act similarly, generalisations are harder to obtain when users are acting as individuals. Nevertheless, in many cases, building stereotypes for people is required, because without it there might not be any indication of what users might like. The lack of indication of preferences can also arise from the specific task at hand, which might require users or items to have one and only one successful interaction. For example, when someone finds a job using a job search website, it is likely that this person will stop using the website and the job position will be closed. For this type of task, scaling down the success requirements and finding intermediate levels of interest are required. For instance, the success of a job search website might not be defined as a job position being fulfilled, but rather the engagement of the website's users such that candidates submit applications and positions receive applications from candidates. In this way, success is a less strict criteria, and preferences can be more easily defined.

Collaborative filtering has been shown to work well for non-reciprocal recommenders, and can be easily applied to a reciprocal setting. However, in situations when generalisations are required by using previous positions and transactions, the set of users who obtained successful interactions might not be currently available. On the other hand, content-based techniques are better able to generalise preferences that were expressed previously by employers in different posts as these techniques are normally not bound by each individual candidate, but by the attributes of these users. In this way, a hybrid approach for a reciprocal job recommender could simply consist of two recommenders: one using collaborative filtering to recommend job positions to a candidate and one content-based to recommend candidates for a job position based on previous transactions between similar job positions and candidates.

Traditional recommenders can also be used to generate recommendations for reciprocal tasks. All successful and unsuccessful reciprocal transactions are used to train the recommender, which will recommend future items which are likely to reciprocate a transaction. However, this is only possible for systems that contain recurrent or long term users who can and will perform multiple successful transactions. For those systems where users are short-term and their expectations are to find a life-long partner or a life-long job career, the use of a combined, reciprocal recommender system is preferable.

Independently of the technique used, the combined reciprocal recommender must account for the lack of information about its users and items. The way of handling the cold-start problem is likely to be one of the major factors influencing the performance of such a system.

Figure 1 illustrates a reciprocal candidate-employer recommender that uses a recommender $R1$ for the candidate and a recommender $R2$ for the employer.

[2] Assuming we have a clear definition of similarity for people. However the concept of personal similarity is a whole problem on its own, which we will not address in this paper.

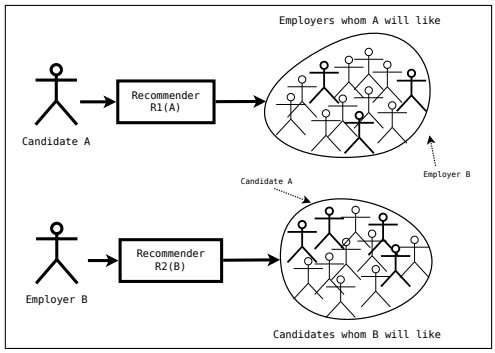


Fig. 1. Example of a reciprocal candidate-employer recommender. Recommendations drawn with thicker lines represent those recommendations which are reciprocal (i.e. Employer B appears in the list of recommended employers for Candidate A and Candidate A appears in the list of recommendations for Employer B).

The final recommendation for a given candidate A is a ranked list of employers (or positions) where the top of the list is populated by those employers whose recommenders suggest the current candidate as someone they might like (e.g. Employer B). Likewise, the final recommendation for a given employer B is a ranked list of candidates where the first candidates are those who would like to be employed by the company (e.g. Candidate A).

Malinowski et al. [6] suggest seeking a Pareto-optimal solution to incorporate the needs of both user and item in a bipartite matching problem. However, we believe that for a reciprocal recommender, one must not seek a Pareto-optimal solution in domains where the recommender accuracy is low, as such a solution assumes that recommendations are mostly successful. For example, in a job recommender, a Pareto-optimal solution may lead us to recommend the "second best" jobs to a person, because we think there are better candidates who will take the job. This decision is not sensible unless we are certain that our predictions are very accurate.

5 Case study - Online dating

Online dating is one of the areas where a reciprocal recommender is very important. In the online dating domain, the item being recommended to a user is another user who also has the same goal when using the system: to find a date. Any recommendation given in a online dating scenario needs to be reciprocal and must take into account the needs of both users being recommended to each other. Otherwise, if a recommendation is given with only one of the users in mind, these “good” non-reciprocal recommendations will be short lived because user interactions resulting from these recommendations are likely not to develop further.

Table 2. Different levels of interest shown by users in dating websites

Action	A likes B	B likes A
A reads profile of B	Possibly	Unknown
A reads profile of B, A sends a message to B	Yes	Unknown
A reads profile of B, A does not send a message to B	No	Unknown
A sends a constrained message to B	Yes	Unknown
A sends a constrained message to B, B replies positively to A	Yes	Yes
A sends a constrained message to B, B replies negatively to A	Yes	No
A sends a unconstrained message to B without previous constrained communication	Yes	Unknown
A sends a unconstrained message to B without previous constrained communication and receives a reply	Yes	Unknown*
A sends a unconstrained message to B without previous constrained communication and does not receive a reply	Yes	Unknown*

Dating can translate into finding a life-long partner or a casual/short-term partner. Both types of users have crucial differences that have high implications for an online dating website and its recommender system. Casual and short-term relationship seekers are likely to use the website for longer periods of time and are more likely to have "successful" relationships with different people, while long-term relationship seekers are hoping to find that one person who will cause them to stop using the website.

Online dating websites (e.g. Yahoo! Personals, Match.com) allow users to create their profiles, browse and create constrained conversations[3] for free, while charging a fee for unconstrained communication such as email, chat and telephone call. Although, in theory, success is measured by the number of people who have found a partner using the website, in practice the best and easiest way of measuring success is the use of unconstrained communication. The increase in unconstrained communication is really important to online dating websites because it can lead to a real world dating scenario and also because it directly relates to most online dating websites' business models.

Online dating is an intrinsically reciprocal task which is very difficult for recommender systems for several reasons: (1) Online dating deals with a range of features that might not be represented in the website data, such as private personal expectations or experiences from previous relationships. (2) There are multiple fuzzy levels of interests between users, which are difficult to capture. (3) Users may seek communication with others whose profiles do not precisely agree with the users' explicit preferences. (4) People change preferences over time.

Difficulty (1) is beyond the scope of this research as it involves psychological and sociological issues, which we cannot currently address. The different levels of interests of difficulty (2) can be addressed during website design. Existing websites show different levels of interest similar to the ones shown in Table 2. Some of the level of interest of user B toward user A that are marked as unknown and are marked with an asterisk can only be determined if the exchanged messages between users are read and analysed.

[3] Constrained conversation are fixed and predefined messages that users exchange in order to show interest (or not) in each other.

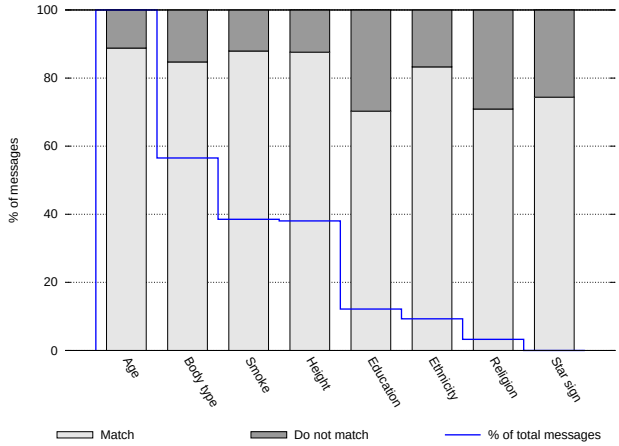


Fig. 2. Percentage of contacts from users which follows and does not follow the extrinsic model of preferences defined by the user

Difficulty (3) involves the creation of intrinsic models of user preference and how they differ from the extrinsic models created by the user. We observed that although people can define the characteristics of those they wish to date, users of online dating systems communicate with people whose features are different to the ones they specified. Figure 2 shows the percentage of constrained messages that were sent to a user who did not possess an important feature that was specified by the sender. For instance, 14% of messages from people with extrinsic preference were sent by users who specified education as an important attribute, 30% of those messages were sent to users who had a different level of education to the one specified in the sender's preferences. This either indicates that users cannot properly define the desired characteristics of their ideal date, or that users are willing to deviate from their ideal date when a *potentially good* date is found.

We believe that although an ideal date description is better than no description at all, the most reliable representation of someone's preferences can be inferred using the person's online date contacts. The inferred representation can be used to create reciprocal or non-reciprocal recommenders that produce lists of recommendations that match each user's own preferences.

As most content-based recommenders, such recommenders based on preference may suffer from lack of training data and over-specialisation. It is important to account for changes to user preferences (difficulty 4); such an effect can be harder to identify when a large body of previous contact is used to inferred the user's preferences. It is also important to learn how much evidence is needed to be able to provide a reliable recommendation. For instance, if user A contacted B and B is the only user contacted by A, a recommender system cannot assume that A only wants to contact users with the same features as B.

The use of reciprocal recommenders in online dating allows the creation of meaningful recommendations even when no preferences can be established for some users. For instance, in the cold start problem, if a user is new to the system and has not explicitly defined his/her preferences, nor contacted any other user (i.e. no implicit preferences), we can assume that this user does not have a preference and he/she will like any user. With this assumption, we can either create a list of recommendations for recommender $R1$ (or for recommender $R2$) that involves all users in the system. With such list, the reciprocal recommendation $RR(u)$ for a new user u is all users i whose preferences match user u. Similarly, if we only know the preferences of a user x then the reciprocal recommender $RR(x)$ for this user is equivalent to his non-reciprocal recommender $R1(x)$.

It is important to highlight that the degree of preference between a user with no known preferences and all users in the database has to be kept small, but non-zero. The degree of preference must be kept small so it does not interfere with the reciprocity when preferences are known. In this way, a ranked list of recommendations should contain all users who reciprocally match each other's preferences followed by a list of users who likes the user being recommended or is liked by him/her.

Another difficulty with implementing reciprocal recommenders and in particular with dating websites is that although some users are clearly more popular than others, the website needs to be careful to balance the load of the recommendations in order not to overwhelm users and to provide a good opportunity for interaction to all the users. Popular users might have characteristics that make them popular, such as a beautiful face or a charming smile, but they will not respond positively to everyone. On the other hand, there is a large number of users who do not have the same appeal as the popular users but are more likely to engage in successful interactions with other users.

6 Concluding remarks

In this paper we defined Reciprocal Recommenders, a class of recommender systems applicable to a number of domains such as online dating, recommending mentors, business partners or friends. This type of recommenders differs from the traditional recommender systems and has not received sufficient attention in the recommender community. We reviewed the relevant literature highlighting the need and importance of reciprocity in certain tasks. We identified challenges and discussed promising approaches to address them. Finally, we presented a case study in the area of online dating, illustrating the important concepts.

We have already implemented different techniques for reciprocal recommendations, which we plan to evaluate using data from an existing online dating website and also from an employment website. Future work will also include measuring the impact of reciprocity in domains where reciprocity is not obvious such as in online auctions and classified advertising.

Acknowledgements

This research was funded by the Smart Services Co-operative Research Centre.

References

1. Z. Abbassi, S. Amer-Yahia, L. V. Lakshmanan, S. Vassilvitskii, and C. Yu. Getting recommender systems to think outside the box. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 285–288, New York, 2009.
2. L. Brožovský and V. Petříček. Recommender system for online dating service. *CoRR*, abs/cs/0703042, 2007.
3. R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.
4. O. Celma and P. Cano. From hits to niches?: or how popular artists can bias music recommendation and discovery. In *NETFLIX '08: Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 1–8, New York, NY, USA, 2008. ACM.
5. R. F. Emmerink. Ai dating: Development of a novel dating application with fuzzy inferencing. Master's thesis, Faculty of Computing Sciences and Engineering, De Montfort University, September 12 2008.
6. J. Malinowski, T. Keim, O. Wendt, and T. Weitzel. Matching people and jobs: A bilateral recommendation approach. In *Proceedings of the 39th Annual Hawaii International Conference on System Sciences*, volume 6, page 137c, 2006.
7. B. Nag. Vibes: A platform-centric approach to building recommender systems. *IEEE Data Eng. Bull.*, 31(2):23–31, June 2008.
8. K. Onuma, H. Tong, and C. Faloutsos. Tangent: a novel, 'surprise me', recommendation algorithm. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 657–666, 2009.
9. S. T. Park and W. Chu. Pairwise preference regression for cold-start recommendation. In *RecSys '09: Proceedings of the third ACM conference on Recommender systems*, pages 21–28, New York, NY, USA, 2009. ACM.
10. D. Richards, M. Taylor, and P. Busch. Expertise recommendation: A two-way knowledge communication channel. *Autonomic and Autonomous Systems, International Conference on*, 0:35–40, 2008.
11. J. B. Schafer, J. A. Konstan, and J. Riedl. E-commerce recommendation applications. *Data Min. Knowl. Discov.*, 5(1-2):115–153, 2001.
12. K. Swearingen and R. Sinha. Beyond algorithms: An HCI perspective on recommender systems. In *ACM SIGIR'01 Workshop on Recommender Systems*, 2001.
13. L. Terveen and D. W. McDonald. Social matching: A framework and research agenda. *ACM Transactions on Computer-Human Interaction*, 12(3):401–434, 2005.
14. J. Vassileva, G. Mccalla, and J. Greer. Multi-agent multi-user modeling in i-help. *User Modeling and User-Adapted Interaction*, 13(1-2):179–210, 2003.
15. P. L.-K. Wong and P. Ellis. Social ties and partner identification in sino-hong kong international joint ventures. *J. Int. Bus. Stud.*, 33(2):267–289, 2002.
16. Z. Wu, S. Jiang, and Q. Huang. Friend recommendation according to appearances on photos. In *MM '09: Proceedings of the seventeen ACM international conference on Multimedia*, pages 987–988, New York, NY, USA, 2009. ACM.
17. B. Yu and M. P. Singh. Searching social networks. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 65–72, New York, NY, USA, 2003. ACM.

Recommending based on rating frequencies: Accurate enough?

Fatih Gedikli and Dietmar Jannach

Technische Universität Dortmund,
44227 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Since the development of the comparably simple neighborhood-based methods in the 1990s, a plethora of techniques has been developed to improve various aspects of collaborative filtering recommender systems such as predictive accuracy, scalability to large problem instances or the capability to deal with sparse data sets. Many of the recent algorithms rely on sophisticated methods which are based, for instance, on matrix factorization techniques or advanced probabilistic models or require computationally intensive model-building phases. In this work we evaluate the accuracy of a new and extremely simple prediction method that uses the user's and the item's most frequent rating value to make a rating prediction. The evaluation on two standard test data sets shows that the accuracy of the algorithm is on a par with the standard collaborative filtering algorithms on dense data sets and outperforms them on sparse rating databases. Besides that, the algorithm's implementation is trivial, has a high prediction coverage, requires no complex offline pre-processing or model-building phase and can generate predictions in a constant time.

1 Introduction

Collaborative filtering is one of the most successful technologies for recommender systems [AT05]. Pure collaborative filtering recommender systems only rely on a given user-item rating matrix to make rating predictions for items that the *active user* has not seen yet. Early neighborhood-based recommendation schemes simply used the k nearest neighbors (kNN) as predictors for unseen items. Later on, a broad range of more advanced and sophisticated methods have been applied to better exploit the given rating information and to improve the recommendation process in one or the other dimension. Examples for such methods include matrix factorization, various probabilistic models, clustering techniques, graph-based approaches as well as machine learning techniques, based on, e.g., association rule mining, see also [AT05].

Typically, the more elaborate approaches outperform the commonly-used kNN baseline method in terms of accuracy in particular for sparse data sets or in terms of scalability as they rely on offline pre-processing or model-building phases. In [LM05], Lemire and Maclachlan formulate additional desirable features of a recommendation scheme such as that they are easy to implement, can

be updated on the fly, are efficient at query time and are "reasonably" accurate. Their evaluation shows that the proposed *Slope One* family of item-based recommender algorithms, which is based on the computation of "popularity differentials between items for users", leads despite its simplicity to relatively accurate predictions (measured in terms of Mean Absolute Error). Due to its simplicity, different implementations of the algorithm in various programming languages and frameworks are available today.

In this paper, we propose an even simpler recommendation scheme, RF-Rec, which is only based on the absolute frequencies of the different rating values per user and per item. The method is therefore trivial to implement, can generate predictions in constant time, does not require a computationally intensive offline model-building phase, and at the same time leads to competitive prediction coverage and accuracy results in particular for sparse data sets.

In the rest of the paper, we will first describe the RF-Rec recommendation scheme in more detail and present results of an experimental evaluation on two commonly-used data sets.

2 Recommending based on rating frequencies

Let us illustrate the RF-Rec recommendation scheme with a simplified and relatively sparse rating database shown in Figure 1. The goal in our example is to predict Alice's rating for item $I3$.

	I1	I2	I3	I4	I5	Average
Alice	1	1	**?**	5	4	2.75
U1	2		5	5	5	4.25
U2			1	1		1.00
U3		5	1	1	2	2.25
Average	1.50	3.00	2.33	3.00	3.67	

Fig. 1. Example user-item rating matrix.

When adopting a user-based kNN scheme, probably no prediction can be made because only one relatively similar user $U1$ exists which could be taken as a predictor for Alice. If we allow also such small neighborhood sizes, the prediction for Alice will usually consist of taking the neighbor's rating for $I3$ and using it for the prediction by making a weighted addition to Alice's average rating. Similarly, in an item-based kNN approach, Alice's rating value for item $I4$, whose rating vector is similar to the one of $I3$ will be taken as a predictor. In both cases, the prediction for Alice for item $I3$ will be rather high.

In our approach, however, the predictions are based on absolute rating frequencies. The prediction function for a given user u and an item i in the RF-Rec recommendation scheme is defined as follows:

$$pred(u,i) = \underset{r \in possibleRatings}{\arg\max} \Big(\big(freqUser(u,r) + 1 + \mathbb{1}_{avg-user}(u,r) \big) * \big(freqItem(i,r) + 1 + \mathbb{1}_{avg-item}(i,r) \big) \Big)$$

where $freqUser(u,r)$ is the frequency of ratings with value r of the target user u and $freqItem(i,r)$ is the frequency of ratings with value r of the target item i. $\mathbb{1}_{avg-user}(u,r)$ and $\mathbb{1}_{avg-item}(i,r)$ are indicator functions which return 1 if the given rating corresponds to the rounded average rating of the target user or target item accordingly and 0 otherwise.
In the example, where the frequency of ratings of Alice are [1:2, 2:0, 3:0, 4:1, 5:1] and the rating frequencies of item $I3$ are [1:2, 2:0, 3:0, 4:0, 5:1] we would do the following calculations:

Rating value 1: (2+1+0)*(2+1+0) = 9
Rating value 2: (0+1+0)*(0+1+1) = 2
...
Rating value 5: (1+1+0)*(1+1+0) = 4

Since the formula result for rating 1 is the highest, we would predict that Alice would give a "1" to item $I3$, which is strongly different from the ratings that we would predict with the other methods.

The rationale of the prediction scheme is as follows. First, instead of taking rating *averages* into account for the calculations (as done in kNN-based approaches and also in Slope One) we rely on rating *frequencies*. Intuitively, this can be advantageous in case of extreme ratings, i.e., since Alice only gave very low and very high ratings and at the same time item $I3$ also only received extreme ratings. Incorporating user or item averages would move the predictions away from these extremes. When using the Slope One scheme, 2.38 would be the predicted rating for Alice, which is slightly below her average. Note that in [HKR00], the authors *Herlocker et al.* have also observed that high variance in the rating data can lead to decreased recommendation accuracy.

The "1" in the middle of our formula is used to avoid that in situations, in which a user has never given a rating (or an item never received a particular rating), the whole term is multiplied with zero. The indicator function in our scheme shall help in situations, in which several ratings have the same frequency counts. If this is the case and in addition one of these ratings corresponds to the average rating, we add some small extra weight to it, thus very slightly preferring the average rating.

Regarding *prediction coverage*, that is, the question for what percentage of items a recommender can generate predictions, note that in contrast to kNN approaches that often use similarity and neighborhood size thresholds, our recommendation scheme can make predictions if at least one rating for the target item or one rating by the user is available.

In order to measure the predictive accuracy of the method, we therefore evaluated our approach on two popular data sets using a common experimental procedure and accuracy metric. The results are described in the following section.

3 Experimental Evaluation

Algorithms, data sets and metrics. As data sets for the evaluation, we used the 100k-MovieLens rating database (100,000 ratings by 943 users on 1,682 items) and a snapshot of the Yahoo!Movies data set (211,231 ratings by 7,642 users on 11,915 items)[1]. The MovieLens data set only contains users who have rated at least 20 items; the minimum number of rated items per user in the Yahoo! data set is 10.

The density level of the data sets were varied by using subsamples of different sizes. The smallest subsample contained 10% of the original data. In this subsample, the average number of ratings per user was around 10 for the MovieLens data set and 3 for the Yahoo!Movies data set. Further measurements were taken in steps of 10% up to the 90% data set, which corresponds to the usual 90% train/test ratio for *Mean Absolute Error* (MAE) measurements.

We compared the following algorithms: user-based kNN (using default voting, Pearson similarity and the neighborhoodsize of 30 as suggested as optimal value in literature), item-based kNN (Mahout's item-based kNN-method implementation with Pearson similarity)[2], Slope One [LM05], Bias from Mean, Per User Average, the recent recursive prediction algorithm (RPA) [ZP07] (using larger, empirically determined neighborhood sizes of 100) and RF-Rec.

Results. Figure 2 (a) shows the MAE values for different training set sizes for the MovieLens data set. Up to the 50% level, RF-Rec has consistently better accuracy than all other techniques. Above that level, the accuracy of RF-Rec (0.742) is comparable to Slope One (0.743) and RPA (0.734). Note that RF-Rec due to its nature leads to 100% prediction coverage also for very sparse data sets. In contrast, the coverage of the user-based kNN method, for example, slowly increases from 60% to 95% when increasing training set ratio from 10% to 90%, see Figure 3. In the experiments, we further varied the neighborhood size ns of the user-based kNN method. Increasing ns to 300 lead to the observation that the accuracy improved and was comparable to the one of RPA for training set sizes higher than 50%.

Figure 2 (b) shows the results for the Yahoo! data set. We can observe similar accuracy values also for this data set. In particular, the improvement of our RF-Rec algorithm is even stronger on that data set. A possible explanation for this observation could be the different sparsity levels of the two data sets, i.e., assuming that RF-Rec works particularly well for sparse settings, it is intuitive that even better results can be achieved on the sparser Yahoo! data set (0.9976 sparsity) than on the MovieLens data set (0.9369 sparsity).

[1] http://www.grouplens.org/node/73, http://webscope.sandbox.yahoo.com
[2] http://lucene.apache.org/mahout

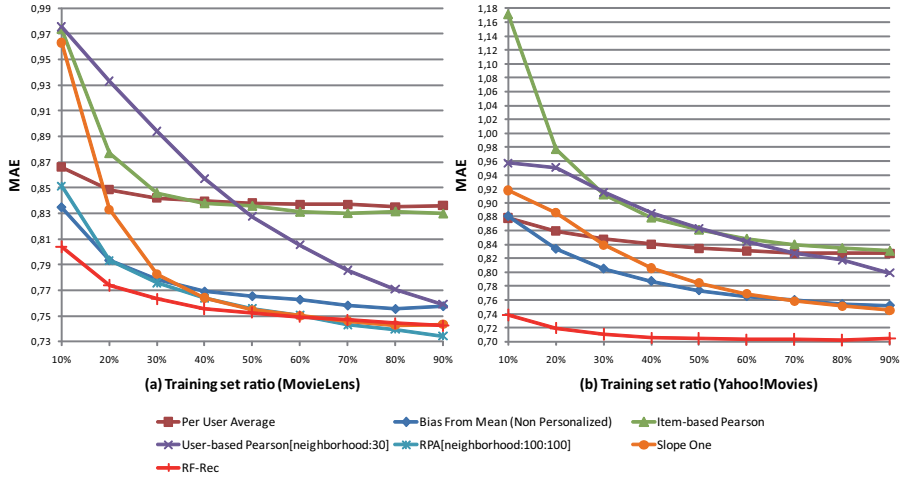


Fig. 2. MAE values for different training set sizes: MovieLens (a), Yahoo!Movies (b).

Training set ratio	RF-Rec	Slope One	User-based kNN	Item-based kNN
10%	100%	97%	60%	44%
20%	100%	98%	58%	94%
30%	100%	99%	70%	98%
..	..	..	..	..
90%	100%	99%	95%	99%

Fig. 3. Prediction coverage of recommendation approaches (MovieLens).

Overall, these accuracy findings indicate that RF-Rec has a constantly good performance which is quite independent of the training set ratio. RF-Rec is despite its simplicity suitable to generate predictions with an accuracy which is comparable to existing approaches and is even better for sparse data sets, which can often be found in practice.

Computational complexity. In the RF-Rec scheme, the "model-building" phase obviously consists of calculating the frequencies of the individual rating values per user and per item, which can be accomplished in a single scan of the matrix; the frequency statistics can be easily updated when new ratings are available. Given u users, i items and v possible rating values, the memory requirements for the model are constant: $(u * v) + (i * v)$. Also the calculation of predictions can be done with the formula from Section 2 in constant time. In absolute numbers, "model-building" requires less than 10 seconds even when hundreds of millions of ratings exist; predictions can be calculated in a few milliseconds on a standard desktop computer. We compared our method with Mahout's item-based kNN-method implementation on the 1 million MovieLens data set: model-building takes 500ms in our approach as opposed to 6 minutes with

Mahout. Generating a prediction takes only 3ms in our framework (and 100ms with Mahout), which is a very important factor in high-traffic recommenders in which up to 1,000 parallel requests have to be served [JH09].

4 Summary

In this work we proposed a new, frequency-based recommendation scheme that leads to good predictive accuracy and is at the same time highly scalable and very easy to implement. Our future work includes the evaluation of the approach on the Netflix data in order to compare it to the results of more recent methods; in addition, we will also compare the predictive accuracy of the different methods based on precision and recall.

Overall, our evaluation demonstrated that comparably good results can be achieved with simple methods and that the accuracy values that can be achieved with the help of "classical" methods such as item-based kNN and even the more recent RPA method are actually very small, which could make the payoff of using more sophisticated methods in some settings questionable.

We hope that light-weight approaches like our RF-Rec method help to further promote the use of recommender systems in practice; by making the software used in our experiments publicly available[3], we hope to contribute to the comparability of different algorithms since our study revealed that relevant algorithmic details and parameters are often not reported in sufficient detail.

References

[AT05] Gediminas Adomavicius and Alexander Tuzhilin, *Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions*, IEEE Trans. on Knowl. and Data Eng. **17** (2005), no. 6, 734–749.

[HKR00] Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl, *Explaining collaborative filtering recommendations*, Proc. ACM Conference on Computer Supported Cooperative Work (New York, NY, USA), 2000, pp. 241–250.

[JH09] Dietmar Jannach and Kolja Hegelich, *A case study on the effectiveness of recommendations in the mobile internet*, Proceedings of the 2009 ACM Conference on Recommender Systems (New York, NY, USA), 2009, pp. 41–50.

[LM05] Daniel Lemire and Anna Maclachlan, *Slope one predictors for online rating-based collaborative filtering*, Proceedings of the 5th SIAM International Conference on Data Mining (Newport Beach, CA), 2005, pp. 471–480.

[ZP07] Jiyong Zhang and Pearl Pu, *A recursive prediction algorithm for collaborative filtering recommender systems*, Proceedings of the 2007 ACM Conference on Recommender Systems (Minneapolis, MN, USA), 2007, pp. 57–64.

[3] http://ls13-www.cs.uni-dortmund.de/rec_suite.zip