Probabilistic Memory-based Collaborative Filtering

Kai Yu, Anton Schwaighofer, Volker Tresp, Xiaowei Xu, Hans-Peter Kriegel

***Abstract*— Memory-based collaborative filtering (CF) has been studied extensively in the literature and has proven to be successful in various types of personalized recommender systems. In this paper we develop a probabilistic framework for memory-based CF (PMCF). While this framework has clear links with classical memory-based CF, it allows us to find principled solutions to known problems of CF-based recommender systems. In particular, we show that a probabilistic active learning method can be used to actively query the user, thereby solving the "new user problem". Furthermore, the probabilistic framework allows us to reduce the computational cost of memory-based CF by working on a carefully selected subset of user profiles, while retaining high accuracy. We report experimental results based on two real world data sets, which demonstrate that our proposed PMCF framework allows an accurate and efficient prediction of user preferences.**

***Index Terms*— Collaborative filtering, recommender systems, profile density model, active learning, data sampling.**

I. INTRODUCTION

Information on the web has been growing explosively in recent years. Information filters emerged to meet the challenge of information search on the WWW, a problem which may be compared to "locating needles in a haystack that is growing exponentially" [1]. Recommender systems are a class of information filters which have proven to be successful. For example, recommender systems on e-commerce web sites assist users to find their favorite CDs or books. Similarly recommender systems assist in locating items like web pages, news, jokes, or movies, from thousands or even millions of items.

Content-based filtering (CBF) and collaborative filtering (CF) are two technologies used in recommender systems. CBF systems analyze the contents of a set of items together with the ratings provided by individual users to infer which non-rated items might be of interest for a specific user. Examples include [2], [3], [4]. In contrast, collaborative filtering methods [5], [6], [1] typically accumulate a database of item ratings cast by a large set of users, and then use those ratings to predict a query user's preferences for unseen items. Collaborative filtering does not rely on the content descriptions of items, but purely depends on preferences expressed by a set of users. These preferences can either be expressed explicitly by numeric ratings, or can be indicated implicitly by user behaviors, such as clicking on a hyperlink, purchasing a book or reading a particular news article.

One major difficulty in designing CBF systems lies in the problem of formalizing human perception and preferences. Why one user likes or dislikes a joke, or prefers one CD over another is virtually impossible to formalize. Similarly it is difficult to derive features which represent the difference between an average news article and one of high quality. CF provides a powerful way to overcome these difficulties. The information on personal preferences, tastes, and quality are all carried in (explicit or implicit) user ratings.

CF-based recommender systems have successfully been applied in areas ranging from e-commerce (for example, Amazon and CDnow[1]) to computer-supported collaborative work [7]. CF research projects include Grouplens (the first automatic CF algorithm, [5]), Ringo [6], Video Recommender [8], Movielens [9], and Jester [10].

A. Collaborative Filtering Algorithms

A variety of CF algorithms have been proposed in the last decade. One can identify two major classes of CF algorithms [11], memory-based approaches and model-based approaches.

Memory-based CF can be motivated from the observation that people usually trust the recommendations from like-minded friends. These methods apply a nearest-neighbor-like scheme to predict a user's ratings based on the ratings given by like-minded users. The first CF systems Grouplens [5] and Ringo [6] fall into this category. In the literature, the term collaborative filtering is sometimes used to refer only to the memory-based methods.

In contrast, model-based CF first learns a descriptive model of user preferences and then uses it for predicting ratings. Many of these methods are inspired from machine learning algorithms. Examples include neural network classifiers [1], induction rule learning [12], linear classifiers [13], Bayesian networks [11], dependency networks [14], latent class models or mixture models [15], [16], item-based CF [17], principle component analysis based CF [10], association rule mining [18], and hybrids of model- and memory-based approaches [19].

B. Motivation

Up to now, research on CF primarily focused on exploring various learning methods, hoping to improve the prediction accuracy of recommender systems. Other important aspects, like scalability, accommodating to new data, and comprehensibility have received little attention. In the following we will review five general issues which are important for CF and greatly motivated the work presented in this paper.

Kai Yu is with Siemens Corporate Technology, Information and Communications, Munich, Germany, and also with the Institute for Computer Science, University of Munich, Germany

Anton Schwaighofer is with Siemens Corporate Technology, Information and Communications, Munich, Germany, and also with the Institute for Theoretical Computer Science, Graz University of Technology, Austria

Dr. Volker Tresp is a research scientist at Siemens Corporate Technology, Information and Communications, Munich, Germany

Dr. Xiaowei Xu is an associate professor at the University of Arkansas at Little Rock, USA

Prof. Hans-Peter Kriegel is a full professor at the Institute for Computer Science, University of Munich, Germany

[1] `www.amazon.com`, `www.cdnow.com`

1) Accuracy: As a central issue in CF research, prediction accuracy has received a high degree of attention, and various methods were proposed for improvement. Still, conventional memory-based methods using Pearson correlation coefficient remain among the most successful methods in terms of accuracy. The experiments presented in Sec. V-D show that our proposed probabilistic interpretation of memory-based CF can outperform a set of other memory- and model-based CF approaches.

2) Interactive Learning of User Profiles: A recommender system cannot provide accurate service to a new user, whose preferences are initially unknown. This has been referred to as the "new user problem" [2], [20], [21] Before being able to make predictions, a CF system typically requires the new user to rate a list of query items in an initial information gathering stage. Efficient heuristics [21] are essential to select informative query items and thus keep the information gathering stage as short as possible, since users may easily lose patience when faced with a long list of query items.

Within our proposed probabilistic framework for CF, we show in Sec. III how informative query items can be selected in a principled way. At each information gathering step, those query items are presented to the user which are expected to maximally sharpen the user's profile. Our experiments (see Sec. V-E) confirm that this interactive approach outperforms other ways of selecting query items [21] both in terms of necessary user effort and achieved accuracy of predictions.

3) Efficiency: Memory-based CF often suffers from slow response time, because each single prediction requires the scanning of a whole database of user ratings. This is a clear disadvantage when compared to the typically very fast responses of model-based CF. In the proposed probabilistic memory-based CF approach, predictions are generated from a carefully selected small subset of the overall database of user ratings, which we call *profile space*. As a consequence, predictions can be made much faster than in a classical memory-based CF system. Still, the accuracy of a system using the full data set can be maintained. We will describe this process of data selection in Sec. IV. The results presented in Sec. V-F confirm that the constructed profile space does indeed allows a both accurate and fast prediction of user ratings.

4) Incrementally accommodating to new data: Recommender systems must be capable of handling new data, be it new users or new items. For example, in a music recommender system, the recommender system must be able to adapt itself to newly arising styles of music and thus new preference patterns. This suggests that the training process of any underlying CF algorithm should be incremental. However, model-based CF approaches are typically trained using batch algorithms. To our knowledge, little work has addressed the use of on-line learning in CF. Thus, re-training a model with new data can become quite expensive, in particular if it needs to be performed regularly [11]. In contrast, memory-based CF can easily accommodate to new data by simply storing them. In the proposed probabilistic memory-based CF framework, this goal can be achieved by a straight-forward extension of the data selection procedure introduced in Sec. IV.

5) Comprehensibility: The results in [22] indicate that allowing users to know more about the result-generating process can help them understand the strengths and weaknesses of CF systems. With this knowledge, users can make low-risk decisions. For example, consider the following two cases: (1) Among Julia's like-minded users there are 50% percent of users who rated 'like' to Titanic, while 50% of them rated 'dislike'. (2) In the other case, most of her neighbors give neutral ratings to that movie. A traditional CF system may only give a neutral rating in both of the cases. A more sophisticated system may remind Julia of the underlying reasons in the first case and, for example, output an estimated distribution of a user's rating for some item, either in graphical or textual form ("I guess you will like that movie, and I am pretty sure (or very unsure) about that"). This suggests that a probabilistic CF approach, as presented in this paper, can improve the comprehensibility and thus the acceptance of a CF system. Furthermore, memory-based CF has a clear interpretation that can be easily conveyed to users, such as "You seem to be sharing opinions with user A, who liked the following items…".

C. Overview of Our Approach

In this paper, we introduce probabilistic memory-based collaborative filtering (PMCF), a probabilistic framework for CF systems that is similar in spirit to the classical memory-based CF approach. A schematic drawing of the components of PMCF is shown in Fig. 1.

As the basic ingredient, we present a probabilistic model for user preferences in Sec. II. We use a mixture model built on the basis of a set of stored user profiles; thus the model clearly links with memory-based CF methods.

Various heuristics to improve memory-based CF have been proposed in the literature. In contrast, extensions to PMCF can be based on a principled probabilistic way. We argue that this is one of the major advantages of PMCF. We use PMCF to derive solutions for two particularly important problems in CF.

The first one concerns the new user problem. An active learning extension to the PMCF system can actively query a user for additional information, in case the available information is insufficient.

The second major extension aims at reducing the computational burden in the prediction phase typically associated with memory-based CF. PMCF allows us to select a small subset, called the *profile space*, from a (possibly huge) database of user ratings. The selection procedure is derived directly from the probabilistic framework and ensures that the small profile space leads to predictions that are as accurate as predictions made by using the whole data base of user ratings.

D. Structure of this Article

This paper is organized as follows. In Sec. II, we describe the framework of probabilistic memory-based CF (PMCF). In Sec. III, we present an active learning extension of PMCF to gather information about a new user in a particularly efficient way that requires a minimum of user interaction. In Sec. IV, we show how to construct the profile space for the PMCF model, which is a small subset of the available user rating data. We

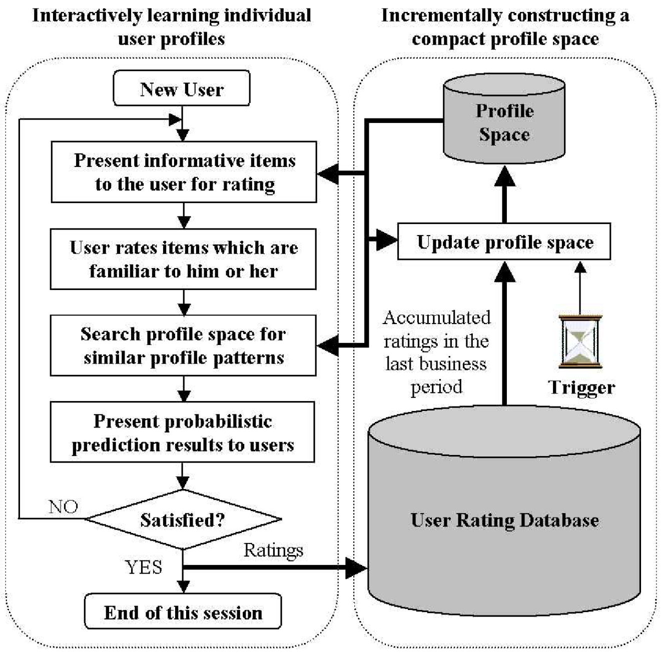

Interactively learning individual user profiles
Incrementally constructing a compact profile space
New User
Present informative items to the user for rating
User rates items which are familiar to him or her
Search profile space for similar profile patterns
Present probabilistic prediction results to users
Satisfied?
NO
YES
Ratings
End of this session
Profile Space
Update profile space
Accumulated ratings in the last business period
Trigger
User Rating Database

Fig. 1. A schematic drawing of the components of probabilistic memory-based collaborative filtering (PMCF). Through an active learning scheme (presented in Sec. III), the profile of a new user can be inferred with a minimum of required user effort. User ratings are stored in a database, from which a compact representation—the profile space—can be constructed in order to make fast predictions (presented in Sec. IV)

present experimental results that demonstrate the effectiveness of PMCF, the active learning extension and the profile space construction in Sec. V. We end the paper by conclusions and an outlook in Sec. VI.

II. Probabilistic Memory-Based CF

In this section a general probabilistic memory-based CF (PMCF) approach is introduced. Probabilistic CF has been a vivid research topic. Examples include Bayesian networks [11], dependency networks [14], latent class models or mixture models [15], [16], and hybrids of memory- and model based systems [19]. The work presented here has been inspired by [19], in that we also aim at connecting memory- and model-based CF in a probabilistic way. While [19] mainly focusses on making predictions, we use the probabilistic model for further extensions of the CF system, some of which will be described in Sec. III and IV.

A. Notation

Suppose that we have gathered K users' ratings on a given item set $\mathcal{I}$ of size $M = |\mathcal{I}|$. Let $x_{i,j} \in \mathbb{R}$ be the rating of user i on item j and let $\mathcal{D}$ with $(\mathcal{D})_{i,j} = x_{i,j}$ be the $K \times M$ matrix of all ratings. $\mathcal{R}_i$ is the set of items for which user i has actually given ratings, $\mathcal{R}_i \subseteq \mathcal{I}$. If an item has not been rated, we set $x_{i,j}$ to a neutral rating n_i, which we will define later. We denote by $\boldsymbol{x}_i$ the vector of all ratings of user i. In the following text, user i's ratings $\boldsymbol{x}_i$ are often referred as user i's *profile*. We also maintain a smaller set of user profiles, the *profile space* $\mathcal{P}$, which consists of a subset of rows of $\mathcal{D}$. Without loss of generality, we assume that the profile space is built up[2] from the ratings of the first N users, i.e. the first N rows of $\mathcal{D}$, where typically $N \ll K$.

In CF terminology, the *active user* is the user that queries the CF system for recommendations on some items. We denote the active user's ratings by $\boldsymbol{a}$. By $\boldsymbol{a}^r$, we denote the ratings the active user has already provided (for items $\in \mathcal{R}_a$), and $\boldsymbol{a}^n$ are the yet unknown ratings. The total rating vector $\boldsymbol{a}$ is thus the union of $\boldsymbol{a}^r$ and $\boldsymbol{a}^n$.

As mentioned above, we use a neutral rating n_i for all items a user i has not given an explicit rating, i.e. $x_{i,j} = n_i$ if $j \notin \mathcal{R}_i$. In order to compute n_i, we assume a Gaussian prior for the neutral rating with mean m_0 which is estimated as the overall mean of user ratings. If we further assume that n_i is also Gaussian distributed with mean m_0 we can estimate the neutral rating as

$$n_i = \frac{\sum_{j \in \mathcal{R}_i} x_{i,j} + C m_0}{|\mathcal{R}_i| + C} \tag{1}$$

where C is the ratio of the variance of the ratings for user i and the variance of m_0. We determined a suitable value for C based on cross validation experiments. We found $C = 9$ to work effectively on the data we consider.

B. A Density Model for Preference Profiles

We assume a generative probabilistic model in which the ratings $\boldsymbol{a}$ of an active user are generated based on a probability density of the form

$$p(\boldsymbol{a}|\mathcal{P}) = \frac{1}{N} \sum_{i=1}^{N} p(\boldsymbol{a}|i), \qquad \boldsymbol{x}_i \in \mathcal{P} \tag{2}$$

where $p(\boldsymbol{a}|i)$ is the probability of observing the active user's ratings $\boldsymbol{a}$ if we assume that a has the same profile class as the ith profile prototype in $\mathcal{P}$, i.e. user i's profile. The density expressed by Eq. (2) models the influences of other like-minded users' preferences on the active user a. For the mixture components $p(\boldsymbol{a}|i)$, we use Gaussian[3] density functions. Assuming that ratings on individual items are independent, given a profile i, we get

$$\begin{aligned} p(\boldsymbol{a}|i) &= \prod_{j \in \mathcal{I}} p(a_j|i) & (3) \\ &= \prod_{j \in \mathcal{I}} \frac{(2\pi)^{-1/2}}{\sqrt{\sigma^2 + d_{j \notin \mathcal{R}_i} \sigma_0^2}} \exp\left(-\frac{1}{2} \frac{(a_j - x_{i,j})^2}{\sigma^2 + d_{j \notin \mathcal{R}_i} \sigma_0^2} \right) \end{aligned}$$

Here, $d_{j \notin \mathcal{R}_i} = 1$ if $x_{i,j}$ is unrated and $d_{j \notin \mathcal{R}_i} = 0$ otherwise. This model can be motivated as a mixture model, with the prototype profiles $\boldsymbol{x}_i$ serving as cluster centers, or as a Parzen density model on the profile space $\mathcal{P}$. The additional variance for unrated items takes into account the uncertainty of the estimated rating.

[2]We will show in Sec. IV how a compact and accurate profile space $\mathcal{P}$ can be incrementally built from a given set of user ratings $\mathcal{D}$.

[3]We are a little inaccurate here and assume for simplicity that our rating scale is continuous and unbounded, ignoring the fact that ratings are often given on a discrete scale. One might also chose mixture components that fit particular data, for example binomial distributions for discrete ratings.

In our experiments, we set σ_0^2 to be the overall variance of user ratings. σ^2 was optimized by maximizing the leave-one-out likelihood of profiles

$$\sum_{\boldsymbol{a} \in \mathcal{P}} p(\boldsymbol{a}|\mathcal{P} \setminus \boldsymbol{a}) \tag{4}$$

with respect to σ^2. σ^2 is tuned after constructing the profile space (see Sec. IV) and left constant thereafter. Note that, technically, profiles take on different meanings: If they are part of the data base, they represent prototype vectors defining the component densities in Eq. (3). If we consider the active user's profile, the profile corresponds to a sample generated from the probability density defined in the same equation.

C. A Probabilistic Approach to Estimating User Ratings

We can now calculate the posterior density of the active user a's ratings on not yet rated items, denoted by $\boldsymbol{a}^n$, based on the ratings $\boldsymbol{a}^r$ user a has already given. Using the previously defined density model for user ratings, we find

$$\begin{aligned} p(\boldsymbol{a}^n|\boldsymbol{a}^r, \mathcal{P}) &= \frac{p(\boldsymbol{a}^n, \boldsymbol{a}^r|\mathcal{P})}{p(\boldsymbol{a}^r|\mathcal{P})} & (5)\\ &= \frac{\sum_{i=1}^N p(\boldsymbol{a}^n, \boldsymbol{a}^r|i)}{\sum_{i=1}^N p(\boldsymbol{a}^r|i)} & (6)\\ &= \sum_{i=1}^N p(\boldsymbol{a}^n|i) \Pr(i|\boldsymbol{a}^r, \mathcal{P}). & (7) \end{aligned}$$

$\Pr(i|\boldsymbol{a}^r, \mathcal{P})$ indicates the *a posteriori* probability of user a having the ith prototype profile, given the ratings user a already has provided. It thus models the "like-mindedness" of active user a to other users i in the profile space $\mathcal{P}$:

$$\Pr(i|\boldsymbol{a}^r, \mathcal{P}) = \frac{p(\boldsymbol{a}^r|i)}{\sum_{i=1}^N p(\boldsymbol{a}^r|i)}. \tag{8}$$

Within the PMCF model, predictions for the active user are thus made by combining the predictions based on other prototype users $\boldsymbol{x}_i$, weighted by their degree of like-mindedness to user a. This puts the key idea of memory-based collaborative filtering into a probabilistic framework.

Note that the computational complexity of prediction is $O(NM)$, i.e. it is linear in the size of the profile space. In Sec. IV we will show how to obtain a profile space that is much smaller than the complete user rating database $\mathcal{D}$. Making predictions only on basis of the small profile space thus brings a significant reduction of overall computational cost.

III. An Active Learning Approach to Learning User Profiles

In the previous section, we introduced the PMCF framework and showed how predictions can be made. In this section we will use an active learning approach to efficiently learn the profile of an individual user. The active learning approach integrates smoothly into the PMCF framework and provides a solution for the "new user problem". By presenting a set of most informative query items in an interactive process, we can learn about the profile of a new user with a minimum of user effort.

A. The New User Problem

For users that are new to a recommender system, no information about their preferences is initially known. Thus, the recommender system typically requests them to rate a set of query items. Using the ratings on these query items, the CF system can then start making recommendations.

There are several important reasons why this set of query items should be selected carefully: (1) Users are not willing to rate a long list of items; (2) Users cannot rate items unknown to them; (3) Rating results for some items might be very informative for determining a user's profile whereas rating results for other items might not provide useful new information. So far little work has been done to address[4] the new user problem. [21].

In the next sections, we will present an approach for selecting query items that requires particularly little user effort, yet allows fast learning about the user's preferences.

B. Identifying Informative Query Items

To achieve an efficient interactive learning of user profiles, we put the selection of query items into a decision theoretic framework (see for example Sec. 4.3 of [24]). First, one needs to define a loss function, evaluating the quality of the system before querying a new item $\lambda(\boldsymbol{a}^r, \mathcal{P})$ and after querying the user for item j, $j \notin \mathcal{R}_i$ and after having obtained rating a_j. We denote the loss after querying by $\lambda(a_j, \boldsymbol{a}^r, \mathcal{P})$. The goal is now to select the query item j such that the expected loss

$$E_{p(a_j|\boldsymbol{a}^r, \mathcal{P})} \left[\lambda(a_j, \boldsymbol{a}^r, \mathcal{P})\right] \tag{9}$$

is minimized. The expectation is calculated here with respect to the predicted probability of user a's ratings for item j.

The most important ingredient is the loss function $\lambda(a_j, \boldsymbol{a}^r, \mathcal{P})$. We propose to use the entropy of the like-mindedness $\Pr(i|\boldsymbol{a}^r, \mathcal{P})$ as the loss function. $\Pr(i|\boldsymbol{a}^r, \mathcal{P})$ describes the like-mindedness of a user i in the profile space $\mathcal{P}$ with active user a, given a's ratings $\boldsymbol{a}^r$. In an extreme case, $\Pr(i|\boldsymbol{a}^r, \mathcal{P})$ has a uniform distribution, which means that the profile of user a is completely unclear. In contrast, a sharp peak in the distribution of $\Pr(i|\boldsymbol{a}^r, \mathcal{P})$ indicates that user a has similar preferences as a small group of like-minded users. It thus seems natural to choose those query items that minimize the uncertainty (thus, the entropy) of user a's like-mindedness.

Putting this into a formal setting, we can write for the loss function

$$\lambda(a_j, \boldsymbol{a}^r, \mathcal{P}) = -\sum_{i=1}^N \Pr(i|a_j, \boldsymbol{a}^r, \mathcal{P}) \log \Pr(i|a_j, \boldsymbol{a}^r, \mathcal{P}). \tag{10}$$

By $\Pr(i|\boldsymbol{a}^r, a_j, \mathcal{P})$ we denote like-mindedness, computed with an updated vector of ratings for the active user, who now also has rated the (previously unrated) item j.

We can now define the expected benefit (Sec. 4.3.2 of [24]) for querying item j as

$$E[B(j)] = E_{p(a_j|\boldsymbol{a}^r, \mathcal{P})} \left[\lambda(a_j, \boldsymbol{a}^r, \mathcal{P})\right] - \lambda(\boldsymbol{a}^r, \mathcal{P}) \tag{11}$$

[4]A method for improving the accuracy of CF systems by adding extra query items has been presented in [23]. This approach might also be adapted to solve the new user problem.

and terminate the query process if the expected benefit is less than a threshold related to the cost of querying.

Our algorithm for query item selection is myopic in the sense that the algorithm only looks one step ahead. In contrast, a hyperopic algorithm would aim at finding the optimal *sequence* of query items to be presented. However, since hyperopic optimization is computationally intractable, myopia is a standard approximation used in sequential decision-making problems [25], [26].

C. Identifying the Items Possibly Known to the Active User

If we wanted to use the active learning approach described in the previous section directly, we would most often get a “don’t know” as the answer to most of the query items. Users of a CF system can typically provide ratings for only few of the items. For example, in a recommender system for movies, users may typically have seen a few dozen movies out of the several hundred movies contained in the data base. It may be quite informative to know the user’s opinion on an unusual movie, yet it is likely that the user will not be able to give this movie any rating.

Thus, we must also predict the probability that a user is able to rate[5] a given query item. This can be achieved by again referring to the like-mindedness of users. In Eq. (5), predictions for active user a were built from a sum of other users’ ratings, weighted by their degree of like-mindedness $\Pr(i|\boldsymbol{a}^r, \mathcal{P})$. Similarly, we can predict the probability of user a being able to rate item j, given his or her other ratings $\boldsymbol{a}^r$, by checking user a’s like-minded users:

$$\Pr(\text{user } a \text{ can rate item } j|\boldsymbol{a}^r, \mathcal{P}) = \sum_{i=1}^{N} \Pr(\text{user } a \text{ can rate item } j|i) \Pr(i|\boldsymbol{a}^r, \mathcal{P}) \tag{12}$$

$\Pr(\text{user } a \text{ can rate item } j|i)$ is the probability that a can rate item j, given that users a and i (as described by prototype profile $\boldsymbol{x}_i$) agree on which items they are able to rate. We assume for simplicity that user a can rate exactly the same[6] movies as user i:

$$\Pr(\text{user } a \text{ can rate item } j|i) = \begin{cases} 1 & \text{if user } i \text{ has rated item } j \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

D. A Summary of the Active Learning Process

Using the ideas described in the previous sections, we propose the following iterative scheme to learn the profile of the active user a:

1) Out of the set of items that have not yet been rated by user a, find those k_1 items with the highest probability of being known to user a, i.e. those items with the highest value for Eq. (12).
2) Out of these k_1 items, select a subset of k_2 items that lead to the highest reduction of uncertainty about the user’s profile, i.e. the items with the highest expected benefit in Eq. (11).
3) Display those k_2 items to the user for rating. Collect the ratings and update the vector of ratings $\boldsymbol{a}$.
4) Terminate if the user is not willing to answer any more queries or if the expected benefit of querying (as defined in Eq. (11)) is below a certain threshold. Otherwise, go to step 1.

In the very first step, where nothing is known about user a, we assume equal like-mindedness of user a with all profiles in $\mathcal{P}$. Thus, user a will be presented the k_2 most popular items as query items.

E. Implementation

1) Parameters for Active Learning: The value of k_1 (see step 1 of Sec. III-D) should be carefully selected. If k_1 is too small, for example, as small as k_2, then the selection procedure is too much biased by Eq. (12), and thus might miss out informative items—the system performs too little exploration. If k_1 is too large, too many items will be presented to the user which the user is not able to rate. In cross validation experiments, we found that $k_1 = 50$ gives the best results for the data we consider. The value for k_2 is rather uncritical. We used $k_2 = 10$, because it seems reasonable to display 10 items on a normal-sized PC screen. Thus, at each iteration, we first find the 50 candidate items with largest probability of being known, and then identify 10 query items according to the expected reduction of uncertainty in like-mindedness.

2) Computational Complexity: The most costly part in this active learning approach is the evaluation of Eq. (11), where the expected reduction of uncertainty in like-mindedness is computed. The algorithm needs to exhaust $O(ck_1)$ possibilities of user feedbacks at each iteration (where c is the number of ratings a user might possibly give to a presented query item, and k_1 is the number of candidate items) and calculate the entropy of the like-mindedness for each case. This again requires evaluating Eq. (2) with changed preference vector $\boldsymbol{a}$. Fortunately, Eq. (2) factorizes along items, thus the distances only need to be re-calculated along the dimensions of the newly rated items. This greatly reduces the overall computational cost.

3) Alternative Methods: Several of the approaches proposed in the active learning literature may be adopted for CF. A common approach is *uncertainty sampling* [27], which has been successfully applied to text categorization [27] and image retrieval [26] to reduce the number of training examples. The general idea behind all proposed variants of uncertainty sampling is to present the unlabeled examples for which the outcome is most uncertain, based on the current predictions. In a CF scenario, one is interested in predicting a user’s ratings for non-rated items. Thus, the variance of predictions $\operatorname{var} p(a_j|\boldsymbol{a}^r, \mathcal{P})$ is an appropriate measure of uncertainty. An advantage of this approach lies in its low computational cost, since we only have to compute the predictions $p(a_j|\boldsymbol{a}^r, \mathcal{P})$ for all yet unrated items.

Another low complexity method for query item selection is *entropy sampling* [21]. Here, we consider $\Pr_j(s)$, the fraction of users who had given a particular rating $s \in \{s_1, \ldots, s_c\}$ for item j. Query items are selected such that the entropy of $\Pr_j(s)$ is maximized.

[5] Another way of solving this problem would be to integrate this probability into the loss function Eq. (10) for the active learning approach. We do not pursue this solution in the present article.

[6] This is a strong assumption, yet due to the weighting introduced by the like-mindedness we obtain meaningful results

We will show in Sec. V-E that the method based on uncertainty of like-mindedness (as outlined in Sec. III-B) achieves best results, both in terms of achieved accuracy and in terms of required user input.

IV. Incrementally Constructing Profile Space

In Sec. II we introduced a probabilistic model for describing user preferences. This model was based on a given set of user profiles, the profile space $\mathcal{P}$. In this section, we will show how this profile space can be constructed, by selecting informative user profiles from the overall database of user ratings $\mathcal{D}$. Since the profile space typically contains only a low number of user profiles (as compared to the often huge $\mathcal{D}$), it allows us to build compact models and make predictions efficiently, while maintaining a high accuracy. It thus solves the well-known problem that predictions of traditional memory-based CF methods are rather time-consuming.

A. Kullback-Leibler Divergence for User Profile Sampling

Let's assume that there exists an optimal density model for user ratings, which we denote by $p^{\text{opt}}(\boldsymbol{x})$. Naturally we do not have access to this optimal model but we work with a non-optimal model $p(\boldsymbol{x}|\mathcal{P})$, as given in Eq. (2), based on some profile space $\mathcal{P}$. The key idea of our proposed selection procedure is to select the profile space $\mathcal{P}$ such that the density $p(\boldsymbol{x}|\mathcal{P})$ is as close as possible to the optimal density $p^{\text{opt}}(\boldsymbol{x})$.

To measure the distance of these two distributions, we use the Kullback-Leibler divergence (KL-divergence [28]). We denote the KL-divergence of the two distributions by

$$D\left(p(\boldsymbol{x}|\mathcal{P})||p^{\text{opt}}(\boldsymbol{x})\right) = \int p^{\text{opt}}(\boldsymbol{x}) \log \frac{p^{\text{opt}}(\boldsymbol{x})}{p(\boldsymbol{x}|\mathcal{P})} d\boldsymbol{x} \qquad (14)$$

where the integral is over the whole space of user rating vectors. The KL-divergence is always non-negative and is zero when two compared distributions are identical. Assuming that the total set of user ratings $\mathcal{D}$ constitutes a set of independent samples drawn from $p^{\text{opt}}(\boldsymbol{x})$, we can approximate the KL-divergence by Monte-Carlo integration [29]:

$$\begin{aligned}\tilde{D}\left(p(\boldsymbol{x}|\mathcal{P})||p^{\text{opt}}(\boldsymbol{x})\right) &= \frac{1}{K}\sum_{i=1}^{K} \log \frac{p^{\text{opt}}(\boldsymbol{x}_i)}{p(\boldsymbol{x}_i|\mathcal{P})} &(15)\\ &= \frac{1}{K} \log \frac{p^{\text{opt}}(\mathcal{D})}{p(\mathcal{D}|\mathcal{P})} &(16)\end{aligned}$$

where K is the number of users in $\mathcal{D}$.

As stated above, we wish to minimize the KL-divergence $\tilde{D}(p(\boldsymbol{x}|\mathcal{P})||p^{\text{opt}}(\boldsymbol{x}))$ so that the density $p(\boldsymbol{x}|\mathcal{P})$ best approximates $p^{\text{opt}}(\boldsymbol{x})$. Since $p^{\text{opt}}(\mathcal{D})$ is constant, Eq. (15) can be minimized by maximizing the likelihood of the user rating database $\mathcal{D}$ with respect to the profile space $\mathcal{P}$. Finding the optimal profile space $\mathcal{P}$ is clearly an intractable task, we thus switch to an iterative greedy approach for constructing $\mathcal{P}$.

B. Incremental Profile Space Construction

For constructing the profile space $\mathcal{P}$ from a data base $\mathcal{D}$ of user ratings, we consider an incremental scenario. Given the current profile space $\mathcal{P}$, which profile pattern $\boldsymbol{x}_i \in \mathcal{D}$ should be included such that the updated profile space $\mathcal{P} \cup \boldsymbol{x}_i$ can achieve the maximum reduction in KL-divergence, according to Eq. (15)?

The reduction in KL-divergence caused by including $\boldsymbol{x}_i$ in $\mathcal{P}$ can be written as

$$\Delta_i = \begin{aligned}&=\tilde{D}\left(p(\boldsymbol{x}|\mathcal{P})||p^{\text{opt}}(\boldsymbol{x})\right) - \tilde{D}\left(p(\boldsymbol{x}|\mathcal{P}\cup\boldsymbol{x}_i)||p^{\text{opt}}(\boldsymbol{x})\right)\\ &=\frac{1}{K}\log\frac{p(\mathcal{D}|\mathcal{P}\cup\boldsymbol{x}_i)}{p(\mathcal{D}|\mathcal{P})}\end{aligned} \qquad (17)$$

Mind that this step causes the optimal density $p^{\text{opt}}(\boldsymbol{x})$ to drop out. According to Bayes' rule, the likelihood of the overall data $\mathcal{D}$, given the updated profile space $\mathcal{P} \cup \boldsymbol{x}_i$ can be written as follows:

$$p(\mathcal{D}|\mathcal{P}\cup\boldsymbol{x}_i) = p(\mathcal{D}|\mathcal{P})\frac{p(\boldsymbol{x}_i|\mathcal{D})}{p(\boldsymbol{x}_i|\mathcal{P})} \qquad (18)$$

where $p(\boldsymbol{x}_i|\mathcal{D})$ is the likelihood of $\boldsymbol{x}_i$, based on a model that uses the complete data as the profile space. Combining Eq. (17) and (18), the optimal profile $\boldsymbol{x}$ to be selected is given by:

$$\arg\max_i \Delta_i = \arg\max_{\boldsymbol{x}_i\in\mathcal{D}\setminus\mathcal{P}} \frac{p(\boldsymbol{x}_i|\mathcal{D})}{p(\boldsymbol{x}_i|\mathcal{P})} \qquad (19)$$

An intuitive interpretation of this selection scheme is as follows: Eq. (19) suggests that profiles $\boldsymbol{x}_i$ with low $p(\boldsymbol{x}_i|\mathcal{P})$ but high $p(\boldsymbol{x}_i|\mathcal{D})$ will be selected. $p(\boldsymbol{x}_i|\mathcal{P})$ encodes how likely a profile $\boldsymbol{x}_i$ is, given our current knowledge $\mathcal{P}$, while $p(\boldsymbol{x}_i|\mathcal{D})$ encodes the likelihood and thus the "degree of typicalness" of profile $\boldsymbol{x}_i$ in the overall data $\mathcal{D}$. The profile selection scheme thus focusses on profiles that are novel to our current knowledge (encoded by the current profile space), but are in fact typical in the real world (represented by the whole data $\mathcal{D}$). Thus, this sampling scheme will result in removing redundancies (we only focus on novel data that is not yet included in the profile space) and in removing outliers (outliers can be considered untypical data).

Still, Eq. (19) does not give a practical algorithm, since it requires evaluating $O(K)$ profiles, $K = |\mathcal{D}|$, where each evaluation requires $O(K)$ steps to actually build $p(\boldsymbol{x}_i|\mathcal{D})$. This leads to the clearly impractical overall runtime of $O(K^2)$. Practical variants will be discussed in the next section.

C. Implementation

Constructing a profile space $\mathcal{P}$ according to Eq. (19) is sometimes referred to as *full greedy selection*. This can only be done efficiently if the associated objective function can be computed cheaply—which is not the case for the likelihood ratio we consider here. In related problems, it has been suggested to consider small subsets of candidates, evaluate the objective function for each candidate, and select the best candidate out of this subset (see, for example, Sec. 6.5 of [30]).

We thus obtain the following profile sampling scheme to build $\mathcal{P}$ from $\mathcal{D}$:

1) Select a subset $\mathcal{C}$ of candidate profiles at random from $\mathcal{D} \setminus \mathcal{P}$.
2) Compute the likelihood $p(\boldsymbol{x}_i|\mathcal{P})$ for each candidate profile $\boldsymbol{x}_i \in \mathcal{C}$, based on the current profile space $\mathcal{P}$.
3) Compute the likelihood $p(\boldsymbol{x}_i|\mathcal{D})$ for each $\boldsymbol{x}_i \in \mathcal{C}$, based on the complete data $\mathcal{D}$.
4) Include the best candidate profile in the profile space:

$$\mathcal{P} \leftarrow \mathcal{P} \cup \arg\max_{\boldsymbol{x}_i \in \mathcal{C}} \frac{p(\boldsymbol{x}_i|\mathcal{D})}{p(\boldsymbol{x}_i|\mathcal{P})} \tag{20}$$

5) Terminate, if the profile space has reached a given maximum size or if the reduction of KL-divergence is below a given threshold.

It has been suggested in [30] that subsets of size $|\mathcal{C}| = 59$ can be guaranteed to select profiles that are better than 95% of all other profiles with confidence 95%. In our experiments, we aim at achieving higher efficiency and thus use subsets of size $|\mathcal{C}| = 7$. This corresponds to selecting profiles that are better than 80% of all others with confidence 80%.

D. Constructing Profile Spaces in a Dynamic Environment

While the sampling approach presented in the previous section works fine in a static environment with a fixed database of user ratings, it needs to be refined to work in a dynamic environment. The dynamics arises from changing preferences patterns (for example, new styles of music in a music recommender system) and the ever growing database of user ratings. Since user profiles are typically collected incrementally, we suggest an incremental extension to the basic sampling scheme presented in Sec. IV-C. We assume that the profile space is being updated after a fixed period of time, e.g. each day or week. The new user profiles gathered during this period are being processed and some of them will be added to the profile space.

Assuming that we have a data base of user ratings $\mathcal{D}$. From $\mathcal{D}$, we have already constructed a profile space $\mathcal{P}$. After collecting user profile data for some time, we get an updated data base $\mathcal{D}^+$, with $\mathcal{D}^+ = \mathcal{D} \cup \Delta\mathcal{D}$. In order to build the according profile space $\mathcal{P}^+$, select the set of candidate items $\mathcal{C}$ from $\mathcal{D}^+$. Select the most informative profile and update the profile space $\mathcal{P}^+$:

$$\mathcal{P}^+ \leftarrow \mathcal{P}^+ \cup \arg\max_{\boldsymbol{x}_i \in \mathcal{C}} \frac{p(\boldsymbol{x}_i|\mathcal{D}^+)}{p(\boldsymbol{x}_i|\mathcal{P}^+)} \tag{21}$$

Terminate if the new profile space $\mathcal{P}^+$ has reached a given size or if none of the candidate items $\boldsymbol{x}_i \in \mathcal{C}$ leads to a reduction of KL-divergence. Otherwise, select a new candidate set and proceed.

Through this straight-forward extension we can retain the basic idea of using a small profile space, as introduced in Sec. IV-B, while now being capable of incrementally processing new data.[7]

E. Computational Complexity

For the basic profile space construction, as outlined in Sec. IV-B, the computational complexity is as follows:

[7]One might also consider the case of removing certain (outdated) user profiles from $\mathcal{P}$, yet we did not evaluate this idea in the present work.

Evaluating the density function $p(\boldsymbol{x}_i|\mathcal{D})$ for a candidate profile $\boldsymbol{x}_i$ (see Eq. (19)) requires scanning the whole data base $\mathcal{D}$ with K user ratings. Its complexity is thus $O(K)$. Since all potential profile spaces $\mathcal{P}$ are subsets of $\mathcal{D}$, $\mathcal{P} \subseteq \mathcal{D}$, one can easily construct $p(\boldsymbol{x}_i|\mathcal{P})$ as a "by-product" when scanning the data base in order to find $p(\boldsymbol{x}_i|\mathcal{D})$. Both steps are thus $O(K)$, with $K = |\mathcal{D}|$. Constructing a profile space of size N requires a total of $O(KN)$ operations. Once the profile space, is constructed, one also needs to update the variance σ^2 according to Eq. (4). This is done with a leave-one-out scheme, its complexity is thus $O(N^2)$.

Since one would typically keep the profile space resident in memory, the memory consumption of the profile space construction is $O(N)$, with $N = |\mathcal{P}|$.

The suggested method for constructing a profile space $\mathcal{P}$ thus has the same complexity as making predictions in a traditional memory-based CF method. Yet, as described in Sec. IV-D, profile space construction can be seen as a background process that is being triggered by time or when unused computing power is available. Thus, its time consumption is not visible to a user of the CF system. We argue that the so achieved *shift of workload* is important, since it greatly improves the efficiency of front-end processing, namely, making predictions.

V. EMPIRICAL STUDY

In this section we report results from applying the probabilistic memory-based collaborative filtering (PMCF) framework to two CF benchmark data sets, EACHMOVIE and JESTER. We report results on prediction accuracy, efficiency of learning individual user profiles (based on the ideas presented in Sec. III) and accuracy of the constructed profile spaces (using the incremental scenario of Sec. IV).

A. Data Sets

We apply the PMCF framework to the following two benchmark data sets:

- EACHMOVIE[8] contains ratings from $72,916$ users on $1,628$ movies. User ratings were recorded on a discrete scale from zero to five. On average, each user rated about 30 movies. EACHMOVIE is one of the most widely used data sets in recommender system research.
- JESTER[9] contains ratings from $17,998$ users on 100 jokes, continuously valued from -10 to 10. On average, each user rated about 50 jokes. We transferred the ratings to a discrete scale $\{-10, -9, \ldots, 9, 10\}$.

B. Evaluation Metrics and Experimental Setup

In collaborative filtering research, one is typically interested in two types of accuracy, the accuracy for predicting ratings and the accuracy for making recommendations. The first one measures the performance when explicitly predicting the active users ratings on some unseen items. The second one focusses

[8]Available from the Digital Equipment Research Center at `http://www.research.digital.com/SRC/EachMovie/`

[9]JESTER stems from a WWW-based joke recommender system, developed at the University of California, Berkeley [10]. It is available from `http://shadow.ieor.berkeley.edu/humor/`

on finding an accurate ordering of a set of unseen items, in order to recommend the top ranked items to the active user. These two scenarios require different experimental setups and metrics, which we will describe now.

1) Accuracy of Predicting Ratings: To evaluate the accuracy when the CF system is asked to predict an active user's ratings, we use the mean absolute error (MAE, the average absolute difference between the actual ratings and the predicted ratings). This measure has been widely used in previous collaborative filtering research [11], [31], [19], [5], [6].

We examine the accuracy of predictions in two experimental setups, ALLBUTONE and GIVEN5, which were introduced in [11]:

- ALLBUTONE evaluates the prediction accuracy when sufficient information about the active user is available. For each active user (from the test set[10]) we randomly hide one of the rated items and predict its rating, based on the ratings on other non-hidden items.
- GIVEN5 evaluates the performance of a CF system when only little information about a user is available. For each active user, we retain only 5 ratings. The CF system predicts the ratings of hidden items, based on the 5 visible ratings.

It has been argued that the accuracy of a CF system is most critical when predicting extreme ratings (very high or very low) for items [19], [6]. Since the goal of a CF system is to make recommendations, high accuracy on high and low rated items is of most importance. One would like to present those items (in particular, products) that the active user likes most, and avoid anything the user dislikes. Therefore, for both of the above ALLBUTONE and GIVEN5 setups, we use two settings EXTREME and ALL (see [19]). The ALL setting corresponds to the standard case where the CF system is asked to predict any of the hidden ratings. In the EXTREME setting, the CF system only predicts ratings that are on the end of the rating scales. For EACHMOVIE, these extreme ratings are $\{0, 1, 2, 4, 5\}$, and ratings below -5 or above 5 for JESTER.

2) Accuracy of Recommendations: We use precision and recall to evaluate the accuracy of recommendations. These two metrics have been extensively used in information retrieval and collaborative filtering research [1], [18]. In our experiments, precision is the percentage of items recommended to a user that the user actually likes. Recall is the percentage of items the user likes that are also recommended by the CF system. For the EACHMOVIE data, we assume that users like those items (movies) which they had rated 4 or 5. For JESTER, we assume that users like those jokes that had been given a rating larger than 5.

To compute precision and recall, we use the following setup. For each active user (from the test set[11]) we randomly hide 30 of the user's ratings[12]. The CF system then predicts the ratings for these items, based on the remaining visible ratings. The top ranked items out of these 30 items are then recommended to the user and used to evaluate precision and recall. We compute precision and recall for two cases, where we either recommend the top 5 or the top 10 ranked items. These two cases will be labeled TOP5 and TOP10 in the table of results.

3) Training and Test Sets: For comparing the accuracy of predictions of PMCF with that of Bayesian network CF [11] on the EACHMOVIE data, we use exactly the same split as reported in [11], [19] with training and test sets of size 5000. To be able to evaluate the significance of our results, we use training and test sets (both of size 5000) drawn at random from the data, and repeat this five times.

Similarly, for evaluating the accuracy of prediction on the JESTER data, we take the first 5000 users as the training set, and the next 5000 as the test set. Five random splits are used for significance tests.

As mentioned above, we skip all test users that have rated less than 31 items when computing precision and recall, respectively less than two (six) items when computing the MAE in the ALLBUTONE (GIVEN5) setup. Final results for MAE, precision and recall are always averaged over all users in the test set.

C. Comparison With Other CF Methods

To compare the results of PMCF with other established CF methods, we report results in terms of MAE, precision and recall for PMCF and for the following methods that have proven successful in the CF literature.

- Memory-based CF with Pearson correlation coefficient [5], one of the most popular memory-based CF algorithms.
- Bayesian network CF [11]. Since we use exactly the same experimental setup and evaluation metrics for the EACHMOVIE data as reported in [11], we can directly compare the performance of Bayesian network CF with other methods. We did not implement Bayesian network CF for the JESTER data.
- Naïve Bayesian CF [32]. Despite its simplicity, the naïve Bayesian classifier has proven to be competitive with Pearson correlation CF.

All methods are evaluated in the setup described in Sec. V-B.

We compare the above listed methods with two variants of PMCF, which we label PMCF $\mathcal{P}$ and PMCF $\mathcal{D}$. For the PMCF $\mathcal{D}$ variant, we use the full training set to build the density model in Eq. (2), that is, the profile space is taken to be the full training data $\mathcal{P} = \mathcal{D}$. The other variant PMCF $\mathcal{P}$ is PMCF with profile space constructed from the training set $\mathcal{D}$ in the way described in Sec. IV. For both EACHMOVIE and JESTER, we constructed profile spaces with 1000 profiles (out of the training data of size 5000).

D. Evaluation of Accuracy

Tab. I and II summarize the performance of all evaluated CF methods in terms of accuracy for prediction and recommendation.

Tab. I lists results for accuracy of prediction that are based on one particular split of the data into training and test set that has also been used in [11]. It can be clearly seen that PMCF

[10]This naturally requires that we skip users in the test set that have only rated one single item, respectively users that rated less than 6 items in the GIVEN5 setup.

[11]The setup requires that we skip users who had rated less than 31 items.

[12]We experimented with different numbers here, for example, hiding 20 of the user's ratings. We found that the results were consistent throughout these experiments, thus we present only results for one setup.

achieves an MAE that is about 7-8% lower than the MAE of the competing methods. The results also suggest that PMCF is particularly suitable for making predictions when only very little information about the active user is given: PMCF achieved a particularly high improvement of accuracy for the GIVEN5 scenarios.

For the accuracy of predictions, we also evaluated all methods (except for the Bayesian network) with five different randomly drawn training and test sets of size 5000, and did a pairwise comparison of results using a paired t-test. The test confirmed that both variants of PMCF performed better than all of the competing method with a significance level of 99% or above. Comparing PMCF $\mathcal{P}$ and PMCF $\mathcal{D}$, we noted that both performed almost identical for the GIVEN5 setups. For the two ALLBUTONE setups, PMCF $\mathcal{D}$ achieved a slightly better performance.

The results for accuracy of recommendation listed in Tab. I are averages over five different random splits into training and test data, as described above. The large advantage of PMCF in terms of accuracy of prediction does not fully carry over to the accuracy of recommendation. Still, a consistent and statistically significant gain in performance could be achieved. Precision and recall of PMCF are typically about 2-3% better than those of the competing methods. A larger performance gain was always achieved in the TOP5 setup. Again, a pairwise comparison of results in a paired t-test was conducted. Results for one of the two PMCF variants that are marked in bold in Tab. I are better than those of the two competing methods with a significance level of 95% or above. Similarly, results marked in italics achieve a significance level of 90% or above.

Overall, we could verify that our proposed probabilistic memory-based CF framework achieves an accuracy that is comparable or superior to other approaches that have been proposed for collaborative filtering.

E. *Evaluation of Profile Learning*

In Sec. III, we proposed an active learning approach to interactively learn user profiles. In this section we investigate the performance of this learning process in a series of experiments that simulate the interaction between users and the recommender system.

We use the training/test split described in Sec. V-B.3. For each test user, ratings are randomly split into a set S of 30 items and the remaining items U. We assume that the test user initially has not rated any items, and we wish to infer his profile using the active learning approach. To obtain long learning curves, we restrict the test set to users who had rated at least 60 items. This leaves us with 972 and 1340 test users respectively for the EACHMOVIE and JESTER data sets.

The interactive sessions are simulated as follows: The recommender system selects the 10 most informative items[13] according to the criterion described in Sec. III-D. User feedback is taken from the actual ratings the user has given on an item, if the item is in set U. Otherwise it is left unrated, simulating that the user is not able to give feedback on this particular item. We make a series of such simulated interactions, $t = 1, 2, \ldots$, gaining more and more knowledge about the user's profile. For test user a, we compute the MAE when predicting the ratings in set S and the precision for making recommendations in set S, denoted by $\mathrm{MAE}(a,t)$ and $\mathrm{precision}(a,t)$. By averaging over all users in the test set, we obtain $\mathrm{MAE}(t)$ and $\mathrm{precision}(t)$.

Using MAE and precision, we compare the following 5 methods for selecting the query items:

1) Query item selection by minimizing the entropy of the like-mindedness, as outlined in Sec. III-D.
2) Uncertainty sampling, as described in Sec. III-E
3) Entropy sampling, as described in Sec. III-E
4) Popularity sampling: At each iteration, we present 10 of the most popular items to the test user
5) Random sampling: At each iteration t, we randomly select 10 query items

Methods 3, 4 and 5 have also been studied in [21].

The resulting learning curves $\mathrm{MAE}(t)$ and $\mathrm{precision}(t)$ for the above 5 methods are shown in Fig. 2 (for the EACHMOVIE data) and in Fig. 3 (for JESTER). The graphs clearly indicate that query item selection based on like-mindedness outperforms all other tested methods. Like-mindedness based selection is thus a method which achieves a maximum gain of information about a particular user with only a minimum of user effort.

For all of the tested methods, we also investigated the average number of items the user is being able to rate at a particular iteration t. The low performance of random and entropy based sampling, in particular on EACHMOVIE, can be explained by the fact that users are not able to answer the posed queries. The remaining three methods all achieve similar results for the average number of rated items. Yet, like-mindedness sampling seems to ask more informative questions, leading to the steepest learning curves among all methods in Fig. 2 and 3.

From the presented results, we conclude that like-mindedness based sampling is a sensible and accurate method of inferring user profiles and requires only a minimum amount of user effort. It has a particularly good performance on data sets with high sparsity such as EACHMOVIE, where only 3% of the items are rated, yet it also performs better than competing approaches on dense data sets (JESTER).

F. *Evaluation of Constructing Profile Spaces*

We showed in Sec. IV how a small profile space $\mathcal{P}$ for the PMCF model can be constructed out of a large data base of user ratings $\mathcal{D}$. In this section, we investigate how the profile space construction relates to the achievable accuracy for predictions and recommendations in the PMCF model.

To this aim, we use the split of training and test data described in Sec. V-B.3. From the training data $\mathcal{D}$, the profile space $\mathcal{P}$ is constructed iteratively as outlined in Sec. IV. At certain intervals[14], we evaluate the performance of the PMCF method, based on the profile space constructed so far, on the test set. We use the mean absolute error MAE in the ALLBUTONE

[13]Query items might also be presented one by one, instead of using batches of 10 items. We chose the variant with 10 items since it seems more natural in an application scenario. Presenting items one by one can easily make users impatient.

[14]Evaluation is done when the profile space has reached a size of 60, 125, 250, 500, 1000, 2000 and 4000.

TABLE I

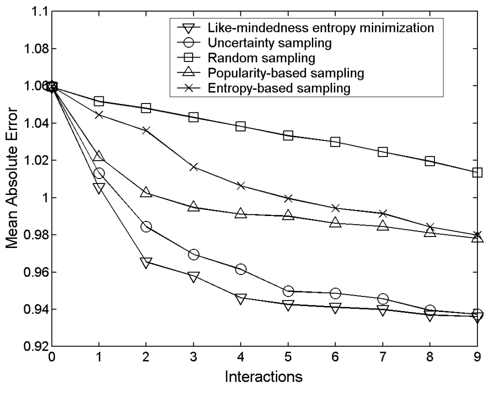
ACCURACY OF PREDICTIONS, MEASURED BY MEAN ABSOLUTE ERROR MAE, OF DIFFERENT CF METHODS. DETAILS ON THE INDIVIDUAL EXPERIMENTS ARE GIVEN IN SEC. V-B AND V-C. BOTH PMCF $\mathcal{P}$ AND PMCF $\mathcal{D}$ CONSISTENTLY OUTPERFORM THE COMPETING METHOD, IN PARTICULAR WHEN LITTLE INFORMATION IS GIVEN ABOUT THE ACTIVE USER IN THE GIVEN5 SCENARIO. THE RESULTS SHOWN HERE ARE BASED ON THE TRAINING/TEST SPLIT REPORTED IN SEC. V-B.3. ADDITIONAL EXPERIMENTS WITH 5 RANDOM SPLITS AND PAIRED t-TEST CONFIRMED THAT PMCF OUTPERFORMED THE COMPETING METHODS AT A SIGNIFICANCE LEVEL OF 99% OR ABOVE

	EACHMOVIE				JESTER			
	ALL		EXTREME		ALL		EXTREME	
	ALLBUTONE	GIVEN5	ALLBUTONE	GIVEN5	ALLBUTONE	GIVEN5	ALLBUTONE	GIVEN5
Pearson correlation	0.996	1.150	1.130	1.290	3.927	4.258	5.062	5.730
Bayesian networks	1.066	1.154						
Naïve Bayes	0.987	1.162	1.096	1.223	4.132	4.263	4.753	5.512
PMCF $\mathcal{D}$	0.966	1.008	1.010	1.112	3.544	3.967	4.408	5.219
PMCF $\mathcal{P}$	0.984	1.008	1.040	1.110	3.724	3.972	4.523	5.464

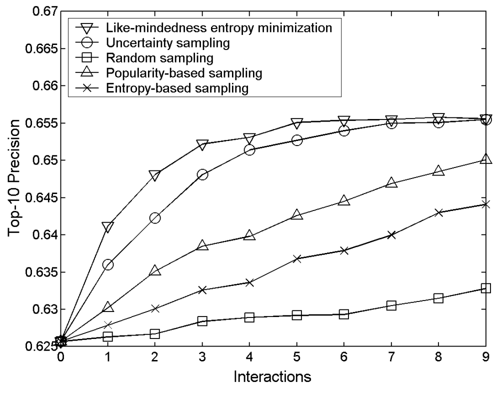
TABLE II

ACCURACY OF RECOMMENDATIONS, MEASURED BY PRECISION AND RECALL, OF DIFFERENT CF METHODS. ALL RESULTS IN THIS TABLE ARE AVERAGED OVER 5 RUNS, WHERE TRAINING AND TEST SETS HAD BEEN DRAWN AT RANDOM FROM THE TOTAL DATA SETS. MARKED IN BOLD ARE PMCF RESULTS THAT ARE SIGNIFICANTLY BETTER (WITH A SIGNIFICANCE LEVEL OF 95% OR ABOVE IN A PAIRED t-TEST) THAN THE COMPETING APPROACHES. MARKED IN ITALIC ARE PMCF RESULTS THAT ARE BETTER THAN THE COMPETING APPROACHES WITH A SIGNIFICANCE LEVEL OF 90% OR ABOVE. FURTHER DETAILS ON THE INDIVIDUAL EXPERIMENTS ARE GIVEN IN SEC. V-B AND V-C

	EACHMOVIE				JESTER			
	TOP5		TOP10		TOP5		TOP10	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Pearson correlation	0.703	0.284	0.656	0.510	0.406	0.251	0.386	0.454
Naïve Bayes	0.663	0.264	0.617	0.484	0.383	0.235	0.381	0.443
PMCF $\mathcal{D}$	**0.715**	**0.291**	**0.665**	**0.520**	**0.425**	**0.264**	**0.397**	**0.468**
PMCF $\mathcal{P}$	**0.713**	*0.288*	*0.659*	*0.512*	**0.416**	**0.256**	*0.391*	**0.464**

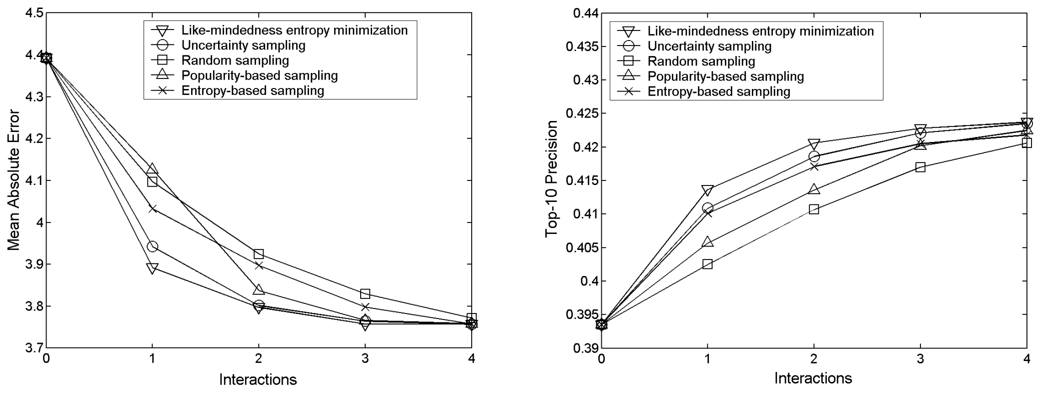


(a) Mean absolute error $\mathrm{MAE}(t)$



(b) $\mathrm{precision}(t)$

Fig. 2. Learning individual user profiles for the EACHMOVIE data. Mean absolute error $\mathrm{MAE}(t)$ and $\mathrm{precision}(t)$ achieved after $t = 1, 2, \ldots$ steps of user interaction with different strategies for query item selection. Details of the experimental setup are given in Sec. V-E



(a) Mean absolute error $\mathrm{MAE}(t)$ (b) $\mathrm{precision}(t)$

Fig. 3. Learning individual user profiles for the JESTER data. Mean absolute error $\mathrm{MAE}(t)$ and $\mathrm{precision}(t)$ achieved after $t = 1, 2, \ldots$ steps of user interaction with different strategies for query item selection. Details of the experimental setup are given in Sec. V-E

setting and precision in the TOP10 setting as the measures of performance.

We so obtain a curve of performance versus size of the profile space. Since constructing the profile space uses a randomized strategy to select candidate profiles (see Sec. IV-C), we repeat this procedure 10 times. Thus, error bars for the performance of PMCF with a profile space of a given size can be plotted. As the baseline method, we use a PMCF model with a profile space drawn at random from the full training data $\mathcal{D}$.

The resulting curves for accuracy of prediction (MAE) and recommendation (precision) on the EACHMOVIE data are shown in Fig. 4, and in Fig. 5 for the JESTER data. All plots clearly indicate that the profile space construction presented in Sec. IV does bring significant advantages in terms of performance over a randomly chosen profile space. The gain in performance was particularly large for accuracy of recommendation on the JESTER data.

VI. CONCLUSIONS

In this paper we proposed a probabilistic framework for memory-based collaborative filtering (PMCF). The PMCF is based on user profiles in a specially constructed profile space. With PMCF the posterior distribution of user ratings can be used to predict an active user's ratings. An experimental comparison with other CF methods (memory-based CF with Pearson correlation, Bayesian networks, naïve Bayes) showed that PMCF outperforms the competing methods both in terms of accuracy for prediction and recommendation.

As one of its major advantages, PMCF allows extensions to the basic model on a sound probabilistic basis. We showed in Sec. III how an active learning approach can be integrated smoothly into the PMCF framework. Through active learning, the CF system can interactively learn about a new user's preferences, by presenting well selected query items to the user. Our results showed that the active learning approach performed better than other methods for learning user profiles, in the sense that it can make accurate predictions with only a minimum amount of user input.

In Sec. IV we used the probabilistic framework to derive a data selection scheme that allows the recommender system to make fast and accurate predictions. Instead of operating on a possibly huge database of user preferences (as traditional memory-based CF does), the data selection scheme allows us to use only a carefully selected subset, which we call the profile space. Using the so selected profile space in the PMCF model allows making fast predictions with only a small drop in performance over a PMCF model operating on the full data.

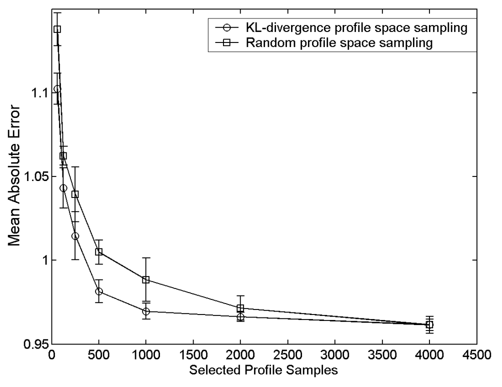
We believe that the PMCF framework will allow more extensions and thus can contribute to further improvements of recommender systems. A particularly promising research direction is the combination of CF methods with content based filtering into hybrid systems. We are currently working on a PMCF based hybrid system for image and text retrieval [33]. This system implicitly also solves the new item problem: If no user ratings are available for an item, predictions can still be made on the basis of the content description.

Our further work on the PMCF model will also include an improved model for user preferences. In Eq. (3), only items that were actually rated contribute to the model. An improved model could also take into account the information which items had not been rated. For example, in the EACHMOVIE data, a movie may have been unrated because a friend had dissuaded the user from seeing the movie. Thus, one may be able to extract a certain degree of information from the set of unrated items as well and further improve the accuracy of a CF system.

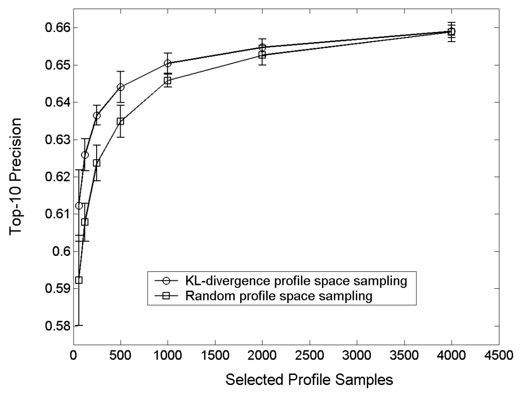
For the current PMCF system, as described in this article, the efficiency of the active learning scheme still needs to be improved. Active learning based on minimization of the entropy of like-mindedness achieves the best recommendation accuracy, yet the computational complexity is higher than that of competing methods such as uncertainty sampling.

ACKNOWLEDGMENTS

Anton Schwaighofer gratefully acknowledges support through an Ernst-von-Siemens scholarship. The authors thank

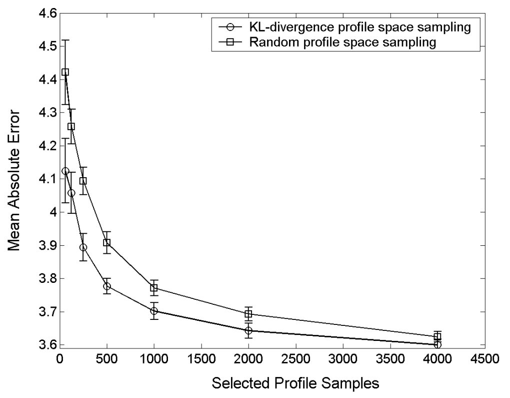


(a) Mean absolute error $\mathrm{MAE}(t)$

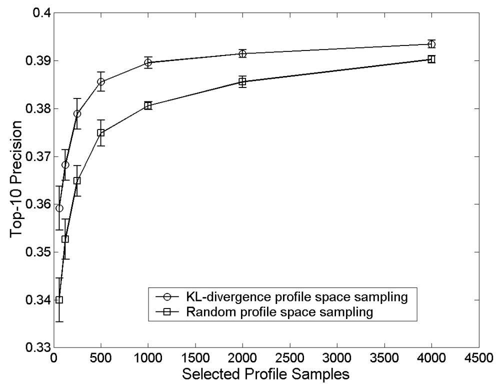


(b) $\mathrm{precision}(t)$

Fig. 4. Evaluating the profile space construction for the EACHMOVIE data set. Mean absolute error MAE and precision achieved with profile spaces of different size, that are either constructed based on KL-divergence (see Sec. IV) or drawn at random from the training data. The plot is averaged over 10 runs, with error bars



(a) Mean absolute error $\mathrm{MAE}(t)$



(b) $\mathrm{precision}(t)$

Fig. 5. Evaluating the profile space construction for the JESTER data set. Mean absolute error MAE and precision achieved with profile spaces of different size, that are either constructed based on KL-divergence (see Sec. IV) or drawn at random from the training data. The plot is averaged over 10 runs, with error bars

the three anonymous reviewers for their valuable comments and suggestions to improve the quality of the paper.

REFERENCES

[1] D. Billsus and M. J. Pazzani, “Learning collaborative information filters,” in *Proceedings of the 15th International Conference on Machine Learning*. 1998, pp. 46–54, Morgan Kaufmann, San Francisco, CA.

[2] M. Balabanovic and Y. Shoham, “Fab: Content-based, collaborative recommendation,” *Communications of the ACM*, vol. 40, no. 3, pp. 66–72, 1997.

[3] R.J. Mooney and L. Roy, “Content-based book recommending using learning for text categorization,” in *Proceedings of the Fifth ACM Conference on Digital Libaries*, San Antonio, US, 2000, pp. 195–204, ACM Press, New York, US.

[4] M. Pazzani, J. Muramastsu, and D. Billsus, “Syskill and webert: Identifying interesting web sites,” in *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, August 1996, pp. 54–61.

[5] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl, “Grouplens: An open architecture for collaborative filtering of netnews,” in *Proceedings of the 1994 Computer Supported Collaborative Work Conference*, Chapel Hill, North Carolina, 1994, pp. 175–186, ACM.

[6] U. Shardanand and P. Maes, “Social information filtering algorithms for automating ’word of mouth’,” in *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems*, 1995, vol. 1, pp. 210–217.

[7] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, “Analysis of recommendation algorithms for e-commerce,” in *Proceedings ACM E-Commerce Conference*, 2000, pp. 158–167.

[8] W. Hill, L. Stead, M. Rosenstein, and G. Furnas, “Recommending and evaluating choices in a virtual community of use,” in *Proceedings of ACM CHI’95 Conference on Human Factors in Computing Systems*, 1995, pp. 194–201.

[9] B.J. Dahlen, J.A. Konstan, J.L. Herlocker, and J. Riedl, “Jump-starting movielens: User benefits of starting a collaborative filtering system with dead data,” Tech. Rep. 7, University of Minnesota, 1998.

[10] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, “Eigentaste: A constant time collaborative filtering algorithm,” *Information Retrieval Journal*, vol. 4, no. 2, pp. 133–151, 2001.

[11] J. S. Breese, D. Heckerman, and C. Kadie, “Empirical analysis of predictive algorithms for collaborative filtering,” in *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence*, 1998, pp. 43–52.

[12] Chumki Basu, Haym Hirsh, and William W. Cohen, “Recommendation as classification: Using social and content-based information in recommendation,” in *Proceedings of the Fifteenth National Conference on Artificial Intelligencen AAAI/IAAI*, 1998, pp. 714–720.

[13] T. Zhang and V. S. Iyengar, “Recommender systems using linear classi-

fiers," *Journal of Machine Learning Research*, vol. 2, pp. 313–334, 2002.

[14] D. Heckerman, D.M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie, "Dependency networks for inference, collaborative filtering, and data visualization," *Journal of Machine Learning Research*, vol. 1, pp. 49–75, 2000.

[15] T. Hofmann and J. Puzicha, "Latent class models for collaborative filtering," in *Proceedings of IJCAI'99*, 1999, pp. 688–693.

[16] W.S. Lee, "Collaborative learning for recommender systems," in *Proc. 18th International Conf. on Machine Learning*, 2001, pp. 314–321.

[17] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of 10th World Wide Web (WWW10) conference*, Hong Kong, 2001, pp. 285–295.

[18] W. Lin, S.A. Alvarez, and C. Ruiz, "Collaborative recommendation via adaptive association rule mining," *Data Mining and Knowledge Discovery*, vol. 6, no. 1, pp. 83–105, Jan 2002.

[19] D. M. Pennock, E. Horvitz, S. Lawrence, and C.L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach," in *Proc. of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000, pp. 473–480.

[20] N. Good, J.B. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl, "Combining collaborative filtering with personal agents for better recommendations," in *Proceedings of AAAI-99*, 1999, pp. 439–446.

[21] A.M. Rashid, I. Albert, D. Cosley, S.K. Lam, S.M. McNee, J.A. Konstan, and J. Riedl, "Getting to know you: Learning new user preferences in recommender systems," in *Proceedings of International Conference on Intelligent User Interface (IUI2002)*, San Fransisco, CA, 2002.

[22] J.L. Herlocker, J.A. Konstan, and J. Riedl, "Explaining collaborative filtering recommendations," in *Proceedings of computer supported cooperative work (CSCW'00) conference*, 2000, pp. 241–250.

[23] Craig Boutilier and Richard S. Zemel, "Online queries for collaborative filtering," in *Ninth International Workshop on Artificial Intelligence and Statistics*, 2003.

[24] Finn V. Jensen, *Bayesian Networks and Decision Graphs*, Statistics for Engineering and Information Science. Springer, 2001.

[25] D. Heckerman, J. Breese, , and K. Rommelse, "Troubleshooting under uncertainty," Tech. Rep. MSR-TR-94-07, Microsoft Research, 1994.

[26] Simon Tong, *Active Learning: Theory and Applicaitons*, Ph.D. thesis, Stanford University, 2001.

[27] D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Proceedings of the Eleventh International Conference on Machine Learning*. 1994, pp. 148–156, Morgan Kaufmann.

[28] Thomas Cover and Joy Thomas, *Elements of Information Theory*, Wiley, 1991.

[29] George Fishman, *Monte Carlo Concepts, Algorithms and Applications*, Springer Verlag, 1996.

[30] Bernhard Schölkopf and Alex J. Smola, *Learning with Kernels*, MIT Press, 2002.

[31] J.L. Herlocker, J.A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering," in *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR'99)*. 1999, pp. 230–237, ACM.

[32] Koji Miyahara and Michael J. Pazzani, "Collaborative filtering with the simple bayesian classifier," in *Proceedings of the Sixth Pacific Rim International Conference on Artificial Intelligence PRICAI 2000*, 2000, pp. 679–689.

[33] Kai Yu, Anton Schwaighofer, Volker Tresp, Wei-Ying Ma, and HongJiang Zhang, "Collaborative ensemble learning: Combining collaborative and content-based information filtering," Submitted to UAI 2003, Apr. 2003.

Kai Yu is a PhD student in the Institute for Computer Science at the University of Munich. His research work is supported through a scholarship from Siemens Corporate Technology in Munich. He has been working in speech separation, noise reduction, information retrieval and data mining. Currently his research interests are mainly focused on statistical machine learning and its applications in data mining, information and image retrieval, and medical data analysis. He received a BS and an MS in 1998 and 2000, respectively, from Nanjing University, China.

Anton Schwaighofer received his MSc degree in computer science from Graz University of Technology, Austria, in 2000. He is currently a PhD student at Graz University of Technology in co-operation with Siemens Corporate Technology in Munich. He has been working in pattern recognition for biometric applications and medical diagnosis systems. His major research interests are kernel based learning systems, in particular Gaussian processes for large scale regression problems, graphical models and clustering methods.

Volker Tresp received a Diploma degree in physics from the University of Göttingen, Germany, in 1984 and the MSc and PhD degrees from Yale University, New Haven, CT, in 1986 and 1989, respectively. He joined the central research and development unit of Siemens AG in 1989 where he currently is the head of a research team. In 1994 he was a visiting scientist at the Massachusetts Institute of Technology's Center for Biological and Computational Learning. His main interests include learning systems, in particular neural networks and graphical models and medical decision support systems.

He has published papers on various topics including the combination of rule-based knowledge and neural networks, the problem of missing data in neural networks, time-series modeling with missing and noisy data, committee machines, learning structure in graphical models, and kernel based learning systems. He is co-editor of Neural Information Processing Systems, 13.

Xiaowei Xu joined the Department of Information Science, University of Arkansas at Little Rock as an Associate Professor, from Corporate Technology, Siemens AG, Munich, Germany in 2002. His specialty is in data mining and database management systems. His recent research interests focus on text mining, database systems for biological and medical applications, and multimodal information retrieval. Dr. Xu has been active member of ACM. He obtained his PhD from the University of Munich in 1998.

Hans-Peter Kriegel is a full professor for database systems in the Institute for Computer Science at the University of Munich. His research interests are in spatial and multimedia database systems, particularly in query processing, performance issues, similarity search, dimensional indexing, and in parallel systems. Data exploration using visualization led him to the area of knowledge discovery and data mining. Kriegel received his MS and PhD in 1973 and 1976, respectively, from the University of Karlsruhe, Germany. Hans-Peter Kriegel has been chairman and program committee member in many international database conferences. He has published over 200 refereed conference and journal papers. In 1997 Hans-Peter Kriegel received the internationally prestigious "SIGMOD Best Paper Award 1997" for the publication and prototype implementation "Fast Parallel Similarity Search in Multimedia Databases" together with four members of his research team.