MEDIATION OF USER MODELS FOR ENHANCED PERSONALIZATION IN RECOMMENDER SYSTEMS

SHLOMO BERKOVSKY

May, 2009

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisors: Francesco Ricci, Tsvi Kuflik, Oliviero Stock, and Martin Golumbic. Their thoughts, ideas, suggestions, guidance, support, motivation, feedback and encouragement, contributed greatly to the successful ending of this work and enriched me substantially. THANK YOU!!

I would like to thank the Haifa-Trento Scientific Collaboration, Caesarea Edmond Benjamin de Rothschild Foundation Institute for Interdisciplinary Applications of Computer Science, and the Authority of Graduate Studies for providing financial support for this work.

Last but not the least; I would like to thank my family and friends for being there for me, for believing in me, and for supporting me all over this long road. Very special thanks to my wife and kids for being my constant encouragers and motivators. Their care and love has been a source of inspiration for me.

Shlomo.

Table of Contents

List of Figures

List of Tables

Part 1:

Introduction

Chapter 1: Introduction

1.1 Research Motivation

During the last decades, the quantity of potentially interesting products or information services available online has been growing rapidly and now exceeds human processing capabilities [Maes 1994]. This has lead to various information search situations, where the users would like to choose among a set of alternative items, services, or information items, but do not have sufficient knowledge, capabilities, or time to make such decisions. For example, consider the number of news items uploaded every minute by the news agencies, or the wide variety of products in E-Commerce Web-site, the number of posts written in Web-logs, and so forth. This trend is referred to in the literature as the *information overload* problem [Hiltz and Turoff 1985; Nelson 1994; Maes 1994].

As such, there is a pressing need for intelligent systems that advise users while taking into account their personal needs and interests. Such systems can deliver tailored services in a way that will be most appropriate and valuable to the users. This type of system is referred to in the literature as a *personalization* system [Mulvenna et al. 2000]. Several types of personalization approaches are exploited in practical personalization systems. For example, *information retrieval* systems, i.e., search engines [Das et al. 2007], allow the location of the information explicitly searched for the users. *Information filtering* systems [Hanani et al. 2001] reduce information overload by filtering out irrelevant information on behalf of their users. Similarly, *recommender systems* advise their users about the items they might wish to purchase or examine from a larger set of available items [Burke 2000]. We would like to stress that information filtering systems and recommender systems are considered complementary approaches. Both of them are aimed at reducing the user's information overload by limiting the amount of information reaching the user. Information filtering systems achieve this by filtering out all the irrelevant information from the user's incoming information stream, while recommender systems achieve this by selecting the relevant information from the user's incoming information stream and displaying only it. This work refers to recommender systems [Resnick and Varian 1997] and various recommendation

approaches exploited in recommender systems, as representative examples of personalization systems.

Extensive research of recommender systems started over a decade ago and yielded a wide variety of recommendation techniques, exploited in numerous practical systems. These techniques include content-based filtering [Morita and Shinoda 1994], collaborative filtering [Resnick et al. 1994], knowledge-based recommendation [Burke 2000] and utility-based recommendation [Manouselis and Sampson 2004] and their multiple hybridizations [Burke 2002]. They are widely discussed in the literature and in several surveys of the state-of-the-art recommender systems [Schafer et al. 2001; Montaner et al. 2003; Adomavicius and Tuzhilin 2005c].

Whatever the specific technology exploited by a recommender system, it can provide high quality recommendations to the users only after having modeled their preferences. This information is typically referred to in the literature as the User Model (UM) [Kobsa 2001a]. The task of collecting the user modeling data is typically performed in two ways: (1) explicit – through provision of the required information explicitly by the user, or (2) implicit – through applying various reasoning mechanisms that infer the required information based on the user's observable behavior [Hanani et al. 2001]. The explicit collection of user modeling data is considered to be an accurate, but time- and effort-consuming task, typically avoided by the user. Alternatively, the implicit collection involves automated reasoning mechanisms, which can misinterpret user behavior. In practice, both explicit and implicit approaches may be combined [Kuflik et al. 2007].

In general, the quality of the recommendations provided to the user depends largely on the characteristics of the UM, e.g., how accurate it is, what amount of information it stores and how this can be actually exploited, and whether this information is up to date. Hence, as a general rule, the more information is stored in the UM, i.e., the more knowledge the system has obtained about the user, the better the quality of the recommendations will be[1]. In this context, quality refers to the capability of the system to suggest exactly those products, services and information items that the user will select and purchase, or to predict correctly those items that the user would like. In practice, obtaining sufficient user modeling data to deliver high quality recommendations is difficult.

[1] In some practical cases this claim is false. For example, if a system stores too much irrelevant, outdated or imprecise data, or controversial or ambiguous data about the user, this may hamper provision of accurate recommendations.

This is especially important at the initial stages of interaction with the user, when little information about the user is available. At these stages, all the existing recommendation techniques face the bootstrapping (also called cold-start) problem, i.e., a situation where the available information about the user and/or items does not suffice to provide high quality recommendations [Linden et al. 2003].

When analyzing current recommender systems, one can see that typically, every system collects and maintains a proprietary collection of UMs [Montaner et al. 2003]. Practically, this means that the collected user modeling data are tailored to: 1) the specific content (products or products categories) offered by the recommender system, e.g., movies [Good et al. 1999], music [Aguzzoli et al. 2002], news items [Claypool et al. 1999], tourism [Ricci et al. 2006c], and so on, and 2) the recommendation technique being exploited by the system, e.g., collaborative filtering [Herlocker et al. 1999], content-based filtering [Morita and Shinoda 1994], demographic filtering [Krulwich 1997], or some of their hybridizations [Burke 2002]. Thus, a large amount of heterogeneous (and possibly overlapping) user modeling data are scattered among various systems.

In general, practical recommender systems (especially commercial ones) neither allow other external recommender systems to access them, nor share their proprietary user modeling data. However, it can be hypothesized that recommender systems could benefit from enriching their user modeling data by importing and integrating user modeling data collected by other, possibly related, systems, and therefore provide better recommendations to their users.

1.2 Mediation of User Modeling Data in Recommender Systems

User modeling data integration can be achieved through a process that is referred to as *mediation* of UMs and other user modeling data [Berkovsky et al. 2006a; Berkovsky et al. 2008]. The term 'mediation' was coined in this work and it is defined as "*a process of importing and integrating the user modeling data collected by other recommender systems for the purposes of a specific recommendation task*". Hence, the primary goal of the mediation is to instantiate the UMs through inferring the required user modeling data from other data imported from other systems. The mediation enriches the existing UMs (or bootstraps empty UMs) in the target recommender systems and, as a result, facilitates provision of better recommendations.

Let us introduce an example mediation scenario in the realm of digital entertainment. Let us consider a network of Web-sites providing personalized entertainment recommendations. The network includes music, movies, TV programs, books and humor recommender systems. Also, consider a user requesting a movie recommendation from the movies recommender system. Although the movies recommender system collected certain user modeling data about the user in the past, these may not be sufficient for providing high quality recommendations. To enhance the recommendations, the movies recommender system can obtain more accurate UMs by importing and integrating user modeling data collected by other systems. For example, such an information can be the list of user's favorite art genres, which can be mined from her favorite TV programs stored in the UM collected by TV programs recommender system or from the books that were recommended by books recommender systems and later purchased by the user, or the list of user's favorite composers, which can be mined from the CDs purchased by the users stored in the UM collected by music recommender system. These data can be used, for example, for recommending a movie of the favorite genre, where the soundtrack music was composed by the favorite composer.

The idea of UM mediation presents a number of challenges. The first challenge refers to the nature of the information market, and its business models. As a result of today's commercial competition between practical real-life recommender systems, the systems typically neither cooperate nor share their user modeling data. In [Adomavicius and Tuzhilin 2005b], the authors point out that typical recommender systems are either provider-centric (i.e., each provider has its own recommendation engine to tailor its content to consumers) or market-centric (i.e., providing recommendations for a specific marketplace in a particular industry or sector). The authors claim that the lack of technical data-sharing solutions in the existing recommender systems is mainly explained by business limitations imposed on the exchange of user-related information among competing parties in the same market.

The second challenge refers to guaranteeing user privacy. UMs collected by a certain recommender system may contain private and sensitive information, that the users would not like to be disclosed to other systems, and possibly to untrusted parties [Cranor et al. 1999]. For this reason, many recommender services that store sensitive information about their users declare in their pri-

vacy policies that no personal information stored by the system will be transferred to other parties under any circumstances [Wang and Kobsa 2006]. As a result, they are committed not to transfer information stored in their UMs to other systems and, therefore, the possibilities of applying the above mediation scenario are limited.

The third challenge refers to practical and technical considerations of the mediation. For example, in a distributed setting various recommender systems need to connect to each other through slow, inherently unreliable, and error-prone communication middleware. Due to various connectivity limitations, certain recommender systems (e.g., those running on personal and mobile devices) may be partially available online, or their communication throughput may be limited. In some cases, the mediation may require time-consuming data processing by one of the systems, which may prevent the provision of real-time personalized recommendations.

The fourth challenge refers to the structural heterogeneity and incompleteness of the user modeling data. As mentioned earlier, current recommender systems usually refer to specific application domains, and the services are provided using specific recommendation techniques, which imply specific UM representations. The lack of a standard representation for the UMs and specific storage and access requirements imposed by various recommendation techniques, result in a situation where various systems collect user modeling data in different ad-hoc forms. This heterogeneity causes several problems: various techniques may store the preferences of the same user in different forms; the information in various systems may be conflicting or outdated, and may be influenced by various cross-lingual and cross-culture dependencies; and so on. All these heterogeneities aggravate the mediation task, since it must support not only the integration of user modeling data, but also resolution of inconsistencies and conflicts among the data obtained from various systems [Francisco-Revilla and Shipman 2004].

This work focuses on resolving only the latter challenge – the heterogeneity of the available user modeling data. Although this work stresses the importance of the other challenges, practically it refers only to certain aspects of the privacy challenge and to the technical challenge of user modeling data storage and retrieval in a distributed environment. However, the reader is referred to [Rabanser and Ricci 2005] for a discussion of integrated business models in recommender systems, to [Kobsa 2007] for an extensive discussion of privacy-preserving approaches in recom-

mender and personalization systems, and to [Han et al. 2004; Ruffo and Schifanella 2007] and [Breese et al. 1998; Goldberg et al. 2001] for discussions on, respectively, distributed decentralized recommender systems and optimized heuristic variants of the collaborative filtering.

In summary, the main contribution of this work is in developing a general abstraction mechanism and some precise mediation methods for aggregating and integrating user modeling data in a distributed environment, which facilitates better interoperability of recommender systems and provision of better recommendations to the user. This contribution is two-fold: (1) from the user modeling perspective it establishes a novel approach for building more accurate UMs by integrating user modeling data collected by a set of distributed recommender systems, while (2) from the recommender systems perspective it provides a basis for a novel hybrid recommendation technique, where the recommendation generation process is based on the information coming from multiple sources of user modeling data. Hence, this work can be considered as the first work that formalizes and experimentally evaluates interoperability of recommender (and personalization) systems through sharing of user modeling data. It can be naturally extended in various research directions, such as more extensive evaluation of the mediation, implementing and evaluating cross-context mediation, implementing the mediation between practical recommender systems, developing information exchange model, and many others.

1.3 Organization of the Book

This work is divided into four Parts, which are further divided into Sections. Part I is an introductory part, providing a motivational discourse and survey of the related research directions. Section 1 describes the motivation behind the work and briefly introduces the proposed solution of the problem of mediation of user modeling data. Section 2 provides the background material by surveying numerous research works in the domains of recommender systems and user modeling, and can be skipped by experienced readers.

Part II provides the main contribution of this work from the theoretical point of view. It presents and extensively discusses the idea of user modeling data mediation and several practical mediation approaches. Section 3 presents a general user modeling data representation and describes its several possible instantiations in the existing recommendation techniques. Section 4 elaborates on

the mediation of user modeling data as a means for enriching the user modeling data available to the target recommender system. In particular, four types of mediation are presented: cross-representation, cross-user, cross-item, and cross-context mediations. Both Sections 3 and Section 4 are based on the conference-level publication [Berkovsky 2006a] and the follow-up extended journal publication [Berkovsky et al. 2008].

Part III presents two implementations and evaluations of the UM mediation. Section 5 presents a cross-technique variant of cross-representation mediation, where the user modeling data are converted between collaborative filtering and content-based recommender systems. As such, it demonstrates a practical scenario, where the data collected by one recommender systems is shared with and exploited by another system. Section 5 is based on the conference-level publication [Berkovsky et al. 2006b] and the follow-up extended journal publication [Berkovsky 2009a]. Section 6 presents cross-domain mediation, a generalized variant of cross-item mediation, where collaborative filtering user modeling data from several application domains are imported and several practical mediation scenarios are defined and evaluated. This demonstrates a more complicated mediation scenario, where several types of user modeling data mediation are shared between recommender systems. Section 6 is based on the conference-level publications [Berkovsky et al. 2007a] and [Berkovsky 2007b].

Part IV presents some practical aspects of the mediation. Two challenges among those mentioned in Section 1.2, are studied in depth. Section 7 present a decentralized distributed model for storage, access and retrieval of heterogeneous user modeling data with no central ontology, such that various data providers may represent their data in different ways. Section 7 is based on the initial conference-level publication [Berkovsky et al. 2005b] and the follow-up extended journal publication [Berkovsky 2009b].Section 8 deals with the privacy challenges of UM mediation in collaborative filtering recommender systems and proposes several data modification methods that can be applied for the obfuscation of sensitive parts of collaborative filtering UMs. Section 6 is based on the conference-level publications [Berkovsky et al. 2005a], [Berkovsky et al. 2006d], and [Berkovsky 2007c].

Finally, Part V and Section 9 conclude and summarize this work. It presents the conclusions that can be drawn from this work, summarizes its research contributions, and proposes several possi-

ble directions for future research, which can be natural extensions of this work. Section 9 is based on the conclusions and on the future research directions presented in [Berkovsky et al. 2008].

Chapter 2: Background and Related Works

Recent studies showed that the world's total yearly production of information is more than 1.5 billion gigabytes [Lyman and Varian 2003]. Also the amount of available information on the Web is huge: there are billions of Web-pages, and millions of newsgroups and forums, which are constantly expanding; another study estimates that the Web grows significantly every single day [Smyth 2005]. This mass of available online information results in an increasing difficulty in finding relevant information in which users are really interested. This issue is referred to in the literature as the *information overload* problem [Hiltz and Turoff 1985; Nelson 1994; Maes 1994]. Initially, information overload was defined as "information presented at a rate too fast for a person to process" [Sheridan and Ferrell 1974]. In the context of Web, information overload might cause the users to fail in locating the required information, or to overlook information that would be considered important in different conditions, or simply to evaluate the available information and select the most appropriate one.

As a result, there is a pressing need for personalization systems (i.e., intelligent systems capable of providing services according to the user's personal needs and interests), and delivering information tailored in such way that will be most appropriate and valuable to the user [Brusilovsky et al. 2007]. In particular, [Good et al., 1999] names three main technologies that are commonly used to address the information overload problem:

- *Information Retrieval* [Chen and Sycara 1998; Brin and Page 1998] – focusing on tasks involving a fulfillment of ephemeral interest queries.
- *Information Filtering* [Belkin and Croft 1992; Hanani et al. 2001] – filtering streams of incoming information and extracting the relevant information from the incoming stream according to a set of predefined criteria.
- *Recommender Systems* [Resnick et al. 1994; Resnick and Varian 1997] – trying to assist users in the selection of available items by predicting the level of interest of a particular user in a given item and suggesting the most valuable items only.

All the above technologies are aimed at overcoming the information overload by finding information that might interest a user and/or filter out the undesirable information reaching the user. Thus, the techniques exploited by these technologies partially overlap and many practical applications can be classified to several technologies.

Information retrieval applications, e.g., search engines [Das et al. 2007], assist users in locating information based on explicit queries, by which the users represent their ephemeral information needs. Because of the vast amount of available information and relatively simple keyword matching algorithms, search engines usually return either thousands or very few answers [Jansen et al. 1998]. Furthermore, their capability is inherently limited to retrieval of textual documents, as they are unable to retrieve other types of information such as audio, video and multimedia files. *Information filtering* applications reduce information overload by filtering out irrelevant information. These systems collect information about the users, infer and build user profiles, representing their interests and preferences, and use them to filter out the irrelevant pieces of information reaching the users. *Recommender systems* applications advise their users about items they might wish to purchase or examine from a larger set of available items [Burke 2000]. Also these applications maintain profiles of users' interests and needs, and apply statistical and knowledge discovery techniques aimed at selecting the most appropriate items [Sarwar et al. 2000]. Hence, these applications aggregate input recommendations provided by various users and redirect them to the most appropriate recipients [Resnick and Varian 1997].

We would like to stress the observation that, in the sense of operational functionality, recommender and information filtering applications are considered complementary approaches [Hanani et al. 2001]. Both are aimed at reducing the information overload: recommender systems limit the amount of information reaching the user by selecting and displaying relevant information only, while information filtering applications achieve the same by filtering out the irrelevant information from the user's incoming information stream. Since this work deals with addressing the information overloading in recommender systems, future discussion will focus on these applications.

2.1 Recommender Systems

Recommender systems mainly use past opinions of a user or a community of users to help individual users to identify items of interest effectively from an overwhelming set of choices. It is important to mention the most popular end-user goals and tasks, for which recommender systems are being used, as expressed by the users themselves [Herlocker et al. 2004]:

- Annotation in context – annotation and grading of potentially selectable items (e.g., email messages, news items, and so forth).
- Finding good items – suggesting a specific set of items, most valuable for the users.

Although intensive research of recommender systems was initiated by [Resnick and Varian 1997], the algorithms exploited in the state-of-the-art recommender systems were proposed earlier in [Malone et al. 1987]. That work referred to the process of removing irrelevant information reaching the user and proposed two basic recommendation techniques: (1) cognitive filtering, which is nowadays referred to in the literature as *content-based filtering* [Morita and Shinoda 1994], and (2) social filtering, which is nowadays referred to as *collaborative filtering* [Resnick et al. 1994]. Later, other recommendation techniques, which are currently considered basic techniques, were added: *knowledge-based recommendations* [Burke 2000] and *demographic filtering* [Krulwich 1997]. Finally, [Burke 2002] compared the advantages and shortcomings of the above techniques and discussed the possible methods for their hybridization. Experiments and comparisons of *hybrid recommender systems*, described in [Burke 2002], validate a hypothesis of prior works that hybridization between recommendation techniques can improve the accuracy of the recommendations provided to the users.

Since the beginning of research on recommender systems over a decade ago, this domain has grown substantially. Currently, there are many practical recommender systems providing personalized recommendations in various application domains. We will mention only a small list of domains:

- Movies [Good et al. 1999] – combines collaborative filtering recommendation techniques and information filtering agents to identify movies that a user would find worthwhile.
- TV programs [Dai and Cohen 2003] – combines content-based and collaborative filtering recommendation techniques to build a pseudo-user profile, which answers the requirements of a group of users to which the current user belongs.
- Music [Aguzzoli et al. 2002] – uses case-based reasoning recommendation techniques to build and recommend music compilations for the users.
- Newspaper articles [Claypool et al. 1999] – combines content-based and collaborative filtering recommendation techniques to filter out the articles that would not interest a user from an online newspaper.

- News items [Resnick et al. 1994] – uses collaborative filtering recommendation techniques to recommend unseen news items that will interest her.
- Academic courses [Farzan and Brusilovsky 2006] – applies a hybridization of collaborative filtering and content-based recommendation techniques to recommend academic courses to graduate students.
- Humor [Goldberg et al. 2001] – uses collaborative filtering recommendation techniques to recommend jokes to the users.
- Food recipes [Svensson et al. 2000] – uses a stereotype-based recommendation techniques to generate food recipe recommendation in an online grocery store.
- Travel routes and plans [Ricci et al. 2003] – uses a hybrid case-based reasoning recommendation techniques to suggest a user tourist site he would enjoy visiting.
- Museum exhibits [Kuflik et al. 2007] – applies content-based filtering recommendation techniques to recommend a user exhibit based on feedback on previous exhibits.

Other domains are presented in a comprehensive survey of commercial recommender systems applications [Schafer et al. 2001], while a more updated listing and classification of the existing recommender systems can be found in [Montaner et al. 2003] and [Adomavicius and Tuzhilin 2005c].

The following subsections briefly overview the details of four recommendation techniques: content-based filtering, collaborative filtering, knowledge-based recommender systems, and demographic filtering. These techniques (and some variants of their hybridizations) constitute the basis of the state-of-the-art recommender systems and various online applications using recommendation techniques for the purpose of providing personalized services to the users.

2.1.1 Content-Based Recommender Systems

The content-based recommendation approach represents the items of interest by their associated features and the interests of the users as a basis for recommendations [Morita and Shinoda 1994; Oard 1997]. For example, consider a news items recommender system, where the words mentioned in the articles represent the content attributes of the items. The simplest content-based recommender systems are basically keyword-based information retrieval systems. For example, [Burke et al. 1996a] describes a natural language question answering system that builds a knowledge base of the most frequently asked questions, including their keywords and terms, and rec-

ommends the users answers to queries based on the terms appearing in the queries. Another example of content-based recommender systems is presented in [Mirzadeh et al. 2005]. That work describes a system that iteratively asks the users to refine their queries by providing additional content features describing the items being searched. The refined queries are searched again in the system's repository, and the recommendations are generated by the system through this iterative and interactive query refinement process.

Content-based recommender systems typically learn the user profiles based on the features of objects, which were rated by the user in the past. Most of the commonly used machine learning techniques can be exploited in order to learn the user profiles. Among others, they include decision trees [Kim et al. 2002; Gmytrasiewicz et al. 1998], neural networks [Alvarez et al. 2007], genetic algorithms [Desjardins and Godin 2000], and other techniques. The profiles used in content-based recommender systems are typically long-term profiles [Burke 2002], i.e., they reflect the stable parts of users' preferences. To keep providing accurate and up-to-date recommendations, the profiles are continuously updated, as more evidence about the user's preferences is observed from the user's feedback on items.

We would like to stress that content-based filtering is limited to recommending only items with contents previously encountered and rated by the user. For example, when a user rated only items from a certain application domain, only items from this domain can be recommended in the future (e.g., if a user rated movies from a particular genre only, the system cannot surprise her by proposing movies from other genres). This problem is referred to in the literature as the *serendipity problem* [McNee et al. 2006] of content-based recommender systems. Moreover, since the state-of-the-art (except the obvious keyword-based search tools) content-based recommender systems acquire accurate list of users' interests, they usually require time for training before being able to produce high quality recommendations. This problem is referred to in the literature as a *new user problem* [Linden et al. 2003] and is considered a particular case of a general *sparsity problem* [Schein et al. 2002], where the amount of available information is not sufficient for generating accurate recommendations.

2.1.2 Collaborative Filtering Recommender Systems

Collaborative filtering is probably one of the most popular and widely-used recommendation techniques. It is based on the notion of *automating word of mouth* presented in [Shardanand and Maes 1995], which assumes that people who agreed in the past (i.e., their ratings regarding a certain set of objects correlated), will also agree in the future. In other words, it uses opinions of similar users to generate future predictions for the user. It differs from the content-based filtering in the sense that opinions of other users and not only the opinions of the user who requested a recommendation are used as input for the recommendation generation.

The opinions of the users on the available items are represented by the *ratings matrix*, where each item is represented by a set of users' opinions and each user is represented by her opinions on the items (referred to in the literature as the *ratings vector*). These opinions can be expressed either as explicit ratings given by the user according to a predefined scale that ranges from 'bad' to 'good', or as implicit ratings accumulated and inferred through logging user's interaction with the system. For example, consider a Web-pages recommender system, where the users can provide explicit ratings on visited pages on a discrete scale from *1* to *5* stars. Alternatively, the system may implicitly acquire opinions of the user, where the actions of viewing or skipping certain Web-pages are interpreted as, respectively, positive and negative ratings.

The main stages of the collaborative filtering recommendation generation process are defined as follows [Herlocker et al. 1999]: (1) recognizing commonalities between users by computing similarities of their rating vectors; (2) selecting a subset of the most similar users; and (3) generating recommendations by aggregating the opinions of the most similar users. Due to its social origin based on the similarities of users, collaborative filtering process is sometimes called 'people-to-people correlation' [Schafer et al. 1999]. Some of the important early systems using this technique were *GroupLens* for filtering of news articles [Resnick et al. 1994], *Ringo* for recommending music albums and artists [Shardanand and Maes 1995], *Tapestry* for filtering of the incoming stream of emails [Goldberg et al. 1992], and *Recommender* for movie recommendations [Hill et al. 1995].

One of the stages of the collaborative filtering recommendation process deals with identifying the set of the most similar users, which is also referred in the literature as "the neighborhood forma-

tion stage [Herlocker et al. 1999]. This is usually performed by applying similarity metrics on the available users and selecting either all the users whose similarity is above a certain threshold or a set of *K* most similar users. The most popular similarity metrics (more can be found at [Breese et al. 1998]) are:

- *Pearson Correlation*. According to this metric, the similarity between the profile P_x of user *x*, and the profile P_y of user *y* is computed by:

$$cor(P_x, P_y) = \frac{\sum_i (P_{xi} - P_x^{'})(P_{yi} - P_y^{'})}{\sqrt{\sum_i (P_{xi} - P_x^{'})^2 \sum_i (P_{yi} - P_y^{'})^2}}$$

  where $P^{'}_a$ represents the average of the ratings vector of user *a*, whose cardinality is $|P_a|$, i.e.,

$$P_a^{'} = \frac{\sum_{i=1}^{|P_a|} P_{a,i}}{|P_a|}$$

  A negative $cor(P_x,P_y)$ indicates a dissimilarity of users, while a positive $cor(P_x,P_y)$ indicates their similarity. When $cor(P_x,P_y)$ is equal or close to *0*, no significant correlation between the users can be inferred. This metric was used in [Pennock et al. 2000a] and in [Sarwar et al. 2000].

- *Cosine Similarity.* This metric defines the similarity between two users by computing the cosine of the angle between their profiles in a multi-dimensional space:

$$cor(P_x, P_y) = \frac{P_x \cdot P_y}{\| P_x \|_2 \times \| P_y \|_2}$$

  where · denotes the inner product between the relevant profile vectors, and $\|P_a\|_2$ denotes the norm of a vector, i.e., the square root of the inner product of a vector with itself. This metric was used in [Good et al. 1999] and in [Sarwar et al. 2001].

- *Mean Squared Difference*. Basically, this metric computes the degree of dissimilarity between two users. The mean squared differences between profile P_x of user *x*, and the profile P_y of user *y* is computed by:

$$cor(P_x, P_y) = \frac{\sum_{i=1}^{|P_x|} (P_{xi} - P_{yi})^2}{|P_x|}$$

  where $|P_a|$ denotes the cardinality of the profile vector of user *a*. The lower is the result of the mean squared difference computation, the greater is the similarity between the users. This metric was used in [Shardanand and Maes 1995] and in [Pennock et al. 2000a].

In comparison with content-based recommendation techniques, the main advantage of collaborative filtering is its independence of any representation of users and items. Hence, collaborative filtering systems can generate recommendations for items regardless of their contents: recommendations for movies, images, and text documents can be generated in a similar manner by a single system. As such, collaborative filtering is considered a universal technique, capable of providing recommendations for items from different domains. Collaborative filtering recommender systems suffer from two sparsity problems that should be stressed: (1) *new user problem* – the number of ratings of a user is insufficient for a reliable similarity computation [Linden et al. 2003], and (2) *new item problem* – the number of ratings on an item is insufficient for a reliable generation of recommendations [Gokhale and Claypool 1999]). Another drawback of collaborative filtering systems is their non-dynamism and insensitivity to 'short-term needs' of users [Hayes and Cunningham 2003]. Although the ratings are collected over time, the sparsity of the data requires taking all the available ratings into account. As a result, the recommendations are sometimes not accurate since they are based on outdated ratings, deterring the users from using the system. This makes collaborative filtering systems too coarse-grained when the recommendation must be tailored to specific short-term needs.

2.1.3 Knowledge-Based Recommender Systems

Knowledge-based recommender systems attempt to suggest items based on inference about user's needs and preferences. In other words, knowledge-based systems comprise *functional knowledge*, i.e., they have a-priori general knowledge about the matching of certain items to certain user needs. Hence, they can generate recommendations through reasoning about the relationship between a given need and a set of possibly recommended items. The above functional knowledge should be defined in advance by the developers of the system, and therefore knowledge-based recommender systems are referred as the "editor's choice" method [Schafer et al. 1999].

In knowledge-based systems, user profiles can be represented by any structure that supports the required inference. In the simplest case, when no information about the user is available, the user's profile can be neglected and the user's behavior during the recommendation session (i.e., queries launched, viewed results, query modifications and so forth) serves as the only basis for the recommendations. However, most knowledge-based systems store detailed representations of users' needs and a history of transactions between the users and the system. This history of past

interactions is typically represented as an iterative process: (1) the user launches a query describing the item interesting her; (2) the system identifies the items that satisfy the query and (3) represents the items in a certain order reflecting the user needs; (4) the user browses the suggested items and if they do not satisfy her need, she (5) modifies the query to refine her needs.

Several knowledge-based recommender systems were described in the literature. For example, [Burke et al. 1996b] describes a recommender system, which supports navigational search for the item in which the user is interested. This system stores a user profile as a series of searches, their results and modifications inserted by the user to refine the search. Similarly, [Schmitt and Bergmann 1999] stores detailed characteristics of the items, viewed by the user within her interaction with the system. In addition to the item characteristics, [Towle and Quinn 2000] proposes storing the user profiles also the information that describes the causalities of their behavior.

Conversational recommender systems [Linden et al. 1997] have become a widely-used and intensively studied type of knowledge-based recommender system. A very early example of a conversation student course advising recommender system was proposed in [Golumbic et al. 1986]. In that work, student preferences for courses were (1) automatically generated according to the formal degree requirements of the students matched against their transcripts of courses taken and grades received, and (2) explicitly modified or specialized by the students. Preferences for time schedules were also provided by the students, and matched by the system to the courses recommended. Further extensions of the system were presented in [Golumbic and Feldman 1990] and [Golumbic et al. 1991].

Unlike in 'single shot' recommender systems, the users of conversational systems iteratively refine their requirements and manage a dialog with the system. After each query, the system presents to the user a set of recommended items satisfying the query, and several ways to refine the query, called critiques [Reilly et al. 2004]. Hence, conversational recommender systems iteratively guide the user by recommending certain items and exploit user's critiques to improve the following recommendations [Smyth et al. 2004; Zhang and Pu 2006]. Due to the above iterative refinement of user queries, conversational recommender systems can more easily adapt their recommendations to the user's short-term needs.

The main advantage of knowledge-based recommender systems is that they are not affected by the *sparsity problem*. Since the recommendation process is based on a-priori functional knowledge, and not on users' ratings, the system is capable of producing accurate recommendations even in the early stages. However, this also introduces the major drawback of knowledge-based systems, as they rely on design knowledge inserted by the system developers. As such, knowledge-based systems are expensive to build and difficult to maintain. Therefore, their domains are typically limited to a relatively small set of pre-coded topics only, and they cannot be easily adapted to the specific needs of individual users or to new application domains [Burke 2000].

2.1.4 Demographic Recommender Systems

Demographic filtering recommender systems aim at categorizing the users using their personal demographic attributes, and generating recommendations based on the demographic classes. An early example of demographic filtering system was described in [Rich 1979b] that produced book recommendations using personal information gathered through an interactive dialogue, whereas the responses of the users were matched against a library of manually assembled user stereotypes. Conversely, [Krulwich 1997] used the data gathered on demographic groups in marketing research to suggest a range of products and services. Demographic-based systems usually exploit machine learning methods to train a classifier based on the available demographic data about the users [Pazzani 1999].

Although demographic filtering, similarly to the collaborative filtering, uses "people-to-people correlation", the advantage of the former is that it does not require the history of user ratings that is required by the latter. Hence, demographic filtering systems are not affected by the data *sparsity problem*. However, demographic filtering systems require demographic data, which are considered private and sensitive, not commonly available, and are typically difficult to collect. Information sharing surveys showed that the users are typically reluctant to share their private demographic information, such as phone numbers, physical addresses and so forth [Cranor et al. 1999]. Another issue that prevents wide use of demographic systems is their relative unreliability with respect to the accuracy of the generated recommendations. In many cases, demographic similarity of users is not representative enough to imply their similarity in various application domains (consider demographic similarity versus the similarity of tastes in music, movies, and other domains). As a result, the generated recommendations may be inaccurate, and demographic recom-

mender systems are relatively infrequent in comparison to the systems exploiting the previous techniques.

2.1.5 Comparison of Recommendation Techniques

The above mentioned recommendation techniques differ in the input they require, the functional knowledge required for recommendation process, and the underlying algorithms. Table 1 (adapted from [Burke 2002]), summarizes the above mentioned recommendation techniques. The following notation is used in the table: *I* is the set of items for which recommendations might be made, whereas *U* is the set of users whose preferences and interests are known. The ultimate goal of the system is to predict the level of interest (or, simply to provide a recommendation) of a user *u*, who is referred to in the literature as the *active user*, in an item *i*.

Table 1: Summary of Recommendation Techniques

Technique	Required Knowledge	Input	Filtering Algorithm
Content-based	Features of items in *I*	Ratings from *u* on the items in *I*	Generate a classifier that fits the ratings of *u*, and use it on *I*
Collaborative	Ratings from *U* on the items in *I*	Ratings from *u* on the items in *I*	Identify users in *U* similar to *u*, and extrapolate their ratings on *i*
Knowledge-based	Features of items in *I*, knowledge if they meet user's needs	Description of the needs of *u*	Infer a match between *i* and the needs of *u*
Demographic	Demographic data about *U* and their ratings on items in *I*	Demographic data about *u*	Identify users in *U* demographically similar to *u*, and extrapolate their ratings on *i*

Each of the above recommendation techniques has its own strengths and weaknesses. Table 2 summarizes the advantages and disadvantages of each one (also adapted from [Burke 2002]).

Table 2: Advantages and Disadvantages of Recommendation Techniques

Technique	Advantages	Disadvantages
Content-based	+ Domain knowledge not needed + Quality improves over time + Implicit feedback is sufficient	− New user problem − Non-dynamism − Recommend expected items
Collaborative	+ Domain knowledge not needed + Quality improves over time + Implicit feedback is sufficient	− New user problem − New item problem − Requires large data sets
Knowledge-based	+ No bootstrapping problem + Sensitive to preference changes + Include non-product features	− Requires knowledge engineering − Difficult maintenance
Demographic	+ Cross-domain inference + Domain knowledge not needed + Quality improves over time	− Requires large data sets − Non-dynamism and unreliability − Requires demographic data

Table 2 shows that content-based, collaborative, and demographic filtering recommendation techniques do not require domain knowledge to provide accurate recommendations. Conversely, knowledge-based technique does require extensive domain knowledge to be defined a-priori by system developers. Collaborative and content-based recommendation techniques suffer from the data sparsity problem [Schein et al. 2002], especially at the initial bootstrapping stages. This problem may hamper the accuracy of the generated recommendations, which may improve over time, as the system collects a sufficient amount of information. Conversely, knowledge-based and demographic recommendation techniques are capable of providing accurate recommendations almost regardless of the amount of information about the users available to the system. This also implies the non-dynamism of content-based, collaborative and demographic filtering recommendation techniques. Once a user profile has been collected and stabilized by the system, the process of changing the preferences in the profile is difficult and requires a sufficient period of time. Conversely, knowledge-based techniques quickly adapt to the immediate needs of the users and do not need retraining when the user preferences change.

2.1.6 Hybrid Recommender Systems

Since all the above mentioned recommendation techniques have their own advantages and disadvantages, prior work tried to hybridize (i.e., to combine) them in various ways in order to achieve the best performance. [Burke 2002] surveys and compares possible hybridizations of the above recommendation techniques. The main hybridization methods applied in the state-of-the-art hybrid systems include:

- *Weighted* – recommendation is computed from the recommendations generated by individual recommendation techniques, e.g., by computing a weighted combination of the recommendations generated by content-based and collaborative recommender systems [Claypool et al. 1999].
- *Switching* – the system selects which of the individual recommendation techniques should be applied depending on the task and available data, e.g., by considering the reputation of every individual technique based on the user's acceptance of past recommendations generated by this technique [Tran and Cohen 2000].
- *Mixed* – several individual recommendations are displayed to the user simultaneously, or the generated recommendation contains parts of the individual ones, e.g., by combining parts of TV programs generated by content-based and collaborative recommender systems [Smyth and Cotter 2000].
- *Feature Combination* – individual recommendation techniques combine several sources of user profiles, e.g., using content-based profiles of the users as a mean for computing their similarity for collaborative filtering recommendation technique [Pazzani 1999].
- *Cascade* – several individual recommender techniques sequentially refine the recommendations generated by previous technique, e.g., knowledge-based recommendations are partitioned into buckets, such that all the items in a single bucket are sorted by a collaborative filtering recommender system [Burke 2002].
- *Feature Augmentation* – data derived from the user profiles of a certain recommendation technique serve as input user modeling data (not only the recommendation) for another recommendation technique, e.g., data that are closely related to the items recommended by a content-based recommender system are further used by a collaborative filtering recommender system [Mooney and Roy 1999].
- *Meta-Level* – user profiles of a certain recommendation technique serve in their entirety as input to another recommendation technique, e.g., content-based data collected by a content-based recommender system are transferred to a collaborative filtering recommender system and used for the similarity computation [Pazzani 1999].

Early work on recommender systems [Malone et al. 1987] hypothesized that hybridization of recommendation techniques is beneficial for generated the recommendations. Experimental evalua-

tion conducted in [Burke 2002] validated this hypothesis and practically showed that the accuracy of the generated recommendations improves in most of the hybridizations.

2.2 User Modeling

Successful generation of recommendations by a recommender system requires accurate information about the users to be stored in the user profiles. This information is referred in the literature as a *user model*, i.e., 'an explicit representation of properties of a particular user' [Fink and Kobsa 2000]. Hence, user modeling can be defined as a process of collecting information about the user and constructing her model. Different methods for the personalization of user's interaction with systems based on the information stored in the user models were successfully developed and applied in a variety of domains. A partial list of such applications includes E-Commerce recommendations, Web-browsing, adaptive natural language processing tools, adaptive educational and learning systems, personalized digital entertainment services, information filtering, and digital libraries; there are many others.

Initial motivation and ideas for collecting information about the users as a key element for providing personalized services were discussed in [Perrault et al. 1978]. Further works [Rich 1979a; Rich 1979b] discussed application domains for user modeling and proposed using stereotyping, i.e., classification of a user to one of the a-priori defined types of user, as the basic user modeling approach. In the following years, several systems that collected various types of information about the users were developed. In some of these systems, e.g., [Kobsa and Wahlster 1989] and [McTear 1993], the user modeling task was performed by the system itself, with no distinction between the components maintaining the primary system functionality and the components dedicated for the user modeling. Some other systems, e.g., [Sleeman et al. 1985] and [Kass 1988], contained a separate user modeling component that was responsible for building a model of user's preferences and needs. An extensive review of these early user modeling approaches can be found in [Kobsa and Wahlster 1989].

2.2.1 Building User Models

User models are typically built by the systems through prolonged accumulation of information about the users' needs, preferences and interests. This can be performed either *explicitly* by active

feedback provided by the user as part of her interactions with the system, or *implicitly* by inferring the required information about the user from these interactions.

In the *explicit* collection of user models, the models are collected by direct interaction between the system and the users through various forms, questionnaires, feedback, and so on. For example, [Alfonseca and Rodriguez 2003] presented a system, where the user modeling data was collected from the user's choice of one of the predefined stereotypes and from the user's feedback to a set of questions. Similarly, in [Petrelli et al. 1999] the list of interests was collected from the user's answers to a questionnaire filled before starting the visit. [Price 2000] presents a system, which collected demographic information about the users through a form that the users filled during the initial registration process.

Clearly, this acquisition of user modeling data is inconvenient, as it requires the users to perform a time-consuming and irritating task of explicitly answering numerous intrusive questions [Hanani et al. 2001]. Furthermore, privacy issues might obstruct acquisition of elaborate and accurate user models and deter many potential users from using the system [Cranor et al. 1999]. Hence, in many systems the models of the users were collected implicitly [Webb et al. 2001]. In this case, the user models were inferred using various machine learning techniques from the data recorded while observing the user's behavior during her interaction with the system.

The state-of-the-art works in the domain of machine learning for user modeling were surveyed in [Webb et al. 2001]. That work surveyed application domains providing personalized services (such as recommender systems, news filtering systems, email assistants, and others) and listed several open issues regarding the exploitation of machine learning techniques for the collecting accurate user models:

- Need for labeled data in supervised learning – supervised learning algorithms require explicitly labeled data, which in many cases cannot be obtained from observing users' behavior.
- Concept drift – capability of quickly adapting the learned model to the unique characteristics of the specific application and to reflect the dynamicity and changes in the user interests.
- High computational complexity of learning algorithms – developing heuristic algorithms decreasing the computational complexity of learning techniques.

Both explicit and implicit acquisition of user modeling data raise an important issue of preserving user's privacy [Kobsa 2007]. In [Kobsa 2001b] the authors surveyed the impacts of various privacy components on the dissemination of personalization systems. That work highlighted the fact that a uniform solution for privacy concern does not exist, since privacy preferences and privacy agreements differ from user to user and from country to country. Hence, it encouraged the establishment of a comprehensive agreement that will restrict the access to the information repositories, and will decrease the likelihood of user's private to be accessed by untrusted parties.

We would like to stress the limitations of both acquisition methods. The explicit acquisition of user models implies a direct interaction between the user and the system. As this may be considered by the users as a time-consuming and irritating task, many users may refrain from it, and not allow the system to collect accurate user modeling data. Conversely, in the implicit acquisition of user models, the user modeling data are inferred indirectly by the system from past users' interactions. Hence, this implies various reasoning and inference mechanisms to be exploited by the systems and introduces a degree of uncertainty into the inferred data. In this work we propose to alleviate these limitations and enrich the user models available to the personalization systems by importing the user modeling data collected by other personalization systems.

2.2.2 Domain-Specific User Modeling

User modeling is exploited for the personalization of services in a wide variety of application domains and specific applications. [Hanani et al. 2001] surveys numerous approaches to building user models in various personalization systems and application domains. Specific approaches differ from one domain to another, as dictated by the specific need of the systems. In this subsection, we focus on two domains of Web-browsing and tourism/traveling services as typical representatives of other domains, where similar user modeling techniques are exploited. The reader is referred to [Hanani et al 2001] for a wider list of domains and applications.

- *Web-browsing.* The state-of-the-art centralized user modeling systems in Web-browsing domain were reviewed in [Fink and Kobsa 2000]. That work discussed and compared three typical representatives of such systems: (1) *GroupLens* [Resnick et al. 1994; Clark 2002] filters Usenet news using collaborative filtering, (2) *Personalization Server* [Price 2000] and *FrontMind* [Manna 2000] employ rule-based personalization of Web-pages by clustering users and applying predefined rules to the generated clusters, and (3) *Learn Sesame* [Caglayan et

al 1997] builds browsing models of the users, performs their clustering based on a predefined set of attributes and supports domain modeling through a dedicated model definition language. [Alfonseca and Rodriguez 2003] presented a system for adapting the contents of Web-pages according to the interest of the user. The user models contained the personal interests of the users determined either through explicitly choosing one of the predefined stereotypes and by employing learning the user's feedback to a set of visited Web-pages and the preferred amount of information in a page. The user models facilitated restricting the set of pages shown to the user and setting the percentage of the original document that should be shown. [Billsus and Pazzani 1999] presented a framework for adaptive news access, designed to build daily news compilations for the users. For the acquisition of user models, the users explicitly provided their feedback regarding a number of sample news items. This was repeated on a daily basis, allowing two user models to be collected: (1) long-term models describing the general interests and needs of the users, and (2) short-term models describing the current needs of the users. Both were used to provide to the users adaptive access to news items.

- *Tourism/traveling*. *PTA* [Waszkiewicz et al. 1999] presented a personal travel assistant system implemented as a multi-agent system. *PTA* stored user models as collections of past interactions of the users with the system containing the requirements of the user, the chosen tourist plan and the rejected alternative plans. To generate personalized traveling recommendations, *PTA* matched the user models stored with the available services (e.g., hotels, flights, and so forth) and recommended to the user the set of best matching services. A similar user modeling approach is presented in the *Trip@dvice* tourism recommender system [Ricci et al. 2003]. *Trip@dvice* stores user models as cases [Aadmot and Plaza 1994], including the user's description, her general traveling preferences, and past interactions between the user and the system during tourist route planning: user queries, recommended and viewed options, query modifications, selected tourist route, and so forth. The importance of user modeling for providing adapted guidance during museum visits was discussed in [Petrelli et al. 1999] and [Kuflik et al. 2007]. Those works assumed that users' behavior during the museum visit (e.g., moving velocity, exhibits visited, viewing certain presentations for the exhibits, and so forth) serve as implicit expressions of their interests. This allowed personalized routes to be built within the museum, exhibits to visit to be recommended, and even specific presentations for the exhibits to be suggested. To bootstrap the system, the initial user models were collected explicitly through a questionnaire [Petrelli et al. 1999], or marking interesting tourism sites

from a list of sites [Kuflik et al. 2007]. Tracking and analyzing further user behavior during the visit allowed the initial user models to be refined and the generation of more accurate recommendations.

Typically, user models stored by personalization systems are designed to answer the needs of the system, as the topics of the collected models are focused on the domain of the system, and their representation is adjusted to the tools and personalization techniques applied by the system. This inherently constitutes a severe limitation and hampers the interoperability of personalization systems, as the collected models are 'proprietary' and cannot be transferred between the systems. As a result, there is a need for generic, comprehensive and application-independent user modeling approaches, which, once initialized with domain knowledge, can facilitate interoperability of personalization systems and provide personalization services for a variety of applications and domains [Kobsa 2001a].

2.2.3 Generic User Modeling Systems

The notion of general application-independent user modeling was presented in [Finin and Drager 1986]. That work described *GUMS* general user modeling system, which allowed personalization application developers to define general user stereotypes and relationships between them. For each stereotype, *GUMS* facilitated defining facts that described the stereotype and rules for the system's reasoning about it. At runtime, *GUMS* received new facts about the user from the underlying personalization application, verified their consistency with the current information about the user (informing the application about recognized inconsistencies), and answered the application queries regarding the user. Although *GUMS* was not applied with a real personalization application, it outlined the basic functionality of a general user modeling system: *provisioning of user modeling services that can be configured during the development time*. *GUMS* also demonstrated a new paradigm of generic user modeling system, which could be initialized with application-specific user modeling knowledge, and further serve as a separate independent component in personalization applications.

Most of the general user modeling systems exploited the collected data as input for a certain inference mechanism, mapping their users to one of the predefined stereotypes. For example, *UMT* [Brajnik and Tasso 1994] allowed the definition of hierarchically ordered user stereotypes, and

inference rules for specific types of user modeling data. Conversely, *BGP-MS* [Kobsa and Pohl 1995] allowed assumptions about the individual users and generalized user stereotype groups to be represented using a first-order predicate logic, such that inferences across various types of assumptions could be defined in a first-order modal logic. Similarly, *TAGUS* [Paiva and Self 1995] represented assumptions about the users using first-order formulas, with meta-operators expressing various assumption types. Moreover, it allowed the definition of a stereotypes hierarchy and contained an inference mechanism, a truth maintenance system, and a diagnostic component including a library of possible misconceptions. [Kay, 1995] presented the *um* toolkit, a mechanism for reusing generic user modeling data originated by various personalization systems. The core of the system is a centralized repository of information about the users. Every personalization system had its own view of the relevant parts of the user modeling data via a resolver, which was aimed at providing the required view of the available user modeling data. This architecture and toolkit served a basis for a generic user modeling server called *Personis* [Kay et al. 2002].

[Kobsa 1990] extensively discussed and redefined generic user modeling systems and introduced the term of *user modeling shell systems*. That work defined the user modeling shell systems as *empty expert systems that had to be filled with domain-specific rules for deployment as a real expert system*. Later, [Kobsa 2001a] surveyed the state-of-the-art research of user modeling shell systems, and presented the requirements essential to facilitating their wide dissemination for academic and commercial purposes:

- Generality – domain independence, usability in as many application and domains as possible, and for as many as possible user modeling tasks within these applications.
- Expressiveness – ability to express as many as possible types of assumption about the user.
- Inferential capabilities – capability of performing and supporting various reasoning and inference mechanisms, and resolving the detected conflicts and contradictions.
- Import of external user-related information – ability to integrate user modeling information collected by the current system with the models collected by other remote systems.
- Privacy requirements – support of privacy policies and conventions, national and international privacy legislations, and privacy-enhancing tools and services.
- Quick adaptation – ability to adapt the user modeling services quickly to new users, applications and specific services.

- Load balancing – providing accurate user modeling services, while keeping reasonable technical performance: robustness, availability, and response time.
- Future requirements – support for future applications, such as ubiquitous modeling of mobile users, personalization functionalities in everyday-life devices, and usage not limited to personalization purposes, and expansion to other types of applications.

The rationale for assigning importance to these requirements lies in the exploitation of user modeling approaches in a wide variety of research and application domains, such as artificial intelligence and machine learning [Webb et al. 2001], natural-language dialog [Kobsa and Wahlster, 1989], intelligent tutoring [Kass 1988] and many others. Hence, user modeling shells are expected to support complex assumptions and reasoning about the users, maintain the essential privacy and adaptation constraints, and be usable in as wide as possible range of domains and applications.

Although the notion of a centralized generic user modeling server seems to be an adequate solution, it severely violates various privacy regulations. In this setting the general user model is built in a single place, which can be potentially exposed to attacks by malicious users. Successful attacks may endanger users' privacy, as they allow the attacker to access and reveal all the private, and possibly sensitive, information about the users. This privacy breach of a centralized user modeling server might cause many potential users and systems to refrain from using it [Cranor et al. 1999].

2.2.4 Ontology-Based Representation of User Models

Since the state-of-the-art personalization systems are mostly domain-specific, they usually store partial user models related to their application domain, i.e., the data stored in their user models are dictated by the specific information needs of the systems. However, due to the architectural decisions and specific requirements posed by various personalization techniques, the representation of the user models in various systems from the same application domain may also differ (e.g., natural-language or semi-structured description, features vector, the relevant stereotype and its characteristics explicit ratings, neural network, and others). Moreover, even for the same representation of the user models, various systems might use different terms and languages to express the same underlying concepts models (e.g., synonyms, hyponyms, hypernyms, or simply different

natural languages). Since the accuracy of the personalization increases with the accuracy of the available user modeling data, there is an emergent need for a standardized representation of the user models. Such standardization simplifies sharing and integration of the collected user modeling data and allows more reliable and comprehensive user models to be generated.

[Pohl 1999] identified the need for a standardized user modeling information representation through a comprehensive overview of logic-based representations and reasoning in user modeling systems, and presented an *AsTRa* framework for logic-based user model representation and reasoning. Basically, *AsTRa* obtained its power and flexibility through integrating two popular approaches: (1) assumption-like storage of user modeling data, and (2) logic-based inference formalism. That work also discussed the integration of *AsTRa* prototype in *BGP-MS* user modeling shell, which was proposed in [Kobsa and Pohl 1995].

Much work focused on the issue of ontology-based representation of user models in a *scrutable* [Kay 2006] and *reusable* [Kay 1999] manner. In the context of user modeling, the term 'scrutable' means that the available user modeling data are understandable upon examination and observation, whereas 'reusable' means that various systems can benefit from the content of the same user models. Such representation allows disregarding the issue of user model limited to a particular system or application domain, as the structure and content of the user models are based on ontologies, which facilitate access to a customized explanation of the user model components.

For example, [Razmerita et al. 2003] presented a general ontology-based user modeling architecture called *OntobUM*. *OntobUM* integrated three ontologies: user ontology defining the users, domain ontology defining the relationships between various domains, and log ontology defining the user-system interaction. User models were collected both through a user profile editor, and by classifying the users to one of the predefined stereotypes based on their past activities. Another approach to a standardized representation of the user models was presented in [Heckmann and Krueger 2003]. That work presented *UserML*, a knowledge representation language for describing user models in various domains, where every object contained a pointer to the ontology, specifying its particular domain.

On the basis of *UserML*, [Heckmann et al. 2005] introduced *GUMO*, a comprehensive set of general user modeling ontologies, allowing uniform interpretation of distributed user models in intelligent environments. The ontologies of *GUMO* were represented using *OWL* semantic language [McGuinness and van Harmelen 2004] and were available for all the involved user modeling and personalization systems at the same time. The fact that *GUMO* was commonly accepted by all the involved systems and focused on user modeling tasks facilitated the exchange of user modeling data between various systems to be significantly simplified and allowed the inherent problem of syntactical and structural differences between the systems to be overcome. That work also discussed possible combinations of *GUMO* and *UserML* [Heckmann and Krueger 2003]. A situation, where *UserML* serves as a uniform user modeling language and *GUMO* serves as a commonly accepted ontology across multiple user modeling and personalization systems, facilitates exchange of user modeling data between various systems and better composition of partial user models collected by various systems.

The main limitation of the ontology-based approaches lies in the static nature of the ontologies. As can be observed from the above mentioned works, they all assume that the ontology is predefined and available, i.e., it is modeled a-priori by external domain experts. This inherently implies a centralized management of the ontology, which can hardly evolve and be modified over time. This contradicts our assumption about a decentralized and highly dynamic setting, where each personalization system can potentially provide valuable user modeling data, and appeals for a more flexible approach to the data representation.

2.2.5 Ubiquitous User Modeling

Nowadays, a situation where multiple personalization systems store information about their users is not unusual, since the users are surrounded in their everyday activities by various devices, which can provide personalized services. These devices vary from PDAs, embedded computers in cameras, cars, or mobile phones, up to high performance wearable computers. This paradigm of 'present everywhere' computing is referred to in the literature as *ubiquitous computing* [Weiser 1991]. Due to the limitations of ubiquitous computing devices, one of the main challenges of ubiquitous computing is the issue of ubiquitous personalization services, which, in turn, highlights the importance of *ubiquitous user modeling* [Heckmann 2005; Carmichael et al. 2005; Lorenz and Zimmermann 2006]. According to the definition of [Heckmann 2005], ubiquitous user

modeling is "an ongoing modeling and exploration of user behavior with a variety of systems that share their user models."

A rather simplistic approach to resolving the issue of ubiquitous user modeling was suggested in [Potonniee 2002]. That work proposed building an application adaptation framework using a centralized storage of the UMs on the users' personal smart cards. The smart cards stored and managed the users, partially resolving the privacy and availability issues which are of the highest importance in a decentralized ubiquitous environment. Compared to a solution, where the profiles are stored in a central server, the use of smart cards made the profiles available in any context, enhanced the data privacy and security by allowing the users to control their user models fully (e.g., what user modeling data should be exposed, to what extent, to which systems, and so forth), and avoided the communication delays. However, the smart cards aggravated the problem of a single point of failure, as all the available personal information was stored in a single repository, the disclosure of which by an attacker could have a disastrous effect.

[Kay et al. 2003a] presented and overviewed a centralized architecture for user modeling services in a ubiquitous environment. That work presented a general user model stored on a central server as a composition of partial user models, stored by various ubiquitous personalization applications. Each application maintained its own inference mechanism that allowed it to update the general user model and to extract from it the needed information. When the user modeling data were needed, the relevant information was extracted from the general user model and adapted to the specific needs of the personalization system that requested the data. Thus, every application could generate its own view of the general user model. It should be noted that according to this approach, the users are in full control of the parts of the general model that are accessible by other applications through defining the access permissions. In the following work, [Kay 2003b] presented a visualization tool that allowed the users to explore their models, to navigate through them, to edit the components of the model, and to set its access permissions.

A similar approach was presented in [Niederee et al. 2004]. That work proposed using the Unified User Context Model (*UUCM*) for the purposes of exploiting parts of the general user model in several personalization systems. The representation of user modeling data in *UUCM* was based on a single shared ontology, and allowed it to support two main features: (1) generality – usability

in as wide as possible variety of domains and specific applications, (2) expressiveness – capability to express as wide as possible range of knowledge about the user. [Mehta 2005a] presented an approach for cross-system personalization, based on the idea of a context passport that was inspired by *UUCM*. The personalization system extracted the required user modeling data from the context passport, performed the personalized activities based on the extracted data, and updated the passport. Finally, [Mehta 2006] defined three main stages of cross-system communication protocol for sharing user modeling data across different systems in a user-centric way:

- Negotiation – achieves an understanding on the type of user modeling data that is needed by the target personalization system and agreement on common ontology and vocabulary.
- Personalization – extraction of the required user modeling data from a set of source systems and their transfer to the target personalization system.
- Synchronization – replication and update of the user modeling data upon completion of personalization tasks by the target personalization system.

[Lorenz 2005] proposed an agent-based architecture for a distributed user modeling in the ubiquitous environment through sharing of partial user models. That work proposed achieving user models sharing through a network of cooperating agents, acting as active components and using predefined communication framework. Each agent might manage a part of the general user modeling data, but the whole network of agents will integrate together these partial user models into a distributed representation of the knowledge about the user. The negotiation between the agent, and exchange and integration of partial user models were achieved through designated brokering agents, virtually representing other systems in the network.

In all those works, the task of user model integration (maintaining the inference mechanism from/to the centralized model, or direct negotiation between the agents) is aggravated by the wide variety of existing user modeling techniques and representations. As neither the policy for the sharing of user modeling data, nor the standards or protocols for the conversion between various representations of the data are defined, practical interoperability of any two personalization systems should be explicitly provided by the system developers. Taking into account the large number of existing personalization approaches and user model representations requires to develop a vast number of conversion mechanisms. Hence, the technology proposed in this work introduces a certain level of standardization into the required interoperability of personalization systems.

Part 2:

User Model Mediation

Chapter 3: User Modeling Data Representation

A unified model for user modeling data representation is required in order to facilitate the UM mediation. The discussion starts with a two-dimensional representation which is an abstraction of the data representation adopted by most of the existing systems. This model is then extended to a three-dimensional representation reflecting the context-awareness aspects.

3.1 Two-Dimensional Representation of User Models

Most of today's recommender systems base the warehousing, i.e., storage, access and retrieval, of their UMs on a two-dimensional matrix representation. The two *generalized* dimensions of this representation are the *users* and the *items*. These dimensions are referred to as generalized because they may be described by sets of specific features. Hence, if the user is described by *n* features (e.g., age, gender, and others) and the item by *m* features (e.g., color, shape, price, and others), the space of all possible user and item pairs is described by an *n+m* dimensional space. For instance, when the users and the items are described by their unique identities only, the space of all possible users and items pairs is two-dimensional, where the first dimension refers to the user identifier and the second to the item identifier. In this case, the ratings given by users to items are described by a map from the two-dimensional space to a numeric range, e.g., *{1, 2, 3, 4, 5}*. In a more concrete way, in this situation, an *NxM* matrix (representing *N* users and *M* items) either represents directly or reflects the ratings given by the users to the items. The ratings are given on a predefined scale and could be given in an explicit or implicit way. Explicit ratings are typically provided by the users, while implicit ratings are inferred by the system through observing user behavior indicators. For example, if the user bought the recommended product, the system implicitly interprets it as a positive rating.

When the users and the items are described by sets of features, the matrix is still referred to as the description of ratings given by users to items. However, such matrix is a high dimensional matrix, i.e., it is a function R'_{gen} from the *n+m* dimensional space of pairs $User_{feat} \times Item_{feat}$ to a set of ratings:

$$R'_{gen}: User_{feat} \times Item_{feat} \rightarrow rating.$$

In the above definition $User_{feat}$ represents the user features, $Item_{feat}$ represents the item features, and *rating* represents the ratings given by the users to the items.

In fact, this function is not defined for all the possible user and item pairs, i.e., the system may not know the rating values given by a user to all the items. Hence, given a user who requests a recommendation, the goal of a recommender system is: (1) to estimate the rating value for some items, which the user has not previously rated, and (2) to suggest some items to the user, for instance those items having maximal predicted rating. Note that the actual recommendation task heavily depends on the exact functionality (and service) provided by the system. This can include recommending the best item, ranking *N* best items, filtering highly irrelevant items, and many others [Adomavicius and Tuzhilin 2005c]. In the rest of this work the recommendation task refers to predicting a future rating, which would be assigned by a certain user *u* to a certain item *i*.

Although R'_{gen} was defined as a function from a two-dimensional matrix, both of its basic dimensions $User_{feat}$ and $Item_{feat}$ can be described using a multidimensional representation by a set of features. However, since the described data representation does not imply use of any commonly agreed ontology, the separation between the basic dimensions of $User_{feat}$ and $Item_{feat}$ is ambiguous and somewhat artificial. Some systems may classify certain ephemeral features as features describing the users, while other systems may classify them as features describing the items. For example, consider a travel recommender system and a feature representing the *season* of the travel. This feature could be interchangeably considered as one of the user features (e.g., the user searching for a holiday resort in winter), or as one of the item features (e.g., holiday resort in winter). To overcome this, the representation can be considered as a single multi-dimensional space of features, which reflects a single integrated list of features, where certain sets of features can be grouped into the basic dimensions of $User_{feat}$ and $Item_{feat}$.

This R'_{gen} representation is applicable to a wide variety of state-of-the-art recommendation techniques, as can be seen in the following examples:

- In collaborative filtering [Herlocker et al. 1999], the two-dimensional matrix R_{CF} is referred to as the *ratings matrix* and is represented by:

$$R_{CF}: User_{id} \times Item_{id} \rightarrow rating,$$

where $User_{id}$ and $Item_{id}$ are the unique one-dimensional identifiers of users and items[2], and *rating* is the rating given by the user to the item. In this case, an individual UM is represented by a set of ratings given by the relevant user, and is referred to in the literature as the *ratings vector*. For example, consider the following ratings matrix for the domain of movies R_{CF}*={ ((Alice, "The Lord of The Rings"),1), ((Alice, "The Matrix"), 0.8), ((Bob, "Psycho"), 0.2), ((Bob, "Friday the 13th"), 0) },* representing the ratings of two users, Alice and Bob, given on a continuous scale between *0* and *1*. Typically, collaborative filtering systems do not store any additional information about the features and content characteristics of users and items, beside their identities.

- In content-based filtering [Morita and Shinoda 1994], the two-dimensional matrix R_{CB} is represented by:

$$R_{CB}: User_{id} \times Item_{feat} \rightarrow rating,$$

where $User_{id}$ represents the unique identifier of the users, $Item_{feat}$ represents a feature space describing the item's features, and *rating* reflects the user's ratings (e.g., in form of weights) to the items characterized by that feature. In this case, the UM is represented by the values reflecting the ratings given by $User_{id}$ to certain $Item_{feat}$ features, originated by the descriptions of items. For example, content-based matrix R_{CB} from the previous example can be R_{CB}*={ ((Alice, science-fiction), 0.9), ((Bob, horror), 0.1) }*. In this example, each movie is described by a content-related feature representing the genre of the movie, taking in these examples the values *horror* and *science-fiction*. It can be observed that in content-based recommender systems the *raw* UMs contain the ratings of the users to the items described by a set of features. This information is typically used to build a refined model that depends on the specific classification technique used by the recommender system.

- In demographic filtering [Kurlwich 1997], the two-dimensional matrix R_{dem} is represented by:

$$R_{dem}: User_{feat} \times Item_{feat} \rightarrow rating,$$

where $User_{feat}$ represents a set of features describing certain demographic characteristics of a group of users to which the user belongs, $Item_{feat}$ represents either the unique identity of the item or a set of features reflecting the item's content, and *rating* virtually represents the ratings given by a group of users with certain demographic characteristics to the items. In this case, the UM is represented by a combination $(user_{feat}, item_{feat})$ and the ratings provided by the user described by $user_{feat}$ to the items, containing $item_{feat}$. For example, R_{dem} of the users Alice and

[2] Although in this case the identity of the user (item) is considered as one of the features, it should be stressed the identity is a special feature, which facilitates a unique identification of user (item) in all the systems.

Bob from previous examples can be R_{dem}=*{(female, science-fiction), 0.9), ((male, horror), 0.1)}*. In this example it should be pointed out that the very notion of user and item can depend on the specific recommendation technique. Here, for instance, the users are represented by a single user stereotype that is defined only by the gender feature. Similarly, the items are described only by the genre feature.

All the previous recommendation techniques could, in principle, adopt the generalized user and item representations, as defined for the demographic filtering approach, which generalizes the user description, and content-based method, which generalizes the item description. In fact, many recommender systems providing personalized recommendations based only on the ephemeral session data adapt the generalized user and item representation to the specific needs of the systems and exploit this representation for the purpose of generating the recommendations [Ricci et al. 2006b].

In addition, the way the ratings of the users are represented is highly heterogeneous across various recommender systems. Some systems store numeric ratings given by the users on a predefined, but not standard, scale (e.g., the scale may be discrete or continuous, the range of possible values may vary from one system to another, and so on), some store symbolic ratings (e.g., positive or negative ratings, thumbs-up or thumbs-down, and so on), some store system-specific feedback derived from user behavior (e.g., examining or not the recommended item, purchasing or not the recommended product, and so on), some store the resultant navigation history of the user (e.g., opening or not the recommended Web-link, period of time spent viewing the recommended Web-page, and so on), and others store the free-text feedback provided by the users [Hanani et al. 2001].

Several types of the user feedback are discussed and compared in [Montaner et al. 2003]. Such feedback is classified into four categories:

- No feedback – modifications of the user data are done manually by the users using a specific component provided by the system.
- Explicit feedback – explicit opinion provided by the user. For example, numeric ratings assigned to artists or music bands [Shardanand and Maes 1995], annotations of viewed docu-

ments [Goldberg et al. 1992], or binary opinions regarding the interestingness of Web-pages [Pazzani and Billsus 1997].

- Implicit feedback – user opinion is inferred by the system from monitoring the user's behavior. For example, analyzing lists of preferred leisure activities [Kurlwich 1997], reading times of the received messages [Morita and Shinoda 1994], or usage data of hypermedia systems [Kobsa et al. 2001c].
- Hybrid feedback – combines both the explicit and implicit user feedback, such as in [Resnick et al. 1994; Joachims et al. 1997; Sakagami et al. 1997].

To resolve this heterogeneity and to refer to the wide variety of user feedback to the provided recommendations in a uniform manner, all the possible types of feedback are generalized and denoted by the term *evaluation* [McNee et al. 2003].

To address the heterogeneity of the evaluations assigned by (or predicted for) the user to an item, the R'_{gen} function was generalized to the *experience* of a user for an item. An experience is defined as an evaluation function that maps a pair, the user who had the experience and the item experienced by the user, to an evaluation. An experience evaluation details how the user and the item are linked together. Formally, the experience is represented by:

$$Exp: User_{feat} \times Item_{feat} \rightarrow evaluation$$

where $User_{feat}$ and $Item_{feat}$ represent the feature spaces describing user and item features, and *evaluation* represents the feedback given by the user described by $User_{feat}$ for the item described by $Item_{feat}$. Figure 1 schematically illustrates the representation of experiences in a two-dimensional space.

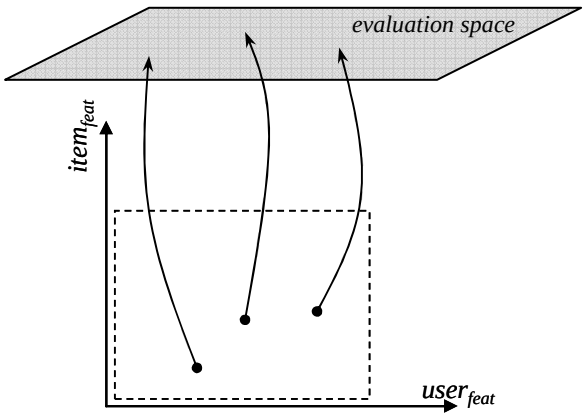


Fig. 1. Representation of Experiences and their Evaluations in Two-Dimensional Space

For example, consider an experience *e* described verbally by "Alice likes science-fiction movies". Using a simple object-oriented-like notation, this experience can be represented by *Exp(user.name=Alice, item.movie.genre=science-fiction)=like*. This representation of the experience shows that the *evaluation* of a user, whose feature *name* is assigned the value Alice for the *item movie*, whose feature *genre* is assigned the value *science-fiction* is *like*.

This allows further generalization of the R'_{gen} representation of the matrix to the *Exp* representation for the UM warehousing comprising user- and item-dependent representation of experiences and evaluations. Also over the *Exp* representation, a single UM, i.e., the model of a concrete user, is considered as a set of the *Exp* contents restricted to values of the features of this user. In other words, the user model for user *u* is the range of the *Exp* function restricted to the user *u*. Moreover, the *Exp* representation of experiences allows one to devise the following formulation: "*the recommendation is a task of predicting future evaluation of a new experience for a specific combination of* ($user_{feat}$, $item_{feat}$) *values, based on a set of past experiences*". In other words, the recommendations are aimed at predicting evaluations of the new experiences using the knowledge obtained from past experiences.

3.2 Three-Dimensional Representation of User Models

The above generalization of the classical recommendation problem does not overcome a severe limitation of the majority of recommendation techniques: ignoring the context of the experience [Buriano et al. 2006]. There is a variety of definitions for the term *context* in the literature. For example, it can refer to the user location, the time or the temperature of the day [Brown et al. 1997], or it can be considered as the subset of physical and conceptual states of interest to a particular entity [Pascoe 1998]. One of the most comprehensive definitions of context is given in [Dey and Abowd 1999]: "*Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves*".

With respect to recommender systems, [Goker and Myrhaug 2002] defines context as a description of aspects of a situation and splits the general user context into five components: (1) environment context – captures the entities that surround the user; (2) personal context – captures the

state of the user and consists of two subcomponents, the physiological context and the mental context; (3) task context – captures what the persons (actors) are doing in this user context; (4) social context – captures the social aspects of the current user context, such as friends, enemies, neighbors, co-workers and so on; and (5) spatiotemporal context – captures aspects of the user context relating to the time and spatial extent for the user context.

In fact, user preferences represented by the UM are generally valid only within specific contextual conditions, such as spatial, temporal, emotional, and other conditions. That is, a user's preferences stored in the UM may change as a function of various contextual conditions. Nonetheless, the generalized matrix representation *Exp* considers an experience as a user- and item-dependent entity only, not influenced by the contextual conditions, which may actually affect the evaluation of the experience. For example, two experience evaluations may be defined as "*Alice likes to see comedy movies with her friends*" and "*Alice does not like to see comedy movies with her parents*". In this example, if the companion of a user is treated as a contextual condition, the evaluation of the same experience is positive in one contextual condition and negative in another. Hence, to facilitate provision of context-aware recommendation, the above two-dimensional (user- and item-dependent) representation *Exp* should be extended by a third general dimension, reflecting various contextual conditions and features that may be considered by the recommender system.

The context-awareness issue has lead to a multidimensional warehousing of the UMs that captures the dependencies between the ratings and a generalized user-, item- and context-dependent model [Adomavicius et al. 2005a]. This model extends the two-variables function *Exp*, ignoring the context-awareness issue, to a three-variables function Exp_{CA}, incorporating a third dimension of context. Given the above generalization of ratings to the experiences *evaluation*, context-aware experience is defined by:

$$Exp_{CA}: User_{feat} \times Item_{feat} \times Context_{feat} \rightarrow evaluation.$$

Figure 2 schematically illustrates the representation of context-aware experiences in a three-dimensional space.

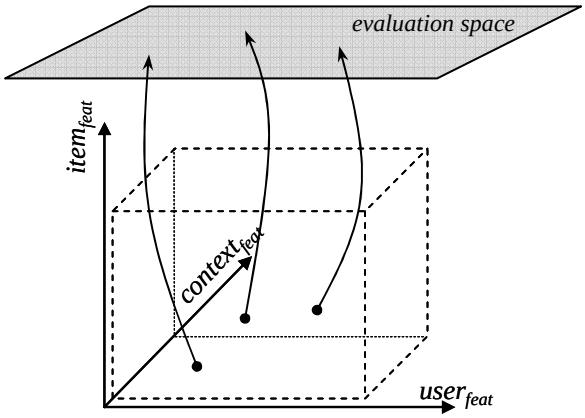


Fig. 2. Representation of Context-Aware Experiences in Three-Dimensional Space

This representation, in addition to the standard $User_{feat}$ and $Item_{feat}$ features, also includes $Context_{feat}$ that represents the contextual conditions (or the values of the contextual features) of the experience. Similarly to $User_{feat}$ and $Item_{feat}$, $Context_{feat}$ is also described using a multidimensional representation by a set of features. Hence, a specific contextual condition of the experiences is referred to as a subspace of this multidimensional contextual space. For instance, in the above mentioned example, only one contextual feature of *companion* out of a large $Context_{feat}$ set of contextual features is mentioned. When the *companion* feature is assigned the value of *friends*, the evaluation is positive, and when it has the value *family*, the evaluation is negative.

It should be stressed that modifying the representation of *Exp* function to Exp_{CA}, i.e., incorporating the contextual features, does not have any effect on the UM warehousing. UMs are still referred to as collections of past experiences, whereas the experiences are now context-aware. Also, although the definition and representation of experience was modified to capture the context-awareness issue, the definition of the recommendation task remains unchanged. This still consists of predicting the evaluations of the new experiences based on a set of past experiences and their evaluations. However, since the experiences were now modified to include the contextual features $context_{feat}$, the recommendations should be provided in a context-aware manner, i.e., they should refer to a certain combination of $user_{feat}$, $item_{feat}$ and $context_{feat}$ values, and not of $user_{feat}$ and of $item_{feat}$ values only.

Previous observations regarding the ambiguous separation of features between the basic dimensions are still valid in the three-dimensional representation. Various recommender systems can misinterpret the same feature and classify it to different basic dimensions. For example, the above

feature of the *season* in the tourism recommender system in the previous section can naturally be considered as a contextual feature. Hence, although the third basic dimension of context was introduced, the whole representation can still be considered as a single multi-dimensional space of features, where certain sets of features can be grouped into the basic dimensions of $User_{feat}$, $Item_{feat}$, and $Context_{feat}$.

Chapter 4: User Modeling Data Mediation Framework

The main problem in providing high quality context-aware recommendations using the context-aware Exp_{CA} representation for the UM warehousing is the sparseness of the data stored in the UMs, i.e., the lack of sufficient user modeling data about the user in specific contextual conditions [Ricci 2006a]. The problem of insufficient information for generating high-quality recommendations is a well-known problem of traditional recommender systems [Linden et al. 2003], which rely only on the two-dimensional representation R'_{gen} and ignore the contextual dimensions. This problem is aggravated when the contextual information is considered, as the initially sparse two-dimensional experiences are partitioned among multiple contexts, reflecting the specific contextual conditions of the experiences. As a result, the amount of available user modeling data referring to a specific contextual condition significantly decreases when the context-awareness issue is taken into account. Hence, a major question refers to the tradeoff between the specialization of context-aware recommendation generations and the reduction of the available user modeling data.

This work discusses an approach aimed at overcoming the sparseness problem using a *mediation* of UMs and user modeling data. The exact definition of mediation is formulated as follows: "*mediation of UMs is a process of importing the user modeling data collected by other (remote) recommender systems, integrating them and generating an integrated user model for a specific goal within a specific context*". In this definition, the term integration refers to a set of techniques aimed at resolving the heterogeneities and inconsistencies in the obtained data. The mediation process facilitates instantiating the UMs through inferring the required user modeling data from past experiences and their evaluations in a three-dimensional context-aware representation space. Hence, it enriches the existing UMs (or bootstraps empty UMs) in the target recommender system using the data collected by the remote systems and facilitates provision of better context-aware recommendations. In the rest of this section a general architecture of UM mediation will be presented and four practical mediation methods will be extensively discussed.

4.1 User Modeling Data Mediation Architecture

Two parties are involved in the mediation process. On the one hand, there is a **target** recommender system, i.e., the system requested to provide personalized recommendations to the user. Formally, this system acts as the initiator of the mediation process by requesting the available user modeling data from other systems. On the other hand, there are numerous **remote** recommender systems that may provide relevant user modeling data (i.e., past experiences) to the target recommender system. More precisely, these might not be recommender systems only, but also various services, Web-sites, sensors and even a user's personal devices, that collected past experiences of the user. These two parties are interconnected via the UM mediator, which constitutes the core element of the mediation process. The general architecture of the UM mediation process is illustrated in Figure 3.

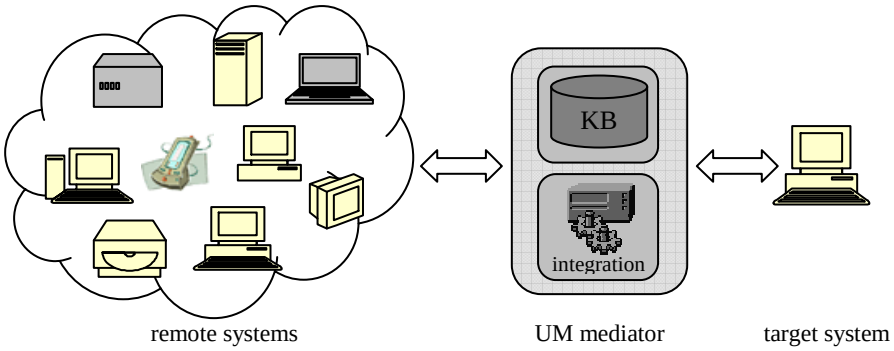


Fig. 3. Architecture of the User Modeling Data Mediation

As discussed earlier, the main difficulty of the UM mediation and the main focus of this work is overcoming the heterogeneity of the user modeling data. For example, recommender systems from different application domains imply different user modeling data stored in the UMs. Even within the same domain, different systems may store different information in their partial UMs, according to the specific recommendation technique being exploited (e.g., ratings vector in collaborative filtering UMs [Herlocker et al. 1999] vs. a feature vector of interest topics in content-based UMs [Morita and Shinoda 1994]). Moreover, even the UMs of two recommender systems from the same application domain exploiting the same recommendation technique may use different terms to describe equivalent underlying objects, i.e., users, items, or domain features.

Hence, successful completion of the UM mediation task requires (1) developing and applying reasoning and inference mechanisms for converting user modeling data between various representations, applications and domains, and (2) identifying and exploiting semantically-enhanced knowledge bases, actually facilitating the above reasoning and inference. Combinations of various reasoning and semantic tools will allow commonalities between the user modeling data representation of various systems to be identified. As a result, the mediator consists of two principal components:

- **Integration Mechanism.** The obtained past experiences may be represented in different ways, e.g., using various ontologies, domain-specific and application-specific terminology, or even different languages. In addition, the evaluations of the same experience in different systems may be contradictory. Hence, this component is responsible for resolving conflicts and heterogeneities in the obtained user modeling data using various reasoning and inference mechanisms. This requires the integration mechanism to implement and apply certain policies for conflict resolution in the obtained data[3].
- **Knowledge Base (KB).** This is an auxiliary component, used by the integration mechanism. It contains semantically-enhanced inter-domain and intra-domain knowledge bases representing dependencies and relationships between various user, item and context features. The data stored in the knowledge bases facilitate resolving the heterogeneities in the obtained user modeling data. For example, it allows reconciling the ontologies exploited by various recommender systems, converting the terms used by certain systems to a standard representation, and even provides machine translation tools resolving cross-lingual dependencies.

The envisioned flow of the user modeling data mediation process consists of the following stages (as illustrated in Figure 4):

1. The recommendation request is treated by the target system as a request for a prediction of the evaluation of the new experience for a specific combination of $user_{feat}$, $item_{feat}$, and $context_{feat}$. By predicting this evaluation, the target system can also determine whether the item should be recommended to the user in a given context. The target recommender system queries the mediator for the UMs, containing past experiences that are relevant for predicting the evaluation of the new experience. The query contains the required $(user_{feat}, item_{feat}, context_{feat})$ combination.

2. The mediator analyzes the query and determines the set of remote recommender systems, which may store potentially relevant past experiences. This analysis is done using semantic data provided by the knowledge base.
3. The mediator forwards the query to the set of remote recommender systems that were determined in the previous stage.
4. Remote recommender systems, which actually store the relevant experiences, respond to the query and send to the mediator their locally collected UMs and/or the relevant experiences only.
5. The mediator integrates the obtained experiences using the semantic data provided by the knowledge base. Clearly, different combinations of UM representations in remote and target systems will require different integration mechanisms.
6. The generated user modeling data are sent to the target recommender system. Since the user modeling data of the target system are enriched in comparison to the locally collected data stored before the mediation, the system is capable of providing better recommendations to the user.

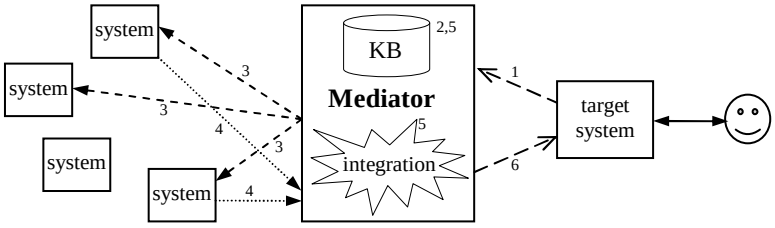


Fig. 4. Stages of the Mediation

To illustrate the mediation flow, consider again the above example of a network of recommender systems dedicated to digital entertainment. The network consists of music, movies, TV programs, books, and humor recommender systems. Consider Alice, one of the users of the movies recommender system who asks the system to suggest movie. To provide a better recommendation, the target movies recommender system queries the mediator for the relevant user modeling data (step *1*). The mediator analyzes the request and the data provided by the movie recommender system (e.g., past opinions of Alice on the movies she has already seen and a list of potentially recommendable movies in theaters tonight) and identifies the set of remote recommender systems that

[3] It is reasonable to assume that various systems will provide user modeling data with different levels of accuracy and up-to-date information. Although the paper highlights the importance of resolving such conflicts, developing conflict resolution policies falls beyond its scope.

can potentially provide relevant past experiences (step *2*). Imagine that these systems are TV, books, music, and jokes recommender systems. The mediator forwards the query for the relevant past experiences to these systems (step *3*).

The remote recommender systems, which store the relevant experiences, send them back to the mediator (step *4*). Imagine that only TV programs and books recommender system stored the relevant experiences: the TV programs system sends the list of programs seen by Alice during the last week and the books system sends the list of books purchased by Alice through the Web-site of the books recommender system. The mediator integrates the acquired past experiences into a single UM using the knowledge base and converts it to the format required by the specific recommendation technique exploited by the target movies recommender system (step *5*). For example, it mines the information available about the TV programs seen by Alice, extracts the topics of these programs, and checks whether there are recommendable movies with overlapping or similar topics. The mediator also identifies the writers of the books purchased by Alice and checks whether there are recommendable movies based on the novels written by these writers. Finally, the derived user modeling data (i.e., the list of topics and writers) is forwarded to the movies recommender system (step *6*), where it is used for generating personalized recommendations.

4.2 User Modeling Data Mediation Methods

The above scenario immediately raises a question: "What user modeling data are relevant for the mediation and need to be imported from the remote systems"? In principle, any available user modeling data (i.e., any past experience) may be relevant to some extent as input to the mediation process, since they may help predict the evaluations of the new experience. For example, consider a target recommender system that is supposed to predict the new experience evaluation for a specific combination of $(user_{feat}, item_{feat}, context_{feat})$ values. The possible groups of $(user_{feat}, item_{feat}, context_{feat})$ combinations in past experiences stored by other recommender systems are as follows (the respective mediation methods will be extensively discussed later in this section):

- Experiences that refer to the same combination $(user_{feat}, item_{feat}, context_{feat})$. They represent past experiences of the same $user_{feat}$ for the same $item_{feat}$ in the same $context_{feat}$, where the values of certain experience features, referring to the same objects, may be represented in different ways.

- Experiences where the values of two features are the same and the value of one feature differs. Three possible combinations are:
  - *(user'$_{feat}$, item$_{feat}$, context$_{feat}$)* – past experiences of another *user$_{feat}$* for the same *item$_{feat}$* in the same *context$_{feat}$*.
  - *(user$_{feat}$, item'$_{feat}$, context$_{feat}$)* – past experiences of the same *user$_{feat}$* for another *item$_{feat}$* in the same *context$_{feat}$*.
  - *(user$_{feat}$, item$_{feat}$, context'$_{feat}$)* – past experiences of the same *user$_{feat}$* for the same *item$_{feat}$* in another *context$_{feat}$*.
- Experiences where the value of one feature is the same and the values of two features differ. Three possible combinations are:
  - *(user$_{feat}$, item'$_{feat}$, context'$_{feat}$)* – past experiences of the same *user$_{feat}$* for another *item$_{feat}$* in another *context$_{feat}$*.
  - *(user'$_{feat}$, item$_{feat}$, context'$_{feat}$)* – past experiences of another *user$_{feat}$* for the same *item$_{feat}$* in another *context$_{feat}$*.
  - *(user'$_{feat}$, item'$_{feat}$, context$_{feat}$)* – past experiences of another *user$_{feat}$* for another *item$_{feat}$* in the same *context$_{feat}$*.
- Experiences where the values of all three features are different. These experiences refer to *(user'$_{feat}$, item'$_{feat}$, context'$_{feat}$)* and represent past experiences of another *user$_{feat}$* for another *item$_{feat}$* in another *context$_{feat}$*.

Clearly, the first group of experiences is the most important for UM mediation, as it provides past evaluations of the target user for the required item in the relevant context. Such experiences require integration mechanisms for resolving possible heterogeneities in the representations of feature values or experience evaluations to be applied. The second group of experiences (with one feature different from the required combination) is also important for the mediation. These experiences represent past evaluations, where the values of two out of three features match the values of the features in the new experience. In this case, the mediation requires applying inference mechanisms for identifying the relationships between the different values of the feature that differs and projecting the available evaluations onto the new experience.

In the third group of experiences, the values of two out of three features are different, and only one feature matches the values of a feature in the new experience. Hence, mediation of such ex-

periences requires applying more complicated inference mechanisms (e.g., several consecutive inferences similar to the inferences from the previous group of experiences, where the value of only one of the features was different). Although this user modeling information may be relevant and may enrich the user modeling data in the target system, applying complicated inference mechanisms may degrade the original data represented by the past experiences. Therefore, it is not currently suggested that such experiences will be used in the mediation process. Obviously, the situation is even worse for the fourth group of experiences, where the values of all three features are different, and the mediation requires three inference mechanisms to be applied. Hence, these experiences are also not considered for mediation at the moment.

In summary, two groups of experiences that may be considered as input for the mediation process are: (1) experiences having the required values of all three features, and (2) experiences having the required values of two features and a different value of one feature. Further analysis yields four particular types of UM mediation over the context-aware three-dimensional representation of experiences.

The first type of mediation is conducted between experiences having the required values of all three features, i.e., between heterogeneous representations of the same experience. Such mediation is referred to as **cross-representation mediation**. The other three mediation types are referred to as **cross-dimension mediations**. They are conducted over the experiences having the required values of two features and a different value of one feature. This means that the values of two out of three dimensions in the space are fixed and the mediation is performed across the third dimension.

Three types of cross-dimension mediations are possible: (1) **cross-user mediation**, where the values of item and context features are fixed and the user in the experiences is allowed to be modified; (2) **cross-item mediation**, where the values of user and context features are fixed and the item in the experiences is allowed to be modified; and (3) **cross-context mediation**, where the values of user and item features are fixed and the context in the experiences is allowed to be modified.

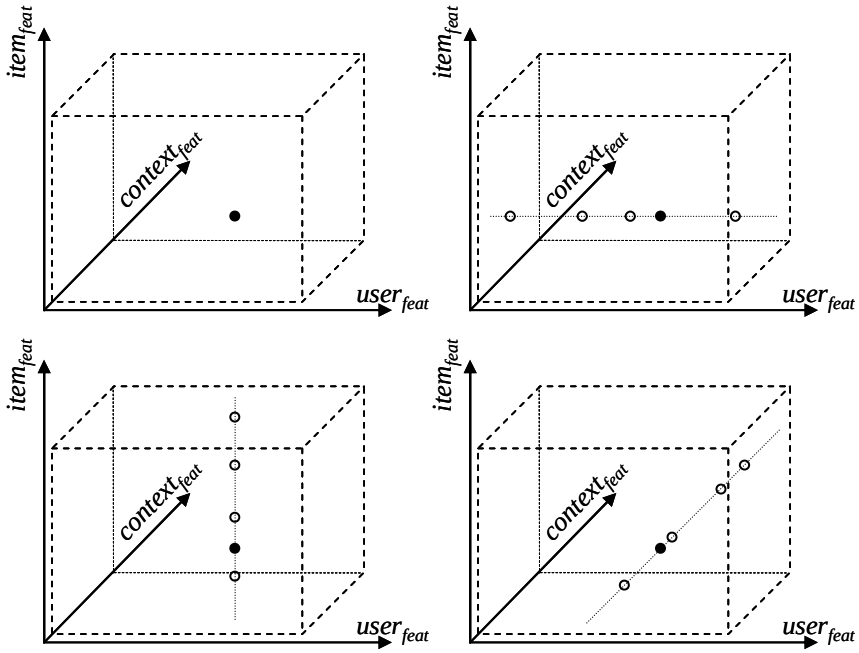


Fig. 5. Cross-Dimension Mediations: top-left – the new experience to be predicted; top-right – experiences imported at cross-user mediation; bottom-left – experiences imported at cross-item mediation; bottom-right – experiences imported at cross-context mediation

Figure 5 schematically illustrates the three cross-dimension mediations in a three-dimensional space. The top-left chart represents only the new experience, i.e., a certain combination of *(user$_{feat}$, item$_{feat}$, context$_{feat}$)* features, where the user's future evaluation needs to be predicted. Three other charts represent past experiences that can be imported at various types of cross-dimension mediation: the top-right chart represents the experiences imported at cross-user mediation, the bottom-left represents the experiences imported at cross-item mediation, and the bottom-right represents the experiences imported at cross-context mediation. In all the charts, the black dot represents the new experience, where the evaluation needs to be predicted and the circles represent past experiences that are imported and integrated at the respective type of mediation. Note that cross-representation mediation is not shown graphically in Figure 5. This type of mediation can be considered as mediation conducted between experiences stored in a single cell of the three-dimensional space, i.e., between the experiences stored in the black dot representing the new experience. In the rest of this section, all four possible types of UM mediation are extensively discussed.

4.2.1 Cross-Representation Mediation

This mediation is aimed at resolving the heterogeneity in the representations of the experiences of the same user for the same item in the same context. In other words, it incorporates past experiences of the same user for the same item in the same context, but expressed in different ways. For example, consider the following representation of the same item, a movie "*Gone with the Wind*", in two datasets: EachMovie [McJones 1997] and MovieLens [Herlocker et al. 1999]. In EachMovie, it is classified as a *classic* movie, while in MovieLens it is classified as a *drama, romance* and *war* movie. In addition, a movie evaluation in EachMovie is a number between *0* and *1*, while in MovieLens it is expressed by a number of stars on a *5*-star scale. To implement mediation between these two systems, the mediator should be able to cope with such heterogeneities.

Hence, cross-representation mediation can be considered as an integration of user modeling data between heterogeneous representations of the values of the experience features. This means that although different representations of the features refer to the same underlying objects, they are expressed in different ways. As a result, the mediation is conducted between past experiences, where the values of all the components $user_{feat}$, $item_{feat}$ and $context_{feat}$ imply the same, but are represented differently. This type of mediation is divided into two groups:

- Different representation of $user_{feat}$, $item_{feat}$ and $context_{feat}$ values. This mediation deals with a situation where the representation of one (or several) of the experience components is heterogeneous. This means that although the $user_{feat}$, $item_{feat}$ and $context_{feat}$ are semantically identical and reflect the same user modeling data, one (or several) of them is (are) syntactically expressed in different ways. For example, collaborative filtering systems represent an item using its unique identifier only, while content-based systems represent the item using the set of its features. This mediation requires applying inference mechanisms identifying commonalities and dependencies between various representations of semantically identical $user_{feat}$, $item_{feat}$, or $context_{feat}$ using an external domain-specific knowledge base. Hence, this variant of cross-representation mediation is referred to as *cross-technique mediation* [Berkovsky et al. 2006b].
- Different representations of the evaluation values. In this mediation, the representations and the values of $user_{feat}$, $item_{feat}$ and $context_{feat}$ features are identical, but the evaluations of the

experiences are expressed in different ways, i.e., the heterogeneity is in the representation of the evaluations. For example, the target recommender system represents the evaluation as a discrete numeric rating on a scale between *1* and *10*. However, the remote system represents it as positive or negative evaluation only. To overcome this heterogeneity, there is a need to map and reconcile the evaluation representation values between the two scales. This case is separated from the previous one since overcoming the heterogeneity of feature representations is considered a more complicated task than the mapping between the evaluation scales.

4.2.2 Cross-User Mediation

Although the term mediation is not explicitly mentioned in collaborative filtering recommender systems, cross-user mediation and inference actually constitute the basis of this popular recommendation technology [Herlocker et al. 1999]. Collaborative filtering is based on the assumption that people with similar tastes (i.e., people who agreed in the past) will prefer similar items (i.e., will agree in the future) [Shardanand and Maes 1995]. Or, in a simplistic view, collaborative filtering recommends items liked by similar users[4]. In order to generate a recommendation, collaborative filtering systems initially create a neighborhood of users with the highest level of similarity to the active user, and then generate a recommendation by integrating the ratings of these users. Hence, this process can be considered as a cross-user inference, or a particular case of cross-user mediation, where the mediated user modeling data are the ratings of similar users. In addition, other variants of cross-user inference are applied in several existing recommendation techniques, e.g., demographic filtering [Krulwich 1997], collaborative by content recommendations [Pazzani 1999], and some hybrid approaches [Vozalis and Margaritis 2004].

It should be noted that the existing implementations of cross-user mediation in the state-of-the-art recommender systems mostly ignore the context-awareness issue. This means that they project the collected experiences onto the two-dimensional representation R'_{gen}, not reflecting the contextual conditions of the experience. Hence, these recommender systems apply inference mechanisms assuming that the collected experiences were recorded for the same contextual conditions. Thus, the prediction of the new experience evaluation can be considered as an inference process incorporating past experiences of other users for the same item in the same, actually *undefined*, context, i.e., the prediction generation process is pure cross-user inference.

[4] The reader is referred to [Helocker et al. 1999] for an discussion on collaborative filtering similarity metrics.

4.2.3 Cross-Item and Cross-Domain Mediation

Cross-item mediation is also applied in various existing recommendation techniques, such as content-based filtering [Morita and Shinoda 1994], item-to-item collaborative filtering [Sarwar et al. 2001], utility-based recommendations [Manouselis and Sampson 2004], and in some hybrid approaches [Pazzani 1999]. In general, these techniques assume that the similarity of items may also be used for providing personalized recommendations, i.e., items which are similar to the items the users liked in the past should be recommended to the users (most cross-user similarity metrics discussed in [Herlocker et al. 1999] may be applied also for computing cross-item similarity).

In these practical systems, the context-awareness issue is also mostly ignored. The collected experiences are represented using the two-dimensional representation R'_{gen}, such that the collected experiences are considered as if they were recorded in the same contextual conditions. Thus, the prediction of the new experience evaluation can be considered as the result of an inference process, incorporating past experiences of the same user for other items in the same, actually *undefined*, context. This means that the recommendation generation process is performed through cross-item inference.

To conduct cross-item mediation, the similarity of items (or, relationships between the items) should be defined for any arbitrary pair of items. However, not for any pair of items can the similarity be easily defined. For example, in item-to-item collaborative filtering [Sarwar et al. 2001], cross-item similarity is computed by means of user ratings to items. In this case, computation of the similarity between two items requires the items to be rated by a non-empty set of overlapping users. This requirement may be too strong for sparse ratings matrices. Moreover, in many conditions, the available past experiences do not necessarily reflect the user's evaluation for an individual item, but rather for a generalized group (or category) of items. For example, a user may express his opinion not on a single movie, but on a genre of movies, or on movies directed by a certain director.

Hence, in a broader view, the individual items need to be grouped. This allows a more complex type of mediation to be applied, incorporating the evaluations of past experiences for a generalized group of items [Mehta et al. 2005a]. Generalizing individual items into groups and domains and then exploiting cross-domain dependencies and inferences introduces the issue of **cross-**

domain mediation, where the evaluation of the new experience for a generalized group of items from a certain domain is inferred from past experiences for items in other domains [Berkovsky et al. 2007a]. In this sense, cross-domain mediation can be considered as a mediation incorporating past experiences of the same user in the same contextual conditions, but for another generalized group of items.

4.2.4 Cross-Context Mediation

The issue of cross-context mediation is a new research direction in user modeling. Such mediation is based on context-aware representation of the experiences, and its goal is to predict the evaluations of the new experiences in a given context using past experiences in other contextual conditions [Berkovsky et al. 2006c]. This means that cross-context mediation incorporates past experiences of the same user for the same item in other contextual conditions. For example, it can predict future evaluation for an item by a user in the *evening* given past evaluation of the same user for the same item in the *morning*, or, it can predict future evaluation for an item by a user when accompanied by a group of friends given past evaluations of the same user for the same item when accompanied by a parent.

Since the state-of-the-art recommender systems mostly ignore the context-awareness aspect and are not capable of providing context-aware recommendations, this type of mediation requires the definition of various novel cross-context reasoning mechanisms. Two simple mechanisms, exploiting semantically enhanced OWL [McGuinness and van Harmelen 2004] representations of $user_{feat}$, $item_{feat}$ and $context_{feat}$, were discussed in [Berkovsky et al. 2006c]:

- **Rule-Based Reasoning.** This reasoning mechanism exploits the semantically-enhanced representations of the experience components for the purposes of defining a set of reasoning rules that exploit the relationships between the values of the features. For example, consider a semantic representation of times of day, and a reasoning rule defining that user's preferences regarding a certain item (e.g., stocks news report) in the *evening* are opposite to preferences in the *morning*. Or, consider another rule based on the same semantic representation of times of day, which defines a projection of user's preferences at *4PM* onto a more general *afternoon* time period. Applying these rules facilitates the inference of the required user modeling data across various contextual conditions.

- **Similarity-Based Reasoning**. This reasoning mechanism exploits the semantically-enhanced representations of the experience components for the purposes of defining an explicit similarity metric, capable of computing the similarity between any arbitrary pair of contextual conditions. For example, such metric may express similarity between Tuesday and Wednesday as mid-week days and dissimilarity between Tuesday and Sunday as mid-week and week-end days. Such cross-context similarity metric allows various adaptation rules to be derived, similar to the rules used in Case-Based Reasoning [Aamodt and Plaza 1994; Ricci et al. 2006b; Ricci et al. 2006c], and facilitates reuse of the evaluations of past experiences. For example, this can be done by a collaborative-like weighted aggregation of the evaluations of past experiences of the same user for the same item in similar contextual conditions.

Comparing the above discussed rule-based and similarity-based reasoning approaches shows that, on the one hand, rule-based reasoning may produce more accurate user modeling data, as the reasoning rules are typically defined by domain experts. On the other hand, defining and updating the inference rules in today's highly dynamic information world may hamper the scalability of the mediation process. Conversely, the typical scenario for similarity-based reasoning is fully autonomous and therefore gives a more flexible mediation process. However, similarity-based reasoning requires a large number of past experiences to bootstrap the reasoning process. Other machine learning approaches can be considered for this purpose.

Part 3:

Experimental Evaluation

Chapter 5: Cross-Representation Mediation of User Modeling Data

The main goal of the UM mediation is to acquire UMs collected by other recommender systems, and to convert and consolidate them into a single UM, as needed by the target recommender system. Earlier analysis yielded the definition of *experience* as the core user modeling data representation unit, referring to the connection of three components: *user*, *item*, and *context*. Based on this representation, there are four groups of experiences that may be valuable for the mediation process:

- Experiences of the same user for the same item in the same context, where certain experience components may be represented in different ways.
- Experiences differing only in one component. These include experiences of another user for the same item in the same context, experiences of the same user for another item in the same context, and experiences of the same user for the same item in another context.
- Experiences differing in two components. These include experiences of the same user for another item in another context, experiences of another user for the same item in another context, and experiences of another user for another item in the same context.
- Experiences where the values of all three components are different, i.e., experiences of another user for another item in another context.

This section deals with the first group of experiences, i.e., experiences of the same user for the same item in the same contextual conditions, where some of the experience components may be represented in different ways. Clearly, such experiences are the most valuable for the UM mediation, as they provide experiences of the target user for the required item in the relevant context. However, their mediation requires resolving possible heterogeneities in the representations of the experience components, e.g., structural heterogeneity, use of synonyms for description of the same concept, or multilingualism [Bernstein and Melnik 2004]. Hence, the mediation of such experiences requires applying inference mechanisms, capable of identifying the relationships between the heterogeneous representations of the experience components and projecting the available user modeling data for generation of new recommendations. This mediation is henceforth referred to as *cross-representation mediation*.

Specifically, this section focuses on a particular type of cross-representation mediation from the collaborative filtering to the content-based recommender system. In collaborative filtering systems, the UMs are represented by vectors of explicit ratings provided by the users on a set of items managed by the system [Herlocker et al. 1999]. Conversely, in content-based systems, the items are represented by the values of features characterizing the items, and the UMs are typically represented by weights of these features, representing the degree of the user's preference for these features [Morita and Shinoda 1994]. Mediation between these types of UMs requires the identification of regularities among the features of positively or negatively rated items, where the ratings on the items are derived from the collaborative filtering UM. As no item features are originally stored by the collaborative filtering recommender systems, the features describing the rated items should be extracted from an external domain knowledge base. Then, the user features are assigned numeric values depending on the ratings given by the user, and a set of ratings in the collaborative filtering UM is generalized into the weighted list of features liked/disliked by the user, as needed by the content-based UM.

The rest of this section is organized as follows. Section 5.1 describes the proposed approach to cross-representation UM mediation and elaborates on the conversion of collaborative filtering UMs to content-based UMs. Section 5.2 presents the conducted experimental evaluations and discusses their results. Finally, Section 5.3 summarizes the section.

5.1 Collaborative Filtering to Content-Based Mediation

Collaborative filtering is probably one of the most widely used recommendation techniques. It recognizes cross-user correlations and generates recommendations for items by weighting the opinions of similar users [Herlocker et al. 1999]. Hence, a collaborative filtering algorithm is typically partitioned into three general stages: (1) Similarity Computation: weighting all the users with respect to their similarity with the active user (i.e., the user who requested the recommendation); (2) Neighborhood Formation: selecting a set of the most similar users for the recommendation generation; and (3) Recommendation Generation: computing the recommendation by weighting the ratings of the selected users on the required item. In other words, collaborative filtering systems recommend items that were liked in the past by other users, similar to the active user.

The input for the collaborative filtering is a matrix of users' ratings on a set of items managed by the system. In this matrix, each row represents the ratings of a single user and each column represents the ratings on a single item. Thus, collaborative filtering UMs are represented as *ratings vectors*, i.e., a fixed-size list of pairs $UM_{CF}=\{i_1:r_1, i_2:r_2, \ldots, i_n:r_n\}$, where every pair $i_k:r_k$, corresponds to a rating r_k provided explicitly by the user on an item i_k. If a user's rating on an item is not available then a special *null* value is used. In fact, the UMs in collaborative filtering systems typically store ratings for only a very small subset of the items managed by the system. Moreover, collaborative filtering systems typically do not store any item- or user-related content features, besides their unique identities.

Content-based filtering [Morita and Shinoda 1994] builds personalized recommendations by taking as input: (1) a list of features describing the contents of the items in a given domain, possibly weighted according to a predefined scale; (2) a set of weights assigned by the user to the above list of features, possibly derived from the user's ratings on the items; and (3) the set C of available items, which have not yet been rated by the user, i.e., the items that are candidates for the recommendations. The output recommendation is a subset of C, containing the items whose features match the features that were preferred by the user. In other words, content-based systems recommend items, similar to the items that were positively rated in the past by the active user.

Thus, in content-based recommender systems the UMs are represented as a list $UM_{CB}=\{f_1:w_1, f_2:w_2, \ldots, f_n:w_n\}$, where f_k denotes one of the application domain features and w_k denotes the level of the user's preference regarding this feature. It should be noted that the information about a user's preferences, which is contained in the ratings on the items, is typically transferred into the feature weights using various machine learning techniques, e.g., Winnow [Littlestone 1988], or by computing the centroid of the feature-vector representation of the items liked by the user [Billsus and Pazzani 2000]. Note the heterogeneity of the content-based UM representation, as the features of the items are typically largely dependent on the application domain of the recommender system. For example, features useful for a music recommender system will not be very useful for a travel recommender system, and vice versa. Even within the same application domain, the features may vary between one recommender system and another. Moreover, the repre-

sentation of the user's preference may also vary across different systems. For example, it may be only a like/dislike expression or a numeric value between *0* and *1*.

This section aims at developing a mediation mechanism capable of converting the collaborative filtering UMs, represented by a set of ratings explicitly given by a user, to content-based UMs, represented by a set of content-related features and their corresponding weights. The rest of this section describes the cross-representation UM mediation applied in the application domain of movies. First, it presents the UM mediation mechanism, and second, it discusses the details of the content-based prediction mechanism fine-tuning. Although the following discussion focuses on the domain of movies, it should be stressed that the proposed mediation approach can be applied in a similar manner also for other application domains.

5.1.1 User Models Conversion and Content-Based Recommendations

For the movies domain, a collaborative filtering UM comprises a set of movies and their respective ratings, explicitly provided by the user. For example, consider the following sample UM_{CF}*={"The Lord of The Rings":1, "The Matrix":0.8, "Psycho":0.2, "Friday the 13th":0, "Star Wars":0.9, "The Nightmare on the Elm Street":0.1", "Alien":0.9}*, where the movie ratings are given on a continuous numeric scale ranging between *0* and *1*. Although this collaborative filtering UM represents the user with a set of ratings only, it can easily be recognized that the user likes science-fiction movies, and dislikes horror movies. Hence, the content-based UM of this user may be UM_{CB}*={science-fiction:0.9, horror:0.1}*, where the genre weights are computed as an average of the ratings given on the movies from this genre. Similarly as for the genres, the weights of other features of the movies (e.g., directors, producers and actors) can also be computed using the ratings of movies, which include the features. This process will be fully illustrated in the rest of this section.

To handle the translation of collaborative UMs into content-based UMs, a rich movie knowledge base is needed, from which the features of the movies, such as the lists of genres, actors, directors, and so forth, can be extracted. In this work, an offline version of the IMDb movie database [IMDb 2007] was downloaded from the Web and served as the translation knowledge base. The IMDb provides movie information from *49* feature categories, such as *genres*, *actors*, *directors*, *writers*, *cinematographers*, *composers*, *keywords*, *languages*, and many others. For the sake of

simplicity, only *7* feature categories were initially extracted in this work: *genres, keywords, actors*, *actresses*, *directors*, *production countries* and *languages*, as these categories seem to have the strongest effect on the user's decision in selecting, seeing, and rating a movie [Tintarev 2007].

Conversion of collaborative filtering UMs to content-based UMs receives the user's ratings vector as input. It should be stressed that a certain numeric rating assigned by a user to a movie is not an objective, but rather a subjective, i.e., relative value, depending on the user's expression style, emotional characteristics, and so forth. For example, consider two ratings of *3*: one provided by a user, whose average rating is *2*, and another provided by a user whose average is *4*. Clearly, the former is an expression of a positive opinion, while the latter is of a negative opinion. Hence, the values of the ratings should be normalized in order to eliminate the effect of users' individual tendencies of expression. This is done by subtracting the average rating of the user from the values of each one of the provided ratings:

$$r'_i = r_i - r_{av}$$

where r_i denotes the original value of the rating on the movie, r'_i denotes the normalized rating, and r_{av} denotes the average rating of the user computed over all the available ratings provided earlier by this user. Hence, when a movie is assigned a rating above the user's average rating, it is treated as a positive rating. Conversely, when a movie is assigned a rating under the user's average rating, it is treated as a negative rating.

The main assumption behind the collaborative filtering to content-based UM mediation is that the users' ratings for the movies implicitly reflect their preferences regarding certain features of the movie, such as movie genre, director, or actors. However, a single rating cannot reliably determine the exact set of features of the movie which are preferred by the user. Hence, the rating given by the user is projected onto all the features of the movie. This means that for each movie rating in the collaborative filtering UM, the lists of the movie features from the above *7* categories are extracted from the IMDb and the weights of the features are updated according to the normalized movie rating r'_i. In other words, the weights of all the actors and directors involved in the movie (and in a similar way, of all the features from the rest of the categories) are updated according to the normalized movie rating r'_i. The exact way to update the feature weights will be described later in this section.

For example, consider a rating *"Star Wars":0.9*, given by a user whose average rating is *0.6*. Clearly, this is a positive rating and the normalized value of such rating is *0.3*. According to the data extracted from the IMDb, the *genres* of *"Star Wars"* are *action*, *adventure*, *fantasy* and *science-fiction*. Thus, the existing weights of these four features are updated accordingly, i.e., increased by a positive rating of *0.3*. If the weight of one of the features is unknown, it is assigned the normalized value of a rating. Similarly to the movie *genres*, also the weights of the movie *director George Lucas*, of all the *actors*, and actresses involved in the movie, and of all the other features are increased by *0.3*.

Inherently, the weights of the frequently occurring features are higher than the weights of the infrequent features. In order to balance the stronger influence of the frequently occurring features, the frequency of each feature (i.e., the number of movies rated by the user and including the feature) is recorded. Hence, in the above *"Star Wars"* example, the frequency of the above four movie genres, for *George Lucas*, and of all the actors and other movie features is increased by one.

One could hypothesize that using the normalized movie rating to increase the weights of all the features of the movie could be inaccurate, as the motivation for the user's positive evaluation of the movie should reside on a subset of the features, i.e., some of the movie features may not be related to the value of the rating. However, this should be balanced by the fact that the collaborative filtering UM typically comprises a large number of ratings. Hence, the features that are not preferred by the user will be assigned in some movies positive ratings and in some movies negative, i.e., their overall rating will be neutral. Conversely, strongly liked (or disliked) features will consistently get positive (or negative) ratings, and therefore their overall weight will be easily recognized.

After the content-based UM is generated, the user is modeled as a set of weights $\{w_{i(1)}, \ldots, w_{i(k)}\}$ for k features available in the *7* categories, and the corresponding feature frequencies $\{c_{i(1)}, \ldots, c_{i(k)}\}$. The exact value of k depends on the number of movies rated by the user and the number of features available for these movies in the above *7* categories. For example, the number of features available for *"Star Wars"* in the above *7* categories is *213*, whereas for *"Psycho"* it is only *116*. Clearly, the overall number of features will increase with the number of movies rated by a user.

Given a movie *m*, which has not yet been rated by the user, a predicted rating for *m* is then generated by (1) extracting all the relevant features of *m* from the IMDb, and (2) computing the movie recommendation as a weighted average of the weights of the features that are both in the content-based UM and in the movie description:

$$prediction(m) = \frac{\sum_{j \in F(u) \cap F(m)} w_i c_i}{\sum_{j \in F(u) \cap F(m)} c_i}$$

In this formula *F(u*) denotes the set of features in the content-based UM and *F(m*) denotes the set of features in the movie model. If the system should recommend only one movie, then a separate recommendation is generated for every movie and the movie with the highest predicted rating is recommended to the user. If a set of movies should be recommended, then the movies are sorted according to their predicted values and top-*N* (the value of *N* depends on the system constraints) are recommended to the user.

Note that the recommendations are generated based solely on content-based UM, which is derived from the original collaborative filtering UM. As such, the prediction mechanism is capable of building content-based recommendations regardless of the number of ratings available for the given movie. Hence, being a pure content-based recommendation approach, the proposed mediation allows resolving the *new item problem* [McNee et al. 2003], typical of the collaborative filtering recommender systems, where accurate recommendations for an item cannot be generated unless the system obtains a sufficient number of ratings on this item. Nevertheless, being a pure content-based recommendation approach, it suffers from the well-known *serendipity problem* [McNee et al. 2006], typical to content-based recommender systems, where a system can recommend only movies that are similar to the movies already rated by the user and cannot provide 'surprising' recommendations for new types of movies. Hence the serendipity of the generated recommendations largely depends on the exact similarity function implemented by the above discussed rating prediction mechanism.

5.1.2 Fine-Tuning of the Prediction Mechanism

Although the proposed mechanism is capable of generating recommendations regardless of the number of available ratings on a movie, it may suffer from instability (i.e., undesired fluctuations effected by minor factors). Since the IMDb contains a large amount of content information for

each movie, content-based UMs built from collaborative UMs containing only a few ratings already include thousands of features. When the number of rated movies in the collaborative filtering UM increases, the overall number of features in the generated content-based UM increases with a higher order of magnitude. Some of these features may be important as they really reflect user's preferences, while some may be irrelevant, i.e., *noisy* features, which hamper the accuracy of the generated recommendations. Two issues should be resolved in order to improve the accuracy of the generated recommendations: (1) identification of features that insert noise into and hamper the accuracy of the prediction mechanism, and should be ignored by the prediction mechanism, and (2) identification of categories of features that are important for the recommendation generation.

The first issue deals with determining the important features that should be taken into account by the prediction mechanism and the noisy features that should be ignored. Clearly, for a content-based UM containing tens thousands of features, most of the features are irrelevant to the user's rating on a movie. Although the content-based recommendations are generated as a weighted average of the weights of the features in the movie description, a large number of such irrelevant features may become a dominant factor, *overshadowing* the important features and hampering the accuracy of the generated recommendations. Two categories of irrelevant features should be distinguished:

- *Low-frequency features*. These are features that have a small number of occurrences c_i among the movies rated by the user. In the domain of movies, these features typically represent extra actors or actors playing marginal roles in the movies. Although their frequency (i.e., also their weight in the weighted average in the rating recommendation) is low, the number of such features increases quickly with the number of rated movies in the collaborative filtering UM. Hence, a large number of such features may outweigh the important features and hamper the accuracy of the generated recommendations.
- *Neutral features*. These are the features, to which the user is indifferent, i.e., features of no special importance to the user's rating on a movie. As the user is indifferent to the neutral features, they are sometimes assigned positive and sometimes negative values. Hence, when the content-based UM is generated, the weight w_i of the neutral features is supposed to be close to *0*, regardless of their frequency c_i. Similarly to the low-frequency features, although the weight

of the neutral features is low, a large number of such features may outweigh the important features and hamper the accuracy of the generated recommendations.

To minimize the impact of the above two types of irrelevant features, two thresholds were defined. The *min-occurs* threshold denotes the minimal frequency c_i of a feature, for which the feature is taken into account by the prediction mechanism. It is designed to eliminate the influence of the *low-frequency* features by considering only the features whose frequency is above the *min-occurs* threshold. The *conf* feature denotes the minimal absolute value of the weight of a feature, for which the feature is taken into account by the prediction mechanism and reflects the confidence in the user's preference of a feature. It is designed to eliminate the influence of the neutral *features* by considering only the features, where the absolute value of the feature weight is above the *min-occurs* threshold.

The second issue deals with determining the important feature categories, and it can be resolved using a feature selection approach [Kohavi and John 1997]. Feature selection is defined as follows: "given an inducer *I* and a dataset *D* with a set of features $\{X_1, X_2, \ldots, X_n\}$, an optimal feature subset $\{X_{i1}, X_{i2}, \ldots, X_{im}\}$, where $m \leq n$ is a subset of the features such that the accuracy of the induced classifier *I(D)* using this set of features is maximal". In our case, the inducer *I* is the content-based prediction mechanism and *D* is the IMDb with $n=7$ feature categories: *genres*, *keywords*, *actors*, *actresses*, *directors*, *production countries* and *languages*.

Most of the feature categories contain a large number of possible features. Therefore, instead of addressing the problem of selecting independently the best features from each category, this work focuses on the selection of relevant categories of features. In other words, the search process is limited to a subset of all the possible subsets of features, where the features from a certain feature category are either all present or all absent. The basic motivation for such simplification is the large number of features, here more than *50.000*, which makes a full search of the best features subset practically infeasible. Hence, the goal of the feature selection is to select the feature categories that should be taken into account by the content-based prediction mechanism for the recommendation generation, while ignoring the other categories.

The wrapper feature selection approach [Kohavi and John 1997] conducts the selection as an automated search in the space of states, where each state consists of *n=7* bits, representing a certain combination of categories that are taken into account for the recommendation generations. For example, consider a state *S* represented by *genres=1*, *keywords=0*, *actors=1*, *actresses=1*, *directors=0*, *production countries=0* and *languages=0*. This means that the categories of *genres*, *actors*, and *actresses* are taken into account, while *keywords*, *directors*, *production countries* and *languages=0* are ignored for the recommendation generation. For the sake of clarity, we keep the above order of feature categories fixed and denote the states by their respective binary vectors, e.g., *S=(1,0,1,1,0,0,0).* Figure 6 shows the search space for *n=3* as a tree of states, where the edges indicate insertion or deletion of a certain feature category.

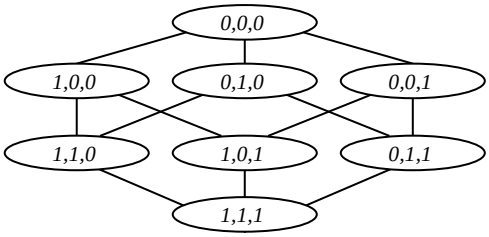


Fig. 6. Feature Selection Search Space

As can be seen, the size of the search space is $O(2^n)$ states. The goal of the search is to find the state having the greatest inducer evaluation accuracy, i.e., the greatest generated recommendations accuracy. Since the size of the search space is $O(2^n)$ and the task of evaluating the accuracy of the recommendations in each state is relatively expensive, applying an exhaustive search is impractical. Hence, a heuristic search over the space of states is applied. In this work, the hill-climbing, relatively simple greedy search algorithm was applied [Russell and Norvig 1995]. The pseudo-code shown in Figure 7 describes the stages of the hill-climbing heuristic search:

```
Hill-climbing (Initial-state s, Evaluation-function eval)
(1) let v=s
(2) expand v: generate all v's children states
(3) compute eval(w) for each child w of v
(4) let v'= the child w with the highest eval(w)
(5) if eval(v')>eval(v)
(6)   v=v'
(7)   goto (2)
(8) return v
```

Fig. 7. Hill-Climbing Heuristic Search for the Feature Selection

The algorithm starts with the initial search state, representing the initial combination of features categories, as the current node (stage *1*). It then expands the current node by generating its children states, i.e., by either adding or removing feature categories (stage *2*), and evaluates each one of the children states by computing the accuracy of the recommendations for the respective combination of features categories (stage *3*). Then, the child state with the highest evaluation value is chosen as the current state (stage *4*), and the algorithm iteratively repeats the stages of the current state expansion and child evaluation until the evaluation values of the current state improve (stages *5*-*7*). Finally, the algorithm returns the state with the highest evaluation value, i.e., the combination of feature categories, where the accuracy of the generated recommendations was maximal (stage *8*).

Hence, the prediction mechanism is modified to take into account only the features, from the categories selected by the feature selection, which occur at least *min-occurs* times in the content-based UM, and whose weight is above *+conf* or under *-conf*. Since a feature weight w_i heavily depends on the frequency of the feature c_i (higher frequency allows a greater weight to be accumulated), the *normalized weight* of the features *norm-*w_i is computed by dividing the weight of a feature w_i by its frequency c_i. The pseudo-code shown in Figure 8 describes the details of the fine-tuned recommendation generation process:

```
Recommend (Content-based-UM u, set-of-movies M)
(1)  foreach m∈M
(2)    retrieve F(m)=set of m features from the categories
                     chosen by the feature selection
(3)    for each j∈F(m)
(4)      if j∈F(u) AND |norm-w_j|>conf AND c_j>min-occurs
(5)        take j into account for recommendation of rating(m)
(6)    compute rating(m)
(7)  return m with maximal predicted rating(m)
```

Fig. 8. Fine-Tuned Content-Based Recommendation Generation

The goal of the above algorithm is to recommend a movie among a set of potentially recommendable movies. Hence, it generates separate rating recommendation for each one of the movies (stage *1*). For this, the set of movie features from the categories chosen by the feature selection is extracted from the IMDb, and the weights of these features (if present) in the content-based UM are determined (stage *2*). Each one of the features is taken into account by the prediction mechanism only if its frequency c_i is above the *min-occurs* threshold and the absolute value of its normalized weight $|w_i|$ is above the *conf* threshold (stage *3*-*5*). Finally, the predicted rating of each

recommendable movie is computed (stage *6*) and the movie with the highest predicted ratings is recommended to the user (stage *7*).

It should be noted that the proposed prediction mechanism assigns equal weights to the features from different categories and incorporates no additional weighting factor that reflects the importance of a certain category for the user. However, it is not uncommon that several categories are the most important criteria influencing a user's rating on a movie. For example, a *director* of the movie may be very important for the user, while the *actors*, *actresses*, and the rest of the feature categories may be less important. Hence, the *directors* category should be assigned a greater weight than the rest of the categories. Moreover, in real-life situations, the user's ratings may depend on a certain combination of features from several categories. For example, the user may like *science-fiction* movies, directed by *George Lucas* and dislike *adventure* movies directed by him, or even like *science-fiction* movies in *English* directed by *George Lucas* in the *USA* between *1975* and *1980* only.

Although the above discussion highlights the importance of the category weighting and discovering the dependencies between various features, the current section focuses on the task of feature categories selection. This restriction is reasonable, since after the categories selection is completed, the fine-grained weights of the specific features within the categories are computed. The other weighting issues remain beyond the scope of this work.

5.2 Experimental Evaluation

The above collaborative to content-based mediation of UMs was tested on the publicly available EachMovie dataset [McJones 1997]. EachMovie is a collaborative filtering dataset, storing *2,811,983* ratings of *72,916* users on *1,628* movies. The ratings are given on a discrete scale between *0* and *1*, such that the possible ratings are *0.0*, *0.2*, *0.4*, *0.6*, *0.8* and *1.0*. In addition, EachMovie provides for every movie a URL to the movie description in the IMDb. Some of the URLs were outdated or invalid, such that in the experimental evaluation we could use only a set of *1,529* movies, whose URLs were identified as valid in the IMDb. For this set of movies, we identified a set of *47,988* users who rated more than *10* movies, such that the variance of their ratings

is not *0* (i.e., not all their ratings are identical). Hence, we obtained an overall number of *2,667,605* ratings, producing a relatively sparse dataset with a density of *3.64%*.

We analyzed the distribution of users in the dataset according to the number of movies rated by them. For this, we partitioned all the available users into *13* groups, where the numbers of rated movies are: fewer than *25*, *26* to *50*, *51* to *75*, *76* to *100*, … , *201* to *225*, *226* to *250*, *251* to *300*, *301* to *500* and over 500 movies. Table 3 shows the distribution of the number of rated movies among the users. As can be clearly seen from Table 3, most of the users in the dataset rated a relatively low number of movies. For example, *64.83%* of the users rated fewer than *50* movies, *78.4%* of the users rated fewer than *75* movies, and *85.92%* of the users rated fewer than *100* movies.

Table 3. Distribution of Ratings among the Users in the Dataset

rated movies	**fewer than *25***	***26* to *50***	***51* to *75***	***76* to *100***	***101* to *125***	***126* to *150***	***151* to *175***	***176* to *200***	***201* to *225***	***226* to *250***	***251* to *300***	***301* to *500***	**over *500***
number of users	*17,321*	*13,788*	*6,514*	*3,609*	*2,302*	*1,349*	*887*	*609*	*441*	*327*	*358*	*436*	*47*
% of users	*36.09*	*28.73*	*13.57*	*7.52*	*4.80*	*2.81*	*1.85*	*1.27*	*0.92*	*0.68*	*0.75*	*0.91*	*0.098*

For the collaborative filtering to content-based mediation of UMs, we used an offline version of the IMDb dataset. Although the IMDb provides movie information from *49* feature categories, in this section the conversion mechanism was restricted to only *7* feature categories: *genres*, *keywords*, *actors*, *actresses*, *directors*, *production countries* and *languages*. We believe the features from these categories have the strongest effect on the user's decision when selecting and rating a movie[5].

To analyze the statistical properties of various feature categories, we computed for each category the overall number of features that occur in the descriptions of the above *1,529* movies. Table 4 shows the number of features in every feature category.

Table 4. Number of Features in Features Categories

category	**genres**	**keywords**	**actors**	**actresses**	**directors**	**countries**	**languages**
number of features	*24*	*8,993*	*29,543*	*14,139*	*1,111*	*60*	*73*

[5] Many feature categories, such as *writers*, *composers*, *awards* and others, were left out in this evaluation.

Typically, a movie belongs to several *genres*, *production countries* and *languages*. Hence, the overall number of features in these categories is relatively small. Conversely, as the number of *keywords*, *actors*, *actresses*, and *directors* for every movie may be high, the overall number of features in these categories is significantly higher.

The first part of the experiments (i.e., fine-tuning of the thresholds and the feature selection) was conducted on a smaller dataset, henceforth referred to as the *FT* set. The *FT* set is a fixed set of *1,000* users who rated at least *100* movies, where every user rated on average *168.94* movies. For each user in the *FT* set, *90%* of the ratings were defined as the training set and the remaining *10%* as the test set, such that the training and testing stages of the experiments using the *FT* set were conducted on completely disjoint sets. The ratings in the training set served as input for the collaborative filtering to content-based UMs conversion mechanism. Then, the generated content-based UM was used for generating pure content-based recommendations for all the items in the user test set. Hence, the overall number of the generated recommendations in the experiments conducted over the FT set was *16,894*.

The accuracy of the generated content-based recommendations was evaluated using the commonly used MAE metric [Herlocker et al. 2004]:

$$MAE = \frac{\sum_{i=1}^{N} | p_i - r_i |}{N}$$

where N denotes the overall number of the generated recommendations, p_i denotes the computed and r_i denotes the real value of the rating for the item in the recommendation number i.

Note that the results of the ratings recommendation computation could be an arbitrary real number between *-1* and *1*, whereas the ratings in EachMovie dataset are discrete values between *0* and *1*. To allow correct computation of the MAE, the ratings in EachMovie were re-mapped to the required range between *-1* and *1*: ratings of *0.0* were converted to *-1*, ratings of *0.2* were converted to *-0.6*, … , ratings of *0.8* were converted to *+0.6* and ratings of *1* remained unchanged.

5.2.1 Feature Selection and Setting the Thresholds

The aim of the first set of experiments was to fine-tune the prediction mechanism and it consisted of two tasks: (1) selecting the most appropriate values for the *conf* and *min-occurs* thresholds to

filter accurately the irrelevant features that have a minor or wrong effect on the user's rating on a movie, and (2) applying the feature selection wrapper approach for selecting the feature categories that should be taken into account by the prediction mechanism.

The goal of the first fine-tuning task was to determine the most appropriate values for the *conf* and *min-occurs* thresholds. Since the two thresholds are independent, to determine their most appropriate values, one of the thresholds was set to a constant value, while the values of the second were gradually modified. For each value of the modified threshold, we used the *FT* set, where the *90%* training set served as input for the collaborative filtering to content-based UMs conversion mechanism. Then, the generated content-based UMs were used for generating pure content-based recommendations for the items from the *10%* test sets. For each value of the *conf* and *min-occurs* thresholds, the accuracy of the recommendations using the given threshold values was also evaluated using the MAE metric [Herlocker et al. 2004].

To find the most appropriate value of the *conf* threshold, the *min-occurs* threshold was set to *min-occurs=1* movies for all the categories (i.e., a feature should occur at least in *1* movie rated by the user), and the values of *conf* threshold (i.e., the minimal absolute value of the weight of a feature, for which the feature is taken into account by the prediction mechanism) were gradually increased from *0* to *0.5*. To provide an initial indication of the different relative importance of different categories, the recommendations were generated in two ways: (1) using all the available *7* feature categories, and (2) using only *6* feature categories, i.e., using all the available categories, excluding the *keywords*. We note that high values of the *conf* threshold reduce the number of features taken into account by the content-based prediction mechanism. Hence, the predicted ratings for certain movies cannot be computed. To check this effect of the *conf* threshold, for each value of *conf*, we computed the prediction rate, i.e., the percentage of movies whose ratings were successfully computed. Figure 9 illustrates the results of the experiments. The horizontal axis shows the values of the *conf* threshold, and the vertical shows the values of the MAE and of the prediction rate. The dotted curves show the prediction rate values and the continuous ones the MAE. The dark curves show the results based on all the available categories, while the light curves are based on all the categories, excluding the *keywords* features.

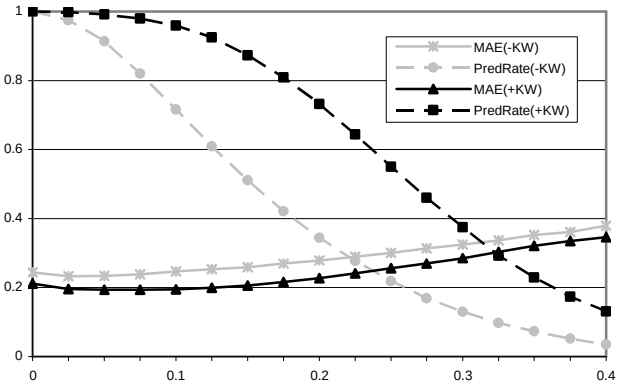


Fig. 9. MAE Values and Prediction Rate vs. *conf* Threshold

To understand the results of the experiment better, we computed for every value of the *conf* threshold the percentage of features that were filtered. In other words, we divided the number of features that were not considered by the prediction mechanism by the overall number of features in the content-based UMs of the users in the *FT* set. Table 5 shows the percentage of filtered features for various values of the *conf* threshold.

Table 5. Percentage of Filtered Features for Various Values of *conf* Threshold

conf	***0.0***	***0.05***	***0.1***	***0.15***	***0.2***	***0.25***	***0.3***	***0.35***	***0.4***
% of filtered features	*0.0*	*28.94*	*49.37*	*64.88*	*74.97*	*83.68*	*88.96*	*92.73*	*95.13*

The results clearly show that the MAE values initially slightly decreased with the *conf* threshold, and then started monotonically increasing. These results can be explained by considering the data presented in Table 5. The initial decrease of the MAE is explained by the effect of the neutral features on the accuracy of the recommendations. If the *conf* threshold is *0* and the neutral features are not filtered, they insert noise into the prediction mechanism and the MAE is higher. When the *conf* values started increasing, a high number of neutral features was filtered (e.g., *28.94%* of features were filtered for *conf=0.05* only), and the MAE decreased. However, for even higher values of the *conf* threshold, a very high number of neutral features was filtered (e.g., *74.97%* of features were filtered for *conf=0.2*), and also the important features were filtered. As a result, the MAE values started increasing with the *conf* threshold.

As for the prediction rate, it monotonically decreased with the *conf* threshold. This behavior is explained by the observation that the number of neutral features filtered from the content-based UM monotonically increases with the values of the *conf* threshold, as clearly shown by Table 5.

As a result of such filtering, the number of features that remain in the content-based UM decreases with the *conf* threshold. This, in turn, decreases the number of features that overlap between the UM and the model of the predicted movie and aggravates the task of rating recommendation generation. To reflect the importance of both the MAE and the prediction rate, *conf=0.025* was determined as the most appropriate *conf* value, since for this value the MAE is minimal and the prediction rate is still high (over *0.99*).

We would like to stress the difference between the experimental results including and excluding the *keywords* feature category in the recommendation generation process. Both the metrics of the MAE and of the prediction rate improved when the *keywords* features were taken into account. This is a particular example motivating the feature selection and showing that the *keywords* feature category is an important category. It is beneficial when the prediction mechanism takes this category into account, as it improves both the accuracy of the recommendations and the number of recommendations that can be successfully generated by the prediction mechanism.

As the most appropriate value of *conf=0.025* was determined, it was applied to choosing the most appropriate value of the *min-occurs* threshold (i.e., the minimal number of occurrences of a feature, for which the feature is taken into account by the prediction mechanism)[6]. To do this, a similar methodology was used: the value of the *conf* threshold was set to *0.025*, and the values of the *min-occurs* threshold were gradually modified to determine the most appropriate threshold value. We used again the *FT* set, and converted the ratings in the *90%* training set from the collaborative filtering to content-based UMs. The generated content-based UMs were used for generating pure content-based recommendations for the *10%* test set. The accuracy of the recommendations using the given threshold values was evaluated using the MAE metric.

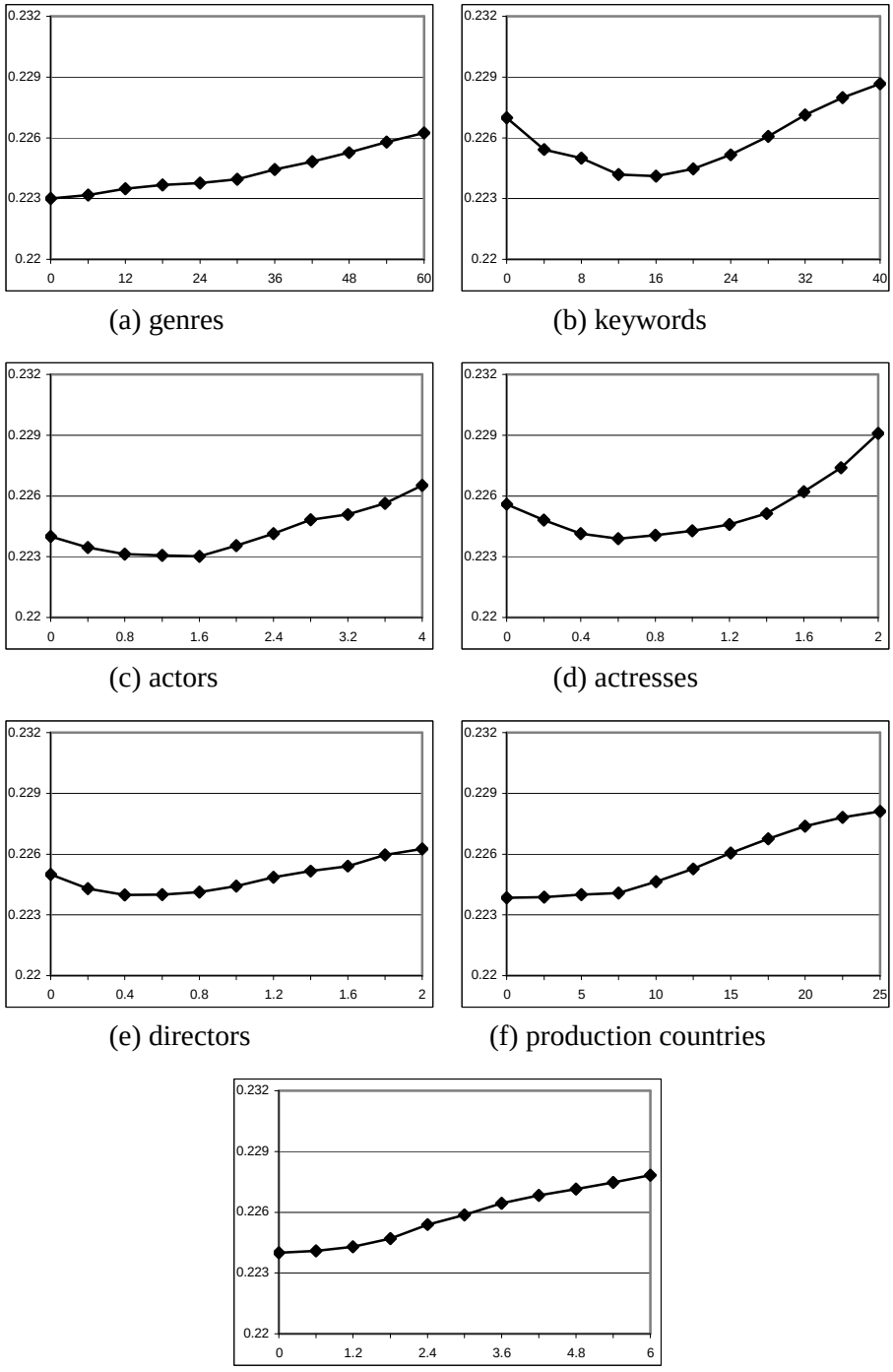
We would like to stress that due to the differences in the frequencies of the features (which, in fact, depend on the category of the feature as shown by Table 4), a separate *min-occurs* threshold experiment was conducted for each feature category. For example, the frequency of the feature *science-fiction* in the *genres* category is significantly higher than the frequency of the feature *George Lucas* in the *directors* category. This is explained by the fact that the number of features within the *genres* category, i.e., *24*, according to Table 4, is significantly smaller than the number

of features in the *directors* category, i.e., *1,111*. Since every movie typically belongs to several *genres* and some are directed by a few *directors*, the frequencies of the *genres* feature are significantly higher than the frequencies of the *directors* features. Since no uniform scale for the possible frequencies of the features could be derived, we conducted separate experiments for the *7* feature categories.

To determine the *min-occurs* thresholds for the categories, we assumed that the thresholds are independent. Hence, we isolated the effect of the *min-occurs* threshold in every feature category by changing only one threshold in every experiment, i.e., the values of *6* out of *7* thresholds were fixed to *0* and the values of *1* threshold were gradually increased. The recommendations were generated based on the features from all the categories, and the MAE values were computed as a function of the *min-occurs* threshold. Figure 10 illustrates the results of the experiments for the *7* feature categories. Note that due to the different scales of the feature frequencies, the *min-occurs* threshold values are represented in Figure 10 by relative (percentage) and not absolute (numeric) values. Hence, in all the charts, the horizontal axis shows the percentage threshold of the rated movies in the collaborative filtering UM containing the given feature category, while the vertical shows the values of the MAE.

As can be clearly seen, for most feature categories the impact of the *min-occurs* threshold is not as strong as that of the *conf* threshold. However, two different tendencies, corresponding to two different types of feature category, can be observed in the MAE behavior. In the first type, which includes the *genres*, *production countries* and *languages* categories, the numbers of possible features are relatively low (*24* genres, *60* countries and *73* languages). For this category, the MAE values monotonically increase with the *min-occurs* threshold. Thus, any feature from this category is valuable and filtering features from this category hampers the accuracy of the recommendations. Hence, the most appropriate value of the *min-occurs* threshold for this category is *0*.

[6] In this experiment, the value of the *conf* threshold was first determined, and then applied for the *min-occurs* thresholds. Repeating the experiment in the opposite order (determining first the *min-occurs* thresholds and applying them for the *conf* threshold) produced similar results.



(a) genres (b) keywords

(c) actors (d) actresses

(e) directors (f) production countries

(g) languages

Fig. 10. MAE Values vs. *min-occurs* Threshold for: (a) *genres*, (b) *keywords*, (c) *actors*, (d) *actresses*, (e) *directors*, (f) *production countries*, and (g) *languages* categories

In the second type of category, which includes the *keywords*, *actors*, *actresses*, and *directors* categories, the number of possible features is significantly higher (*8,993* keywords, *29,543* actors, *14,139* actresses and *1,111* directors). For these categories, the MAE values initially decrease with *min-occurs*, i.e., the noisy features are being filtered. This happens until the most appropriate values of the *min-occurs* threshold is reached. Afterwards, the MAE start monotonically increas-

ing, due to the observation that for higher values of the *min-occurs* threshold important features are also being filtered. Table 6 summarizes the most appropriate values of the *min-occurs* threshold for various feature categories:

Table 6. Values of *min-occurs* Threshold for Various Features Categories

category	**genres**	**keywords**	**actors**	**actresses**	**directors**	**countries**	**languages**
min-occurs (%)	*0*	*16*	*1.6*	*0.6*	*0.4*	*0*	*0*

Table 6 shows the minimal percentage of rated movies, where a certain feature should occur in order to be taken into account by the content-based prediction mechanism. For example, consider the *actors* feature category and a user who rated *500* movies. The *min-occurs* threshold of *1.6%* means that, if a certain actor participated in *1.6%* (or more) of movies rated by a user, i.e., in *8* movies (or more), this actor will be taken into account by the prediction mechanism, and otherwise, he will be ignored. Note that for the *genres* feature category, the *min-occurs* threshold is *0*. This means that all the available features from the *genres* category will be taken into account by the prediction mechanism.

After the most appropriate values of the *conf* and *min-occurs* thresholds were determined, we proceeded to the second task: selecting the feature categories that should be taken into account by the prediction mechanism. To do this we implemented the wrapper feature selection approach [Kohavi and John 1997] discussed above. The initial search state for the feature selection was $S_0=(0,0,0,0,0,0,0)$. That is, the algorithm started from an empty set of features, where none of the available *7* features categories (*genres*, *keywords*, *actors*, *actresses*, *directors*, *production countries* and *languages*) was taken into account by the prediction mechanism. Hence, the implemented feature selection used forward selection of features, i.e., the state expansions could only contribute new feature categories that should be taken into account by the prediction mechanism.

The accuracy of the recommendations for every state was computed using the above described *FT* set. The *90%* training set served as input for the collaborative filtering to content-based UMs conversion. Then, the generated content-based UM was used for generating pure content-based recommendations for the items from the *10%* test sets. Note that for each state of the search space, the content-based prediction mechanism took into account only those feature categories that were assigned *1* in the numeric vector representing the state, and ignored the categories that were as-

signed *0* in the vector[7]. The MAE metric [Herlocker et al. 2004] was used for evaluating the accuracy of the generated recommendations, i.e., for evaluating every state in the search space.

Execution of the wrapper feature selection yielded the following *5* feature categories that should be taken into account by the prediction mechanism: *genres*, *keywords*, *actors*, *actresses* and *directors*. This means that only the *production countries* and *languages* categories were excluded by the feature selection. To validate these results, we performed the accuracy experiment (shown later in Figure 11) in two configurations: (1) taking into account the features from all *7* feature categories, and (2) taking into account the features from *5* feature categories, excluding the *production countries* and *languages* categories. The results showed that the accuracy of the recommendations significantly improved as a result of the feature selection. These results will be extensively presented and discussed in Section 5.2.2.

The results of the wrapper feature selection experiment can be explained by an analysis of the distribution of the features in the excluded categories. In the *production countries* category, *1189* movies (i.e., *77.76%* of the movies) are produced in *USA*, whereas in the *languages* category, *1367* movies (i.e., *89.40%* of the movies) are in *English* language. Since the majority of movies have only one feature value (*USA* or *English*) within these feature categories, they are irrelevant for the vast majority of users, i.e., regardless of the feature values within these categories some movies are rated positively and some negatively. As a result, *production countries* and *languages* features do not affect users' ratings on the movies and these feature categories were filtered by the feature selection.

We hypothesize that, for certain (*non-USA* and/or *non-English*) values of these feature categories, running the feature selection would show that also these feature categories are important, and can potentially improve the accuracy of the recommendations. However, because the vast majority of movies that are stored in IMDb are produced in the *USA* and in *English*, this aspect could not be assessed. For example, for a user who likes *Italian* movies, taking the *production countries* category into account (and, in particular the feature *Italy*) would be extremely important. In fact, the issue of weighting specific features within the categories is beyond the scope of this work.

[7] Since the feature selection algorithm focused on selecting the most important feature categories, the values of both *min-occurs* and *conf* thresholds in this experiment were set to *0*.

5.2.2 Accuracy of the Recommendations

The selected *genres*, *keywords*, *actors*, *actresses* and *directors* feature categories were taken into account and the determined *conf=0.025* and *min-occurs* thresholds were applied in the second set of experiments. These experiments were aimed at comparing the accuracy of the recommendations generated using the original collaborative filtering UMs and of the recommendations generated using the converted content-based UMs.

For this experiment, the users in the dataset were partitioned into *12* groups of users, according to the number of rated movies: fewer than *25*, *26* to *50*, *51* to *75*, *76* to *100*, … , *201* to *225*, *226* to *250*, *251* to *300*, and *301* to *500* movies. Then, *325* users were selected from each group, and the collaborative filtering UM of each selected user was partitioned into *90%* of the ratings training set and *10%* of the ratings test set[8]. The ratings in the user's training sets were considered as their collaborative filtering UMs, and in parallel, served as input for the collaborative filtering to content-based UMs conversion mechanism.

Then, two types of recommendations for the items, which were rated in the test set, were generated: (1) pure collaborative filtering recommendations based on the original collaborative filtering UMs, and (2) pure content-based recommendations based on the converted content-based UMs. The accuracy of the generated recommendations was compared using the MAE metric [Herlocker et al. 2004]. Hence, for each group of users, the MAE values of the collaborative filtering and of the content-based recommendations for all the users in the group were computed. Also we would like to stress here that the training and testing stages of the experiment were conducted on disjoint sets.

To demonstrate the effect of the feature selection and validate our assumption regarding its importance for generating accurate recommendations, we computed the MAE of the content-based recommendations in two ways:

- Using the original *7* feature categories: *genres*, *keywords*, *actors*, *actresses*, *directors*, *production countries* and *languages*. The results of this experiment are denoted in the chart by CB.

[8] In the fine-tuning experiment, we selected the *FT* set of *1,000* users that rated over *100* movies. For the accuracy experiment, we defined *12* other groups of *325* users, i.e., overall *3,900* users. Although there is some overlapping between the new set and the *FT* set, it is partial and only for the groups of users that rated over *100* movies.

- Using the *5* feature categories selected by the feature selection: *genres*, *keywords*, *actors*, *actresses*, and *directors*. The results of this experiment are denoted in the chart by CBFS.

Figure 11 shows the MAE values. The horizontal axis reflects the number of users in a group, and the vertical axis stands for the MAE values.

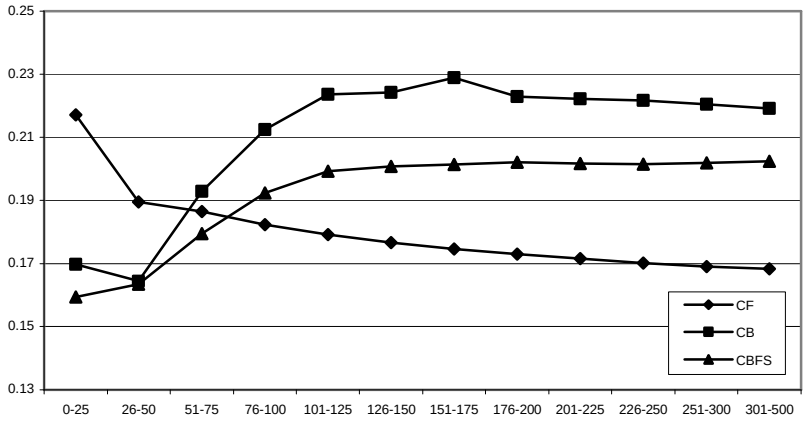


Fig. 11. MAE Values of Content-Based (with and without feature selection) and Collaborative Filtering Recommendations vs. Number of Rated Movies in the UMs

Comparison of the MAE values of the content-based recommendations using the original *7* feature categories and of the recommendations using the selected *5* categories shows that the latter recommendations (using the feature selection) steadily outperform the former (without the feature selection). The result is statistically significant, *p=1.26E-06*. This observation stresses the effect of the feature selection and practically proves its importance in the collaborative filtering to content-based UMs mediation. Hence, in the following analysis of the experimental results we will refer mainly to the content-based recommendations using the selected *5* feature categories.

The chart shows that the MAE of the content-based recommendations for the UMs containing fewer than *50* movies is relatively low: *0.159* for the selected *5* feature categories (and *0.169* for the original *7* categories). We hypothesize that this is explained by the observation that for a low number of rated movies in the collaborative filtering UMs, it is easy to determine the important content-based features and to compute their weights accurately, while the number of neutral features is still low, and they do not dominate in the recommendations generation. For larger collaborative filtering UMs, between *50* and *100* movies, the MAE values of the content-based recommendations increase linearly with the number of rated movies. We hypothesize that this happens due to a larger number of neutral features, influencing the predicted ratings computation and

hampering the accuracy of the generated recommendations. Finally, for the collaborative filtering UMs with over *100* rated movies, the MAE values stabilizes at approximately *0.20* for the selected *5* feature categories (and at *0.22* for the original *7* categories).

We would like to stress that for most of the groups the prediction rate is over *0.99* (except the group of less than *25* movies, where it is *0.97*). High prediction rate values hold for both content-based recommendations using the original *7* feature categories and for the recommendations using the selected *5* categories. This means that recommendation values can be successfully computed for almost every movie.

Comparison of the MAE values of the content-based and collaborative recommendations shows that for the groups of users that rated fewer than *75* movies (or, fewer than *50* movies for the experiment without the feature selection) in the collaborative filtering UMs, content-based recommendation based on the converted artificial UMs outperforms collaborative recommendations based on the original UMs. According to the Table 3, *78.4%* of the users in the dataset rated fewer than *75* movies (and, for the results of the experiment without the feature selection, *64.83%* of the users rated fewer than *50* movies). Thus, improving the accuracy of the recommendations for these groups of users is extremely important. Due to the new user problem, the accuracy of the collaborative filtering recommendations for this size of UMs is relatively low. Hence, mediation from the collaborative filtering UMs to the content-based UMs and further generation of pure content-based recommendations provide a solid alternative technique.

For users who rated a larger number of movies in the collaborative filtering UMs, the accuracy of the collaborative filtering recommendations outperforms the accuracy of the content-based recommendations. However, the difference in the MAE is relatively small. Hence, we hypothesize that applying more accurate weighting mechanisms and discovering the dependencies between specific features may improve the accuracy of the content-based recommendations also for larger UMs.

Finally, we would like to compare the results of the last experiment with the results of a similar experiment reported in [Basu et al. 1998]. That work aimed at combining the collaborative filtering and content-based UMs and recommendation approaches. To do this, (1) content-based in-

formation about the items rated in the collaborative filtering UM was extracted, (2) this information, jointly with the original collaborative ratings, was treated as the features of the items, and (3) content-based and hybrid recommendations were generated using these enriched UMs. In the study, the authors also focused on the application domain of movies and extracted content-based information from *26* feature categories from the IMDb. Experimental evaluation measured the accuracy of the generated recommendations (in terms of precision and recall [Salton and McGill 1983]). The results showed that the accuracy of the content-based recommendations is inferior that of the original collaborative filtering recommendations, and only the hybrid recommendations allow the accuracy of the recommendations to be improved). Conversely, in this section we showed that the accuracy (in terms of MAE) of the recommendations generated using the converted content-based UMs outperforms that of the recommendations generated using the original collaborative filtering UMs. We hypothesize that this improvement was achieved due to the fine tuning of the prediction mechanism using the *conf* and *min-occurs* threshold and feature selection that were applied.

5.3 Summary

This section focused on cross-representation mediation of UMs and demonstrates practical implementation and evaluation of the mediation from the collaborative filtering to the content-based UMs. This mediation procedure allows bootstrapping of the UMs of a content-based recommender system, and facilitates generation of accurate recommendations for a new user, whose UM was imported from the collaborative filtering recommender system.

The experimental evaluation initially focused on the fine-tuning of the prediction mechanism. The experiments showed that for a small number of rated movies in the collaborative filtering UMs (typical for the majority of the users), the accuracy of the content-based recommendations is higher than that of the collaborative filtering ones. Hence, cross-technique mediation of the UMs facilitates improvement of the accuracy of the personalization services provided to the users. Also, the experiments allowed the contribution of the feature selection to be evaluated by comparing the accuracy of the recommendations with and without the preceding feature selection process. The results validated the assumptions regarding the importance of the feature selection, as applying it allowed the accuracy of the generated recommendations to be improved even more.

Chapter 6: Cross-Domain Mediation of User Modeling Data

The collaborative filtering recommendation technique assumes that people with similar tastes, i.e., people who agreed in the past, will also agree in the future. Hence, the collaborative filtering recommendation generation process is typically decomposed into three general stages: (1) similarity computation: weighting all the users with respect to their similarity with the active user; (2) neighborhood formation: selecting *K* most similar users, i.e., nearest-neighbors for the recommendation generation; and (3) recommendation generation: computing the recommendation by weighting the ratings of the users in the neighborhood on the target item [Herlocker et al. 1999].

Collaborative filtering systems suffer from *new item* and *new user* bootstrapping problems. The new item problem refers to the fact that if the number of users who rated an item is small, accurate recommendations for this item cannot be generated. The new user problem refers to the fact that if the number of items rated by a user is small, it is unlikely that there is an overlap of products rated by this user and other users. Hence, users' similarity cannot be reliably computed and accurate recommendations for the user cannot be generated. These problems are referred to as particular cases of a collaborative filtering *sparsity problem* [Linden et al. 2003], where the contents of the ratings matrix are insufficient for generating accurate recommendations.

This work focuses on overcoming this problem through cross-domain mediation of UMs. In this mediation, the user modeling data are imported from remote systems exploiting the same collaborative filtering recommendation technique as the target system, in other, relatively similar, application domains. Hence, both target and remote systems represent the UMs as a list of ratings provided by a user on the domain items. In this setting, four types of user modeling data can be imported: (1) UMs stored by the remote system, (2) lists of the neighborhood candidates, (3) degrees of similarity between the active user and the other users, computed over the data stored by the remote system, and (4) complete recommendations generated by the remote system. This work elaborates on these four types of cross-domain mediation in collaborative filtering and presents their implementation and experimental evaluation using the widely-used EachMovie dataset [McJones 1997].

The section is organized as follows: Section 6.1 discusses cross-domain mediation approaches; Section 6.2 presents the results of the experimental evaluations; and Section 6.3 summarizes the section.

6.1 Cross-Domain Mediation in Collaborative Filtering

Traditional collaborative filtering recommender systems store their user modeling data, i.e., the ratings, in a two-dimensional matrix M:$(user_{id}, item_{id}) \rightarrow rating$, where $user_{id}$ and $item_{id}$ represent the unique identifiers of users and items and *rating* represents the explicit evaluation given by a user $user_{id}$ on an item $item_{id}$. Note that the number of items managed by the system is typically significantly larger than the number of ratings provided by an average user. This leads to a very sparse ratings matrix M and to the sparsity problem of collaborative filtering.

Conversely, in a domain-distributed setting, the ratings matrix M is stored in a semi-centralized way. In this case, every domain d stores a local ratings matrix M_d. The structure of M_d is similar to the structure of M, i.e., it is a two-dimensional matrix representing the ratings given by a set of users on a set of items. However, this set of items in the matrix is restricted to the items that belong to a certain application domain d, i.e., M_d: $(user_{id}, item_{id}) \rightarrow rating$. Hence, this setting can be considered as a vertical partitioning of the ratings matrix M (Figure 12).

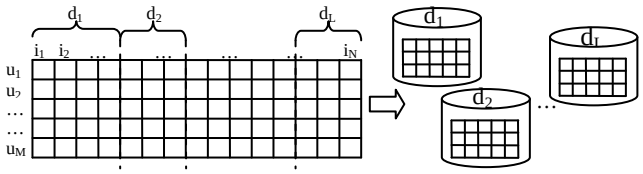


Fig. 12. Domain-Related Vertical Partitioning of the Ratings Matrix

Note that this is not exactly vertical partitioning of the ratings matrix. In a real vertical partitioning, the partitioned sets of items are disjoint, i.e., every item belongs to a single set. In domain-related vertical partitioning, certain items may belong to multiple domains or categories. This setting is not uncommon if the above representation of domains is downscaled to the representation of E-Commerce services. In this case, ambiguous categorization of items may be explained by different classifications of products, their providers, or E-Commerce sites.

Over the above partitioning of the ratings matrix, various types of user modeling data can be mediated and imported. These types of data are those exploited in the three stages of collaborative filtering recommendations generation process: similarity computation, neighborhood formation, and recommendation computation. For the similarity computation, the UMs are imported from the remote recommender systems. For the neighborhood formation, either the list of candidates for being the nearest-neighbors or users' similarities computed by the remote systems are imported. Finally, for the recommendation computation, complete recommendations for items, generated by the remote systems, are imported.

6.1.1 Importing User Modeling Data in Collaborative Filtering

A typical personalization scenario is initiated by a recommendation request issued by a user $user_{id}$ to a collaborative filtering recommender system R_t in the target application domain t. As a result, the target system R_t selects a set of items that can be potentially recommended $\{item_{id}\}$ and initiates a recommendation generation process for every $item_{id}$. To enhance the accuracy of the recommendations, R_t queries a set of available remote collaborative filtering recommender systems $\{R_d\}_{d \in D}$ from other closely-related domains D for the related user modeling data (the relations between target domain t and domains in D will be discussed later). The query is formulated as a triplet $q=<user_{id}, item_{id}, t>$. In the following discussion, let us assume that the identities of the users and items are unique in all the domains and systems.

According to the first mediation approach, the UMs themselves (i.e., the rating vectors) stored by a remote system R_d operating in another domain d, are imported. For the sake of simplicity, let us assume that R_d responds to q by sending to R_t the content of the local repository of UMs, i.e., $resp_d=M_d$, where M_d is local ratings matrix containing only the items that belong to domain d. Upon receiving the set of responses $\{resp_d\}$ and given a policy for resolving conflicts in the ratings of the same user-item pair coming from different systems, R_t constructs the unifying ratings matrix M by integrating local and imported data. Over M, traditional collaborative filtering mechanism of similarity computation, K nearest-neighbors selection and recommendations generation is applied. Since the constructed matrix M can be considered as the standard centralized collaborative filtering matrix, this approach is referred to as *Standard* collaborative filtering and it serves as a baseline for the experimental comparisons.

The second and third mediation approaches deal with importing nearest-neighbors data computed locally by the remote systems R_d. Two approaches are considered:

- *Heuristic* – relies on a heuristic assumption that similarity of users spans across multiple application domains. Hence, if two users are similar in a certain remote application domain *d*, they may be also similar in the target domain *t*. Practically, this means that R_d responds to *q* by sending to R_t the set of *K* identities of the users most similar to the active user, i.e., $resp_d=\{user_{id}\}$. Upon receiving the set of responses $\{resp_d\}$, R_t aggregates these sets of nearest-neighbors into the overall set of heuristic candidates for being the nearest-neighbors, computes their true similarity values according to the local ratings matrix M_t, selects the set of *K* nearest-neighbors, and generates the recommendations.
- *Cross-domain* – computes the overall similarity between two users as an aggregation of their domain-related similarity values. Upon receiving the request *q*, every remote system R_d computes locally, i.e., according to the contents of the local ratings matrix M_d, the similarity between the active user and the other users in M_d. A set of *K* nearest-neighbors is selected, and their $user_{id}$ jointly with their similarity values are sent to R_t. In other words, $resp_d=\{(user_{id},sim_d)\}$, where $sim_d=sim(user_{id},user_{act})$ is the local similarity between a user $user_{id}$ and the active user $user_{act}$, computed according to their ratings in the application domain *d* using a certain similarity metric *sim*. Upon receiving the set of responses $\{resp_d\}$, R_t aggregates the domain-related similarity values into the overall similarity metric using inter-domain correlation values. The overall similarity is computed by:

$$sim(i,l)=\frac{\sum_{d\in D} cor(d,t)sim_d(user_{id},user_{act})}{\sum_{d\in D} cor(d,t)}$$

  where $sim_d(user_{id},user_{act})$ is the local similarity value in application domain *d* and $cor(d,t)$ is the correlation of the target domain *t* and remote domain *d*. As the overall similarity value is computed (various ways for computing inter-domain correlations are described in the following subsection), *K* nearest-neighbors are selected and the recommendations are generated.

The fourth mediation approach deals with complete collaborative filtering recommendations generated locally by the systems from remote domains, referred to as *Local* collaborative filtering. According to it, the recommendations are generated using only the data stored in the ratings matrix M_d of the collaborative filtering system from a certain application domain *d*. This is done similarly to the centralized collaborative filtering, but using a restricted set of ratings on items

from *d*: local similarity values are computed, the set of *K* nearest-neighbors is selected and the recommendations are generated. However, *Local* collaborative filtering disregards the fact that the items typically belong to multiple application domains and treats each domain independently. Hence, according to *Remote-Average* variant of *Local* collaborative filtering, every remote system R_d from another application domain *d*, to which the predicted item belongs, generates a separate local recommendation using the ratings stored in its ratings matrix M_d. The computed recommendations are sent to R_t, i.e., $resp_d=pred_d$. Upon receiving the set of responses $\{resp_d\}$ and generating a local recommendation using its own matrix M_t, R_t aggregates all the recommendations into a single value by averaging the set of received recommendations with the locally generated recommendation. In the experimental evaluation, *Remote-Average* is compared with the *Local* collaborative filtering, applied over the ratings in the target domain *t* only, i.e., with the recommendations generated using the data stored only in its target domain *t* ratings matrix M_t.

6.1.2 Computation of Inter-Domain Correlations

To aggregate multiple domain-related similarities of users into the overall similarity value, there is a need for an inter-domain correlation metric. In other words, this requires a metric that will compute correlation, $cor(d_1,d_2)$ between two application domains d_1 and d_2. This subsection discusses two alternative correlation computation techniques: *content-based* and *ratings-based*. Assuming stable contents of the domains and stable ratings on the domain items, the inter-domains correlation computation can be considered as a one-time pre-processing process.

The content-based correlation computation technique assumes that the textual contents of a domain can be considered as reliable representatives of the domain topics [Berkovsky et al. 2006g]. Hence, the similarity of two application domains d_1 and d_2 is computed as a three-stage process: (1) mining the textual content of the domains, eventually obtained from external data sources, such as the Web or other specialized databases; (2) representing the mined textual contents as feature vectors v_1 and v_2, where $v_i=(w_{i1}, \ldots, w_{in})$ and w_{ij} is the tf-idf weight [Salton and McGill 1983] of the term *j* appearing in the domain *i;* and (3) computing inter-domain correlation as the cosine similarity of their respective feature vectors:

$$cor_{contents}(d_1,d_2) = sim(v_1,v_2) = \frac{v_1 \cdot v_2}{\| v_1 \|_2 \times \| v_2 \|_2}$$

where · denotes the inner product between the feature vectors, and $||v_i||_2$ denotes their norm, i.e., the square root of the scalar product of a vector with itself. The result of this computation is a scalar, measuring the correlation of two domains, computed based on their textual contents.

Alternatively, the ratings-based correlation computation is based on the similarity of ratings on the domain items [Bridge and Kelly 2006]. Given two items, *item*$_1$ and *item*$_2$, their ratings-based similarity *sim(item*$_1$*,item*$_2$*)* can be computed as the correlation, e.g., cosine similarity, of their respective ratings vectors. Using item-based similarity metric, inter-domain correlation is computed as the average similarity of all the possible pairs of different items that belong to these application domains:

$$cor_{ratings}(d_1, d_2) = AVG\{sim(item_i, item_j) : i \neq j,\ \ i \in J_{d_1},\ \ j \in J_{d_2}\}$$

where *sim(item*$_i$*,item*$_j$*)* is the ratings-based similarity of two items. Also the result of this computation is a scalar, measuring the correlation of two domains, but in this case computed based on the similarity of ratings on the domains' items.

6.2 Experimental Evaluation

One of the main difficulties when conducting an experimental evaluation of cross-domain mediation and collaborative recommendations using multiple user modeling data is the lack of publicly available data, representing the ratings of the same users on items from multiple application domains. Although there exist several datasets from various domains (e.g., movies, books, jokes, browsing logs), none of them is cross-linked, i.e., they do not allow their users in other datasets to be identified. Hence, experimental evaluation of the proposed mediation approaches involved EachMovie dataset of movie ratings [McJones 1997], where the items were vertically partitioned, rather than where the ratings from different application domains were collected.

To mimic domain-related vertical partitioning of the ratings matrix, the movies were partitioned according to their genres. Eight genre-related ratings matrices were created: *action*, *animation*, *comedy*, *drama*, *family*, *horror*, *romance*, and *thriller*. In EachMovie, the movies usually belong to multiple (up to *4*) genres such that each movie belongs, on average, to *2.376* genres. Hence, the sets of movies in the genre-related matrices were not disjoint. Table 7 summarizes the distribution

of movies and ratings among genre-related ratings matrices and sparsity of each matrix. The sign *K* in the number of ratings row denotes one thousand ratings.

Table 7. Data Distribution among Genre-Related Matrices.

	action	**animation**	**comedy**	**drama**	**family**	**horror**	**romance**	**thriller**
Num. movies	*198*	*43*	*400*	*536*	*145*	*87*	*137*	*177*
Num. ratings	*1,166K*	*193K*	*2,209K*	*3,056K*	*800K*	*433K*	*681K*	*991K*
sparsity (%)	*91.923*	*93.852*	*92.425*	*92.180*	*92.432*	*93.181*	*93.179*	*92.321*

6.2.1 Inter-Domain Correlations

To compute inter-domain correlations, both content-based and ratings-based techniques were applied. Content-based technique exploited the lists of movie keywords mined from the IBDb movie database [IMDb 2007] to generate genre-related tf-idf feature vectors and compute inter-genre correlations [Salton and McGill 1983]. The ratings-based technique exploited the ratings on the movies in EachMovie. Tables 8 and 9 show the matrices of inter-genre correlation. Table 8 shows the content-based and Table 9 the ratings-based technique.

Table 8. Inter-Genre Correlations – Content-Based Computation

	action	**animation**	**comedy**	**drama**	**family**	**horror**	**romance**	**thriller**
action	*1.000*	*0.860*	*0.935*	*0.932*	*0.820*	*0.902*	*0.913*	*0.943*
animation	*0.860*	*1.000*	*0.913*	*0.848*	*0.914*	*0.765*	*0.838*	*0.787*
comedy	*0.935*	*0.913*	*1.000*	*0.965*	*0.905*	*0.868*	*0.957*	*0.903*
drama	*0.932*	*0.848*	*0.965*	*1.000*	*0.841*	*0.873*	*0.987*	*0.938*
family	*0.820*	*0.914*	*0.905*	*0.841*	*1.000*	*0.739*	*0.832*	*0.772*
horror	*0.902*	*0.765*	*0.868*	*0.873*	*0.739*	*1.000*	*0.850*	*0.939*
romance	*0.913*	*0.838*	*0.957*	*0.987*	*0.832*	*0.850*	*1.000*	*0.913*
thriller	*0.943*	*0.787*	*0.903*	*0.938*	*0.772*	*0.939*	*0.913*	*1.000*

Table 9. Inter-Genre Correlations – Ratings-Based Computation

	action	**animation**	**comedy**	**drama**	**family**	**horror**	**romance**	**thriller**
action	*0.129*	*0.095*	*0.078*	*0.067*	*0.086*	*0.093*	*0.075*	*0.109*
animation	*0.095*	*0.167*	*0.074*	*0.059*	*0.125*	*0.077*	*0.074*	*0.082*
comedy	*0.078*	*0.074*	*0.072*	*0.058*	*0.071*	*0.065*	*0.070*	*0.074*
drama	*0.067*	*0.059*	*0.058*	*0.063*	*0.056*	*0.060*	*0.065*	*0.069*
family	*0.086*	*0.125*	*0.071*	*0.056*	*0.119*	*0.067*	*0.072*	*0.076*
horror	*0.093*	*0.077*	*0.065*	*0.060*	*0.067*	*0.149*	*0.060*	*0.098*
romance	*0.075*	*0.074*	*0.070*	*0.065*	*0.072*	*0.060*	*0.091*	*0.074*
thriller	*0.109*	*0.082*	*0.074*	*0.069*	*0.076*	*0.098*	*0.074*	*0.109*

Both techniques yielded symmetric matrices, i.e., $cor(d_1,d_2)=cor(d_2,d_1)$. The diagonal values in content-based matrix are *1* – the correlation between a feature vector and itself. Also other inter-genre correlations are relatively high, above *0.73*. Conversely, in ratings-based matrix, the diagonal values are lower, since they are computed using the ratings vectors of the movies, which are typically different even within the same genre. Nevertheless, in many genres the diagonal values are still higher than other correlation in the respective column or row.

6.2.2 Cross-Domain Experiments with Collaborative Filtering Data

Four collaborative filtering data mediation approaches discussed in the previous section were implemented and evaluated. Cosine similarity was selected as the users' similarity metric [Salton and McGill 1983]. The minimal number of movies rated by users for the similarity computation was *6* (recommendations could not be generated for users who rated fewer than *6* movies). The number of nearest-neighbors returned by remote domains to the target domain in *Heuristic* and *Cross-domain* approaches was *20*. The number of nearest-neighbors used for the recommendation generation was *20*.

The following experiments evaluated the effect of sparsity of the target user ratings in the matrix of the target domain on the accuracy of the recommendations. Hence, the users in the genre repositories were partitioned into *12* categories, according to the percentage of the rated movies from the target genre: under *3%*, *3%* to *6%*, …, *30%* to *33%*, and over *33%*. For every group, *1,000* recommendations were generated for various combinations of user, movie, and target genre. The recommendations were generated using the following collaborative filtering approaches: *Standard*, *Heuristic*, *3* variants of *Cross-Domain*, *Local*, and *Remote-Average*. The recommendations' accuracy was measured using the MAE metric [Herlocker et al. 1999]:

$$MAE = \frac{\sum_{i=1}^{N} | p_i - r_i |}{N}$$

where N denotes the total number of the recommendations, p_i is the predicted rating, and r_i is the real rating on the movie in recommendation number i. In the following figures, the horizontal axis shows the percentage of rated movies in the target genre and the vertical axis the MAE. The baseline for all the comparisons is *Standard* approach, as its results are similar to the results that would have been obtained in traditional centralized collaborative filtering.

The results of *Local*, *Remote-Average* and *Standard* approaches (Figure 13) show that both Local and Remote-Average CF outperform *Standard* approach for any percentage of rated movies (statistically significant, *p=2.78E-07* and *p=1.63E-06*, respectively). This can be explained by arguing that the similarity computation over the ratings from the target genre only in *Local* approach (or over the ratings from other movie genres in *Remote-Average*) yields more accurate similarity values than the similarity computation over all the available ratings. This can be explained by the observation that the ratings from these genres are important for computing the similarity value in

the relevant genre, whereas the other ratings may introduce noise into the computation. As a result, the recommendations are more accurate.

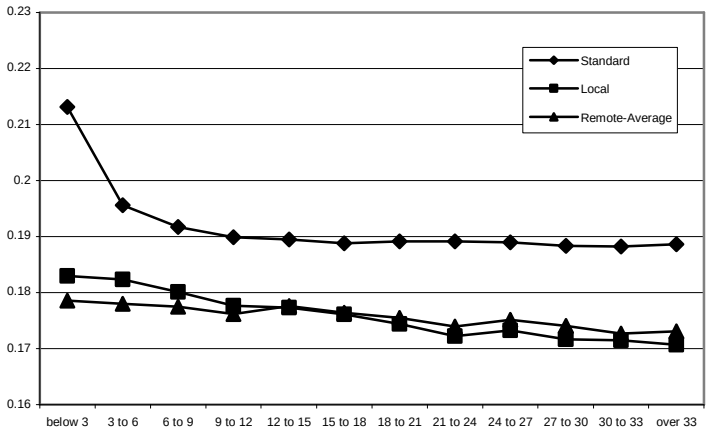


Fig. 13. Local, Remote-Average and Standard Approaches

A comparison of the *Local* and *Remote-Average* approaches shows that for a small percentage of rated movies, i.e., sparse ratings matrix, *Remote-Average* approach is slightly more accurate (statistically insignificant). This can be explained by the fact that the recommendations are generated using additional knowledge acquired by importing data from other relevant genres and not using the data from the target genre only. For a higher percentage of rated movies, the local data are sufficient and the imported data hamper the accuracy of the recommendations.

It should be stressed that in certain conditions *Local* and *Remote-Average* approaches are inapplicable. For example, for a group of under *3%* of rated movies, recommendations can be generated only for comedies and dramas, as only in these genres is *3%* of the number of movies above *6* movies, a minimal number of movies for the similarity computation. Hence, although the accuracy of *Local* and *Remote-Average* approaches is higher, they are not capable of generating recommendations for certain movies, which will negatively affect the ability of the system to recommend all the interesting movies.

The results of *Heuristic* and *Standard* approaches (Figure 14) show that for a small percentage of rated movies, *Standard* approach is more accurate. This can be explained by the fact that the set of nearest-neighbors candidates computed by the *Heuristic* approach is not accurate in comparison with the real set of nearest-neighbors. However, the accuracy of the candidates set increases with the percentage of rated movies and *Heuristic* approach outperforms *Standard* approach starting from *9%* to *12%* (statistically significant, $p=0.03896$). It should be stressed that the accuracy

of *Heuristic* approach is inferior to the accuracy of the *Local* approach: their recommendation generation is identical (uses target genre ratings only), but while the set of *K* nearest-neighbors in *Heuristic* approach is found by an approximated search, in *Local* approach it is found by an exhaustive search of all the users.

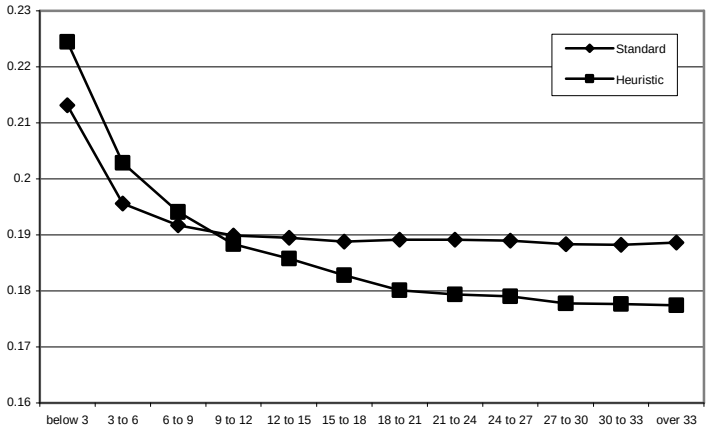


Fig. 14. Heuristic and Standard Approaches

Figure 15 shows the results of *Cross-Domain* (*Cross-Genre*) and *Standard* approaches. Three particular instances of *Cross-Genre* approach were evaluated: (1) *Cross-Genre* with content-based inter-genre correlations, (2) *Cross-Genre* with ratings-based inter-genre correlations, and (3) *Cross-Genre* with uniform inter-genre correlations, where all the correlations are set to *1*. The latter approach was aimed at evaluating the contribution of other *Cross-Genre* approaches, and served as a baseline for experimental comparisons of *Cross-genre* approaches.

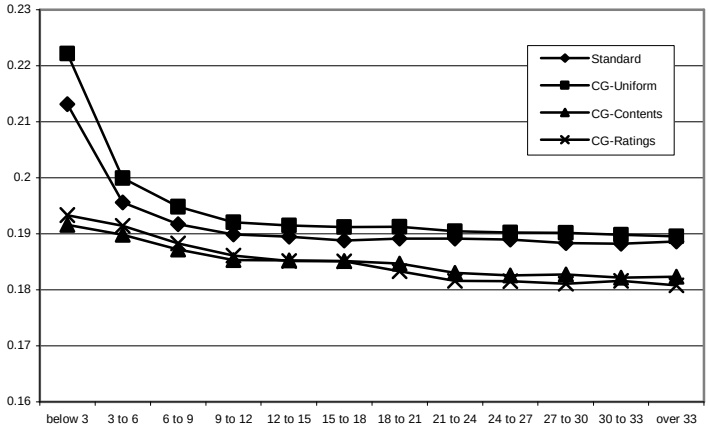


Fig. 15. Cross-Genre Uniform, Content-Based, Ratings-Based and Standard Approaches

The results show that both content- and ratings-based *Cross-Genre* approaches outperform the *Standard* approach (statistically significant, $p=0.00058$ and $p=0.00024$, respectively). This can be explained by the observation that the weighted similarity metric aggregating the set of domain-

related similarities according to inter-genre correlations is more accurate than *Standard* similarity metric assigning equal weights to all the ratings. It should be stressed that the accuracy of uniform *Cross-Genre* approach is lower than the accuracy of content-based and ratings-based *Cross-Genre* approach (and than the accuracy of *Standard* approach). This shows that *Cross-Genre* similarity computation is beneficial and improves the accuracy of the recommendations. Note that, although content-based and ratings-based inter-genre correlation matrices were different, the accuracies of both approaches are very similar. This proves the validity of ratings-based similarity computation [Bridge and Kelly 2006].

Comparing content- and ratings-based *Cross-Genre* and *Local* approaches shows that the latter is more accurate for any percentage of rated movies (statistically significant, *p=4.13E-10* and *p=5.66E-13*, respectively). However, as discussed earlier, *Local* approach may be inapplicable for a low percentage of rated movies due to the sparsity of ratings in the target domain ratings matrix. In this case, *Cross-Genre* approach should be applied, as its accuracy outperforms the accuracy of *Standard* approach.

6.3 Summary

This section focused on cross-domain mediation of collaborative filtering user modeling data for the purposes of resolving the bootstrapping problem, typical to collaborative filtering systems. In particular, it implemented and experimentally evaluated the effect of importing the following four types of user modeling data: (1) complete UMs, (2) lists of the nearest-neighbors candidates, (3) degrees of users' similarity computed over the local data, and (4) complete recommendations.

Experimental evaluation, conducted in the domain of movies, showed that generating recommendations over multiple domain-related user modeling data can yield a higher accuracy of the recommendations than the accuracy of the recommendations built by merging the ratings in a centralized repository and then applying the traditional collaborative filtering technique. However, this approach is not applicable for very sparse data, where aggregating local similarity degrees is more appropriate. Two weighted aggregation methods were evaluated: content-based and ratings-based. Both improved the accuracy of the recommendations, compared to the traditional collaborative filtering method assigning equal weights to all the degrees of inter-genre correlation.

Part 4:

Practical Aspects of User Modeling Data Mediation

Chapter 7: Distributed Storage and Retrieval of User Modeling Data

Peer-to-Peer (P2P) computing refers to a subclass of distributed computing, where system's functionality is achieved in a fully decentralized way by using a set of distributed resources [Androutsellis-Theotokis and Spinellis 2004]. P2P systems usually lack a central point of management, depending rather on a voluntary contribution of resources, e.g., computing power, data, and network traffic, by the connected peers. As a result, P2P systems provide pure distributed communication middleware with theoretically unlimited storage, communication, and processing capabilities [Milojicic et al. 2002].

In this section we propose using the storage resources of P2P networks as a user modeling data storage component. Such component can be applied for the purpose of distributing the storage of UMs and supporting distributed retrieval of similar UMs in similarity-based recommendation approaches, such as collaborative [Herlocker et al. 1999] and demographic [Krulwich 1997] filtering. Since P2P systems do not imply central management, the user modeling data are inserted autonomously by multiple systems and may be described by a dynamic set of terms, such that the data storage component should support heterogeneity in the descriptions of user modeling data. Thus, efficient management of the user modeling data requires a stable semantic-based infrastructure allowing identification of similarities and commonalities between the heterogeneously described data. Moreover, this infrastructure should support a retrieval process that: i) performs decentralized retrieval with low communicational overhead, and ii) guarantees fast discovery of the similar UMs, or a reasonably accurate approximation.

We used the hypercube-based approach of UNSO (UNSpecified Ontology) [Ben-Asher and Berkovsky 2006] for implementing P2P storage of user modeling data and developed an approximated retrieval algorithm designed to reduce the computational and communicational overheads, in comparison with the traditional exhaustive retrieval. The algorithm is based on the observation that UNSO implicitly groups similar UMs, such that they are located in relatively close proximity with respect to the underlying network topology. Hence, an approximated retrieval of UMs in UNSO is performed through a localized expanding search starting at the location of the target UM in the underlying network, i.e., at the location of the UM of the user who requested a

recommendation from the system. This means that the most similar UMs are searched at the locations that are close to the location of the target UM and the number of UM comparisons is reduced with respect to the traditional retrieval, which exhaustively compares the target UM with all the available UMs.

Due to a lack of publicly available heterogeneous user modeling data, the above approach was evaluated in five datasets of E-Commerce data objects from five application domains, which represent ephemeral search models of the users. We compared the proposed approximated retrieval with the traditional exhaustive retrieval algorithm. We showed that the former significantly reduces the number of required comparisons, while preserving the precision and recall of the search [Salton and McGill 1983]. Hence, the contributions of this work are two-fold. First, we propose a novel notion of pure decentralized storage of user modeling data in a P2P environment. Second, we develop and evaluate an efficient and accurate approximated algorithm for retrieval of the most similar UMs over the above decentralized storage of user modeling data.

The rest of this section is structured as follows. Section 7.1 presents semantic data management in P2P systems. Section 7.2 describes the *unspecified* P2P data management using UNSO and the representation of user modeling data using UNSO. Section 7.3 defines the distance metric used in the evaluation and presents the approximated retrieval algorithm. Section 7.4 presents and analyzes the results of the experimental evaluation, and Section 7.5 summarizes this section.

7.1 Semantic Data Management in Peer-to-Peer Systems

The first P2P systems were designed for large-scale data sharing. Applications, such as Napster [Napster 2007], Gnutella [Adar and Huberman 2000], and Freenet [Clarke et al. 2000], allowed users to download multimedia data, provided by other users. Performance of these systems suffered from severe scalability problems. In Napster [Napster 2007], a cluster of central servers maintained the indices of the files provided by the users. The flooding search algorithm of Gnutella did not allow communication-efficient retrieval of data objects over a heterogeneous set of users [Adar and Huberman 2000]. Freenet, despite being fully decentralized and employing efficient routing algorithms, could not guarantee reliable and unambiguous data management [Clarke et al. 2000].

This has led to the development of content-addressable P2P systems, such as CAN [Ratnasamy et al. 2001] and Pastry [Rowstron and Druschel 2001]. These systems implement a scalable self-organizing infrastructure for fault-tolerant routing using Distributed Hashing Tables (DHT) [Harren et al. 2002]. In DHT-based systems, each user and data object is assigned a unique identifier from a sparse space, called respectively *user* and *key*. Note that every *key* uniquely represents the characteristics and contents of the respective data object, whereas its exact format is defined by the system. Since a user may provide multiple data objects, the users are typically represented by the respective sets of the provided data objects: $user_i=\{key_{i1}, key_{i2}, \ldots, key_{im}\}$.

The data objects are inserted into the system through mapping their keys to the logical nodes of the underlying communication middleware, i.e., P2P network. This is achieved using a globally-known hashing function $put(user_i, key_{ij})$, which assigns $user_i$ as a provider of the data object identified by key_{ij}. This assignment is done through coupling the key of the data object key_{ij} and the middleware node $node_k$ obtained by applying the *put* function on key_{ij}. Since in pure decentralized P2P systems the data objects are stored by the users, every user virtually manages a set of middleware nodes, which reflects the set of data objects provided by the user. Thus, in the rest of the section the term *node* refers both to the communication network node and to the user managing this node.

The fact that the *put* functions used to map the data objects to the middleware nodes are globally-known facilitates further discovery of a data object identified by key_{ij} (i.e., the user providing it) by other users. This is done through $get(key_{ij})$ function, which exploits the same hashing function as *put* and returns $user_i$, the identifier of the user providing the data object identified by key_{ij}. Thus, DHT-based P2P systems facilitate decentralized management and search of data objects, where the unique description of the data object key_{ij} and hashing functions *put* and *get* guarantee proper functionality of the system.

For example, consider two users *X* and *Y* providing seven data objects in a CAN-based system [Ratnasamy et al. 2001] exploiting two-dimensional CAN middleware (shown in Figure 16). Three data objects are provided by user *X* and four are provided by user *Y*. The keys of the above data objects are mapped to the nodes of the middleware using the *put* function. Let us assume that

the keys of the data objects provided by user *X* are mapped to the nodes *b*, *c*, and *g*, whereas the keys of the data objects provided by user *Y* to the nodes *a*, *d*, *e* and *f* (shown in Figure 16 in different shades). When other users search for these data objects, the same hashing mechanism exploited by *get* function points them to the relevant nodes of the middleware and to the users providing the requested data objects.

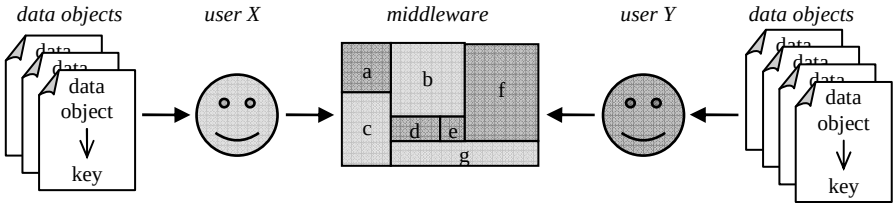


Fig. 16. Management of Data Objects in a DHT-Based P2P Middleware

In comparison to prior P2P systems, DHT-based systems are highly scalable, and provide a robust, self-organizable, and completely decentralized structure. Due to an effective routing algorithm [Plaxton et al. 1997], which routes the messages only to the relevant users instead of the expensive network flooding, their overall traffic is significantly lower [Rowstron and Druschel 2001]. However, DHT-based systems basically rely on hashing-based *put()* and *get()* operations. This results in two major limitations:

- Support for exact-matching lookups only. The keys of two similar, but not identical data objects key_1 and key_2 are treated as two diverse keys. Hence, only the searches specifying the exact *key* used while inserting a data object will succeed in locating it. This limitation hampers the data management characteristics of DHT-based systems and leads to an uncontrolled redundancy in the data objects repository, as similar, but not exactly matching data objects cannot be identified by the search queries.
- Support for single-key lookups only. The above *put()* and *get()* operations handle a single *key* only, i.e., a data object is described by a single string. Although the *key* might be represented as a concatenation of the substrings representing parts of the *key*, any change in one of them will prevent identification of the matching, as the hashing mechanism will map it to a different middleware node.

This has lead to the development of a more complex kind of P2P network, built upon peers using their own schemata to describe the objects. This approach is further referred to as semantic or ontology-based data management. Ontology is defined as a *formal shared conceptualization of a particular domain* [Gruber 1993]. It acts as a standardized reference model, providing both hu-

man-understandable and machine-processable semantic mechanisms, allowing various enterprises and systems to collaborate efficiently. Two techniques for ontology-based data management in P2P networks are discussed in HyperCup [Schlosser et al. 2002] in UNSO [Ben-Asher and Berkovsky 2006].

HyperCup [Schlosser et al. 2002] proposes a flexible ontology-based pure decentralized P2P middleware, generating a hypercube-like graph of data objects represented by the users providing them. This means that on the logical level the P2P hypercube consists of the data objects, whereas on the physical level it consists of the users providing these objects, connected by the underlying P2P network organized as a hypercube. To store the data objects in a distributed manner, HyperCup exploits a predefined ontology of the data objects domain, such that the dimensions of the hypercube match the concepts of the ontology, i.e., the features characterizing the data objects. As a result, every data object described by the ontology concepts is mapped to one of the hypercube nodes. This mapping assigns the user providing a data object to manage the respective middleware node. The data objects can be discovered using the same domain ontology.

For example, consider the following simple ontology-based representation of user modeling data objects in a demographic recommender system. The users are represented using *3* ontological features {*gender*, *age*, *residence*} that are mapped to the dimensions of the hypercube, e.g., *gender* is mapped to dimension number *1*, *age* is mapped to dimension *2*, and *residence* to dimension *3*. The values of the features, as expressed by the specific UMs, are mapped to the numeric coordinates within the dimensions. For example, a value *female* of the dimension *gender* is mapped to coordinate *0*, whereas a value *male* is mapped to *1*. Similarly, the values of *age* are partitioned to age groups of fewer than *10* years old, *10* to *20* years old, *20* to *30* years old, and so forth, and the values of *residence* are enumerated according to the alphabetical list of countries or states. Note that such predefined ontology-based mapping mechanism inherently imposes the order of the values for the features and facilitates the defining of accurate similarity metrics for similarity-based searches and ranking of the retrieved data objects.

To maintain connectivity of the underlying communication middleware, hypercube nodes are connected, such that every user managing a hypercube node stores a data structure of immediate logical neighbors in different dimensions. For example, consider a UM, which is mapped to a

node (*x,y,z*) of the underlying *3*-dimensional hypercube. As a result, the node representing this UM is connected to *6* logical neighbors, i.e., nodes (*x+1,y,z*), (*x-1,y, z*), (*x,y+1,z*), (*x,y-1,z*), (*x,y,z+1*) and (*x,y,z-1*). Note that the *+1* and *–1* in the node addresses do not imply any numeric order or semantic similarity, but a logical neighborhood relation only.

A user who provides multiple data objects, e.g., UMs from recommender systems using different UM representations, or UMs from multiple application domains, maintains a set of hypercube locations. This means that a separate data structure of logical neighbors is maintained for UM and the respective hypercube location. In summary, the overall hypercube is distributed and virtually made of the connected users, whereas each user maintains a set of data structures that allow its neighbors to be located and connectivity of the hypercube to be maintained.

HyperCuP proposes a dynamic P2P algorithm for hypercube construction and maintenance. The algorithm is based on the idea that a user providing a particular data object can manage not only a single node to which the data object is mapped, but also a set of *virtual* neighbor nodes that have not yet been assigned to other data objects and users. This is required in order to simulate the missing nodes for the purposes of maintaining connectivity and guarantee the network routing. For example, consider a *3*-dimensional hypercube with five data objects shown in Figure 17a. In this case, node *4* is simulating three missing nodes of the hypercube. This is schematically shown by the dashed edges *1-4*, *2-4* and *3-4*, as node *4* acts as a virtual neighbor of nodes *1*, *2* and *3*.

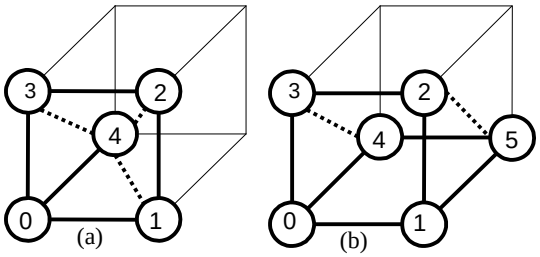


Fig. 17. Hypercube Maintenance in HyperCuP (a) Implicit topology of the complete hypercube; (b) Insertion of a new data object

When a new data object is inserted by one of the users, the ontological description of the data object is mapped to a certain network node, and the user providing the data object replaces the user who simulated this node and starts functioning as a real node. For example, consider an insertion of a data object that is positioned in node *5*, as shown in Figure 17b. A user providing the new data object becomes a neighbor of nodes *1* and *4*, and starts simulating the missing neighbor

of node *2*. Conversely, when a user providing the data object disconnects, one of the remaining neighbors starts managing the node previously managed by the leaving user. For an extensive description of the HyperCuP topology maintenance and examples of new data object insertions and node departures, the readers are referred to [Schlosser et al. 2002].

7.2 Unspecified Representation of User Modeling Data

As mentioned earlier, the ontology of HyperCuP is defined prior to the data object insertions and node creations. Thus, the data objects are described using a fixed set of ontology slots and features. This implies different domains to be modeled a-priori by human experts and inherently requires a central management of the ontology that cannot evolve and be modified over time. These characteristics of HyperCuP contradict the decentralized and highly dynamic characteristics of P2P networks, and appeal for a more flexible approach to the representation of the data objects in a fully distributed P2P environment.

7.2.1. Data Management using UNSO

UNSO (UNSpecified Ontology) [Ben-Asher and Berkovsky 2006] extends the ideas of semantic data management over a hypercube graph of data objects generated over the P2P middleware. The main contribution of UNSO is addressing and resolving the HyperCuP's dependence on a fixed, and a-priori defined ontology. Unlike HyperCuP, UNSO assumes that the ontology is not fully defined and parts of it can be incrementally specified by the users when inserting their data objects. Hence, the descriptions of data objects are modeled by a flexible *unspecified* list, where the term unspecified refers to the fact that the data objects are described by a list of features and their respective values: $<feature_1:value_1, feature_2:value_2,\ldots, feature_n:value_n>$. In this list, $feature_i$ corresponds to feature *i* mentioned in the list and $value_i$ to the value of this feature.

To cope with pure decentralized management of unspecified data objects, UNSO generalizes the HyperCuP's notion of ontology-based mapping. Since the unspecified description by list of $feature_i:value_i$ pairs can grow incrementally when new features are introduced, two hashing functions are used to map the descriptions of the data objects to the hypercube. The first function $hash_1$ maps the feature name $feature_i$ to a dimension of the hypercube, while the second function $hash_2$ maps the respective value $value_i$ to a numeric coordinate value within that dimension.

For example, consider the following unspecified description of a demographic user modeling data object: <*gender:female*, *age:20*, *residence:NY*>. It is inserted by applying $hash_1(gender)$, $hash_1(age)$, and $hash_1(residence)$ to obtain the relevant dimensions of the hypercube and $hash_2(female)$, $hash_2(20)$ and $hash_2(NY)$ to determine the numeric coordinate values within these dimensions. In summary, the coordinate of this data object in the dimension $hash_1(gender)$ is $hash_2(female)$, in the dimension $hash_1(age)$ the coordinate is $hash_2(20)$, and in the dimension $hash_1(residence)$ it is $hash_2(NY)$.

Since UNSO exploits a hashing mechanism for mapping of data objects to the underlying hypercube, it is capable of managing descriptions whose *features* and *values* are not defined a-priori by the ontology. In addition, the hashing-based mapping mechanism of UNSO allows users to insert data objects without any predefined order of $feature_i:value_i$ pairs in the list, increasing even more the flexibility of the system. On the other hand, this flexibility introduces new data management problems, as different users may use different terms, e.g., synonyms, hyponyms, or hypernyms, to describe the same underlying semantic concept. This issue will be discussed at the end of this section.

Analyzing the unspecified descriptions of various data objects, one can recognize that certain features are not independent and appear only if another feature, or even a specific value, appears in the description of the data object. For example, consider a demographic UM feature *children* expressing the number of children of a user. Obviously, this feature should not be mentioned in the UMs of users, whose feature *age* has value *15* or lower. To handle such dependencies in a flexible manner, the data objects are organized within the hypercube in a *hierarchical* multi-layered structure, rather than using a *flat* representation. This organization converts the pure hypercube structure of HyperCuP to a hypercube-like structure, whose nodes recursively contain other hypercubes. This structure is referred to in [Ben-Asher and Berkovsky 2006] as a multi-layered hypercube (MLH). Figure 18 schematically illustrates the organization of the MLH.

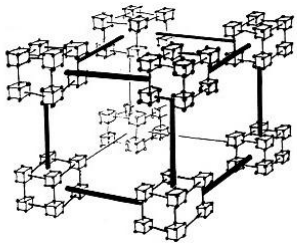


Fig. 18. Multi-Layered Hypercube of UNSO

For example, consider a *2*-layered unspecified description of demographic UMs containing three ontological features in each layer: <*gender*, *age*, *residence*> + <*children*, *occupation*, *salary*>. For the sake of simplicity, let us assume that only two values are possible for each feature. This leads to an *outer* hypercube with up to *8* nodes, recursively containing *inner* hypercubes of up to *8* nodes. In practice, when the UMs are inserted, only some nodes of the first layer <*gender*, *age*, *residence*> are expanded to *6*-dimensional nodes. The generated MLH should be compared to a fixed size *64*-node hypercube, had we used one list of *6* features for the descriptions of the data objects.

If the number of $feature_i$:$value_i$ pairs in the unspecified description of a new data object is lower than the number of dimensions in the respective MLH, the description is automatically expanded to the maximal dimension of the MLH. When the description is expanded, the missing features are assigned an *unknown* value, denoted by *Ø*. For example, consider a UM <*gender:female*, *age:20*, *residence:NY*> + <*children:2*> inserted into the above mentioned *2*-layered *6*-dimensional MLH. The description is expanded to <*gender:female*, *age:20*, *residence:NY*> + <*children:2*, *occupation:Ø*, *salary:Ø*>, and the expanded description is mapped to the MLH. Conversely, if a new data object introduces a *feature* that has not been mentioned in the descriptions of the existing data objects, a new dimension is added to the lowest-level (most inner) hypercube of the MLH. Note that the new dimension is added only to the lowest-level hypercube, whose location reflects the other features mentioned in the description of an object that introduces a new feature, whereas the rest of the MLH is not affected. These expansions constitute the advantages of UNSO over HyperCuP, as the structure of the MLH is highly dynamic and reflects the features mentioned in the descriptions of the data objects.

The differences between a fixed specified ontology and the Unspecified Ontology are summarized in Figure 19. The upper part of Figure 19 shows the mapping of the data objects, described

according to a predefined ontology of HyperCuP to one of the underlying hypercube nodes. The ontology contains three features <f_1, f_2, f_3>, whereas each feature has a predefined set of possible values, e.g., values v_{11}, v_{12}, …, v_{1n} for a feature f_1, values v_{21}, v_{22}, …, v_{2n} for a feature f_2, and v_{31}, v_{32}, …, v_{3n} for a feature f_3. The features are a-priori assigned to the dimensions of the hypercube, and the values of the features are assigned numeric coordinates within these dimensions. As such, a data object described by <f_1:v_{1i}, f_2:v_{2j}, f_3:v_{3k}> is mapped to a location of the hypercube, schematically denoted in Figure 19 by <v_{1i}, v_{2j}, v_{3k}>.

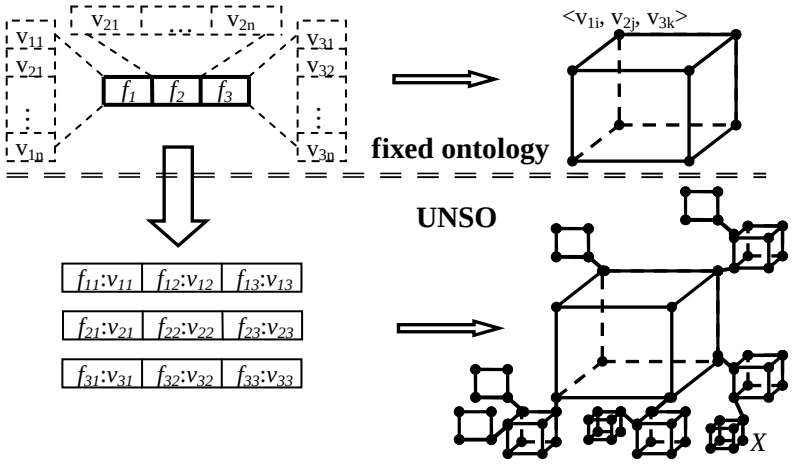


Fig. 19. Generalization of the Fixed Ontology of HyperCuP to UNSO

Conversely, in UNSO (the bottom part of Figure 19) the $feature_i$:$value_i$ pairs in the description of a data object are partitioned to multiple layers, the number of pairs in the descriptions is unlimited, their order within the layers is insignificant, and the set of possible *features* and *values* that can be mentioned in the description is unlimited. For example, consider an unspecified *3*-layered description <f_{11}:v_{11}, f_{12}:v_{12}, f_{13}:v_{13}> + <f_{21}:v_{21}, f_{22}:v_{22}, f_{23}:v_{23}> + <f_{31}:v_{31}, f_{32}:v_{32}, f_{33}:v_{33}>. To map the description to one of the MLH nodes, the features are hashed to the dimensions of the relevant hypercube, and the values are hashed to the numeric coordinates within these dimensions. For example, consider the above *3*-layered description mapped to a node denoted by *X* in Figure 19[9].

Another inherent drawback of UNSO should be mentioned: different users may use different terms to describe the same underlying semantic concept in the unspecified descriptions of the data objects. To address the problem that the data objects may have different terms with the same se-

[9] For the sake of clarity, Figure *19* shows the inner hypercubes of the MLH as a separate hypercubes, 'hung' on the nodes of the outer hypercubes. Despite this, we stress again that the structure of the hypercubes is recursive, where the outer hypercubes are constructed of the inner hypercubes.

mantic meaning, e.g., synonyms, hyponyms or hypernyms, the terms mentioned in $feature_i$ and $value_i$ descriptions are standardized using WordNet [Fellbaum 1998]. In WordNet, English nouns, verbs, adjectives and adverbs are organized into synonym sets, each representing a single lexical concept. To eliminate possible ambiguity and improve the accuracy of the data management, the terms mentioned in the descriptions of the data objects undergo a simple semantic standardization. This standardization substitutes the terms used in the unspecified description by their most frequent synonyms. For example, the terms *residence*, *dwelling*, and *abode* are substituted by their most frequent synonym *residence*. Hence, similar but not identical terms are replaced by a single representative term.

We would like to stress the main advantages of UNSO over a HyperCuP. First, UNSO allows the data objects to be described in a flexible manner, not implying any central ontology. New types of features and values can be introduced: new features are reflected by new dimensions of the low-level hypercubes, whereas new values are just mapped to their numeric values in the appropriate dimensions. Second, the hashing-based mapping mechanism of UNSO allows the order of features and values in the descriptions of the data objects to be disregarded. Third, semantic standardization of the unspecified descriptions using WordNet facilitates enhanced data management capabilities. Finally, the MLH structure of the underlying P2P network allows flexible descriptions of the data objects, expressing the dependencies between various features and values. As a result, the generated structure is denser than in a flat representation of HyperCuP (only realistic combinations of features and values are expressed), it is easier to maintain in a P2P environment, and it allows more efficient operations to be conducted.

The negative effect of hashing in UNSO is losing the order relation among the values of the features, since the hashing function mapping the values to the hypercube dimensions does not keep the order relation. Hence, for instance, two values *20* and *21* can be mapped to absolutely different coordinates within the dimension of *age* and not reflect their proximity. Despite this, UNSO supports the notion of concept clustering, defined in [Schlosser et al. 2002]. This is explained by the observations that data objects, whose descriptions are identical with respect to a subset of mentioned features, are mapped to locations that have a common subset of identical coordinates. For example, consider two UMs of a demographic recommender system <*gender:female, age:20, residence:NY*> and <*gender:female, age:21, residence:NY*>. These UMs are mapped to relatively

close locations in the hypercube due to the fact that their values in the dimensions of *gender* and *residence* are identical, and they differ only in the dimension of *age*[10]. Moreover, also the UMs <*gender:female, age:20, residence:NY*> and <*gender:female, age:20, children:2*> will be located closer than two arbitrary UMs, as two out of three features in their descriptions overlap.

7.2.2. User Modeling Data Representation and Storage

This work adopts the unspecified description of data objects proposed by UNSO, and uses it as the user modeling data description language in recommender systems. Hence, we represent the UMs by dynamic lists of features and their respective values. Different application domains may exploit different features or values to describe a UM. For example, a movies recommender system may store features describing the preferred genre and directors, whereas a restaurant system may store various gastronomic and environmental features. As such, neither the set of features specified when describing user modeling data, nor their values can be defined a-priori. Hence, the main advantage of UNSO is in facilitating management of heterogeneous representations of user modeling data and specific UMs described according to these representations.

Consider a *3*-dimensional MLH, whose dimensions represent demographic features {*gender, age, residence*}. The mapping of these features to the dimensions of the hypercube is shown in Figure 20a. UMs of three users are stored by the system: user *A* is described by <*gender:female, age:30, residence:NY*>, user *B* by <*gender:male, age:30, residence:NY*>, and user *C* by <*gender:male, age:20, residence:NY*>. Let us assume that the user modeling data of user *A* is mapped to the top-left node, of user *B* – to top right node, and of user *C* – to bottom-right node, as shown in Figure 20b. Note that the UMs of users *A* and *B* differ in the value of the *gender* feature, while the UMs of users *B* and *C* differ in the value of the *age* feature.

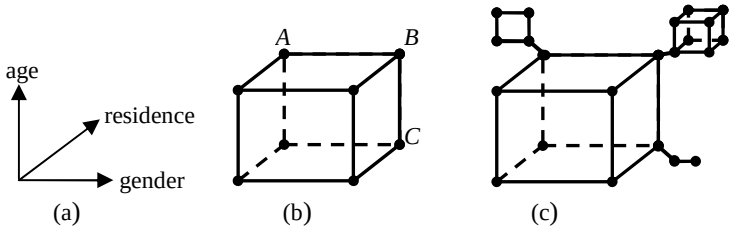


Fig. 20. Generation of an MLH

[10] The clustering may be inaccurate due to the hashing collisions. The probability of such collisions decreases with the number of $feature_i$:$value_i$ pairs and the range of the hashing function.

Later, other UMs are inserted into the system and they introduce new features, i.e., features that were not mentioned in previously inserted UMs. For example, two new features *children* and *occupation* are introduced among *30* year-old female users, three new features *salary*, *hobby*, and *car owned* are introduced among the *30* year-old male users, and one new feature *is a student* is introduced among *20* year-old male users. This causes the nodes of the outer MLH to be split into inner hypercubes, such that node *A* is converted to a *2*-dimensional hypercube reflecting the features *children* and *occupation*, node *B* is converted to a *3*-dimensional hypercube reflecting the features *salary*, *hobby*, and *car owned*, and node *C* is converted to a *1*-dimensional hypercube reflecting the feature *is a student*. The resultant structure is shown in Figure 20c[11].

Note that the resultant MLH generated by UNSO reflects unspecified descriptions containing *9* different features: *3* in the outer hypercube, and another *6* in various inner hypercubes. This should be compared to a flat *9*-dimensional hypercube that would have been generated had we used HyperCuP. Note that further insertions of new features, not shown in this example, may cause the generated MLH to be expanded to a structure similar to the MLH shown in Figure 19.

We would like to stress again the observation that when UNSO is exploited for the storage of user modeling data, it groups similar UMs, i.e., UMs differing in a small number of features, in close nodes. For example, in the above scenario of demographic UMs, users *A* and *B*, and users *B* and *C*, differ only in the value of one feature. As a result, their locations in the MLH differ only in coordinates within one dimension. However, the UMs of users *A* and *C* differ in values of two features and their locations differ in two dimensions. Hence, the distance in terms of network locations is greater than the distance between the UMs of users *A* and *B*, or *B* and *C*[12]. The above example shows that UNSO preserves the semantic similarity of the UMs. This statement will be clarified later, when we will introduce the UMs similarity metric and will use the mapping of the UMs to the coordinates of the MLH for retrieving the set of the most similar UMs.

[11] For the sake of simplicity, the figures illustrate binary hypercubes with two possible values in each dimension: *0* and *1*. Actually, the hypercubes are not binary and the range of values in each coordinate is defined a-priori.

[12] Note that the real distance between the UMs is calculated by comparing their features and values. Here we use a heuristic that real distance is correlated with the number of different coordinates.

7.3 User Modeling Data Similarity and Retrieval over UNSO

Efficient and accurate retrieval of the most-similar UMs is one the tasks performed by various recommender systems, such as collaborative filtering [Herlocker et al. 1999], demographic filtering [Krulwich 1997], and some hybrid recommender systems [Burke 2002]. One of the following two basic criteria for such retrieval are typically exploited:

- Retrieve the whole set of UMs, whose similarity with the target UM is above a given threshold β – typically obtained by computing the similarity for every available UM and returning it if the similarity value is higher than β.
- Retrieve a set of K UMs most-similar to the target UM – typically obtained by computing the similarity for every available UM, ranking the UMs according to their similarity values, and returning the K highest UMs from the ranked list.

In principle, both criteria are similar, as they are aimed at retrieving the most similar UMs to the UM of the target user through an exhaustive comparison of the target UM with the other UMs stored by the system. In fact, in many conditions these techniques are interchangeable; in certain conditions, however, one of the policies may be preferred over the other. For example, consider a system, where the number of potentially similar UMs is high. In this case, retrieving all the UMs, whose similarity is above a certain threshold, may find a large number of UMs, such that limiting the size of the set to K may be beneficial. Conversely, when the number of stored UMs is small, retrieval of K most similar UMs may find UMs that are not sufficiently similar to the target UM. In this case, retrieving all the UMs above a certain threshold may be preferred.

7.3.1. Similarity Metrics

The above mentioned retrieval policies require a similarity metric between the UMs to be defined explicitly [Bernstein et al. 2005]. In this work, the similarity of two UMs c_1 and c_2 is computed by $(1–dist(c_1,c_2))$, where $dist(c_1,c_2)$ is their normalized distance metric between 0 and 1. When the UMs are represented as a list of $feature_i:value_i$ pairs, each feature is treated separately and the individual distances within the features are combined together. Hence, the distance between two UMs c_1 and c_2 is computed by:

$$dist(c_1, c_2) = [\sum_{i=1}^{|F|} w_i \cdot dist(c_1^i, c_2^i)^2]^{1/2}$$

where F denotes the union of features specified in both UMs, i.e., the features that appear in at least one UM, w_i denotes the normalized weight of the feature f_i, and $dist(c_1^i, c_2^i)$ denotes the normalized distance metric between c_1 and c_2 with respect to the values of the feature f_i.

To compute the local distance between two values of a feature, we consider the possible types of the values [Coyle et al. 2004]. For Boolean features with possible values of *true* or *false*, the distance is a trivial comparison between two values, giving *0* if the values are equal or otherwise *1*. For symbolic or free-text values, the distance can be potentially computed either as a difference between the numeric representations of the values reflecting their relation, e.g., by setting *CA=0*, *KS=0.5*, and *NY=1* for possible state values of the feature *residence*, or through exploiting a distance matrix assigning a distance value to each pair of values. Although such definition of distances can be performed manually by the domain experts, it is unclear whether it is applicable when both the features and their values are not defined a-priori. Hence, this work defines the distance metric for symbolic and free-text values similarly to the Boolean features:

$$dist_{Boolean}(c_1^i, c_2^i) = dist_{Free-Text}(c_1^i, c_2^i) = \begin{cases} 0 & c_1^i = c_2^i \\ 1 & \text{otherwise} \end{cases}$$

We would like to stress that the exact matching distance metric strongly depends on the underlying semantic standardization mechanism. Since the terms used in the descriptions of the UMs are substituted by their most frequent synonyms, we assume that if similar but not identical terms are used, they are standardized and their distance metric is *0*. However, if the standardization mechanism is not capable of recognizing the similarity of the terms, their distance metric is *1* and they are treated as distinct.

For numeric features, the distance between the values is computed as the absolute value of their difference, normalized by dividing it by the maximal possible difference R, i.e., the maximal range of values of the given feature:

$$dist_{Numeric}(c_1^i, c_2^i) = \frac{|c_1^i - c_2^i|}{R}$$

Analyzing the available unspecified descriptions showed that the values of most of the features are either numeric, free-text, or Boolean. Thus, in this work we exploit only these three similarity metrics, and do not define metrics for more complex types of features and values, such as tree/graph nodes, enumerated values, and so forth, as suggested by [Coyle et al. 2004].

Since an unspecified description of the UMs does not assume any a-priori defined ontology, even the UMs from the same application domain may be described using different sets of features. As a result, even after standardization, two UMs may differ in terms of the mentioned features. Hence, the similarity metric should be capable of computing similarity between two UMs with partially overlapping sets of features. For this, we define a pessimistic distance measure between the values of the features, according to which the distance is *1* if a feature appears in the description of only one UM:

$$dist'(c_1^i, c_2^i) = \begin{cases} dist(c_1^i, c_2^i) & \text{feature } f_i \text{ appears in both cases} \\ 1 & \text{otherwise} \end{cases}$$

Consider the following example of similarity computation between two demographic UMs described by: c_1=<*gender:male, age:30, residence:USA, occupation:programmer, student:true*> and c_2=<*gender:female, age:20, residence:USA, children:2, student:true*>. The overall set of features mentioned in these UMs is {*gender*, *age*, *residence*, *children*, *occupation*, *student*}, whereas the set of overlapping features for these UMs is {*gender*, *age*, *residence*, *student*}. Let us assume that the featured *age* and *children* are numeric, *gender*, *residence* and *occupation* are free-text features, and *student* is a Boolean feature. Also, let us assume that the values of *age* vary from *0* to *120*. Hence, the distance between the UMs c_1 and c_2 is computed as follows:

$$dist(c_1, c_2) = [(w_{gender} \cdot 1)^2 + (w_{age} \cdot \frac{30-20}{120})^2 + (w_{residence} \cdot 0)^2 + (w_{children} \cdot 1)^2 + (w_{occupation} \cdot 1)^2 + (w_{student} \cdot 0)^2]^{1/2}$$

7.3.2. Similarity-Based Retrieval over UNSO

In this work we propose exploiting the dynamic infrastructure of UNSO and grouping of similar UMs as an indexing approach to improving the efficiency of the retrieval of similar UMs. Intuitively, we base our retrieval on the assumption that similar UMs are grouped by UNSO, and therefore located in close vicinity. Hence, the retrieval of similar UMs is conducted as a localized search in the MLH starting from the node where the UM of the target user is located, and iteratively checking the logical neighbor nodes and comparing the UMs stored in these nodes with the target UM. Figure 21 presents the pseudo-code of the proposed algorithm for retrieval of UMs, whose similarity with the target UM is above a given threshold β.

```
Retrieve (TARGET-UM, β)
(1)  map TARGET-UM to the hypercube of dimension n
(2)  assume the location of TARGET-UM is (c1,…,cn)
(3)  let RETRIEVED-UMs be a set of UMs, initially empty
(4)  for each dimension i from 1 to n
(5)    for each x in the range of values for coordinate i
(6)      let CURRENT be the set of UMs stored in
         the location (c1,…,ci-1,x,ci+1,…,cn)
(7)      for each TEST-UM∈CURRENT
(8)        if similarity(TARGET-UM, TEST-UM) > β
(9)          RETRIEVED-UMs = RETRIEVED-UMs ∪{TEST-UM}
(10) return RETRIEVED-UMs
```

Fig. 21. Algorithm for Retrieving UMs with Similarity Metric above β

Initially, the algorithm determines the location of the target UM, i.e., the *TARGET-UM*, using a hashing-based mapping mechanism for inserting the UMs into the hypercube (steps *1-2* of the pseudo-code) and initializes an empty set of retrieved UMs, referred to as *RETRIEVED-UMs* (step *3*). Then the algorithm analyzes the candidate UMs *TEST-UMs* stored in the neighbor locations of the underlying P2P network by assessing the similarity between each *TEST-UM* and the *TARGET-UM* using the above distance metric (steps *4-7*). The rationale is that similar UMs should be located in close vicinity and, therefore, to find similar UMs, we should check only a small portion of the available UMs. In other words, we exploit an implicit indexing and organization of the UMs provided by UNSO. If the similarity value is higher than the given threshold *β*, the candidate *TEST-UM* is added to the *RETRIEVED-UMs* set of UMs with the required similarity (steps *8-9*). Finally, the whole set of appropriate UMs *RETRIEVED-UMs* is returned (step *10*).

Note that the retrieval process is conducted in UNSO-based MLH through a propagation of queries and the respective answers from the target node to its neighbors and so on. Consider a target UM mapped to a location *(x,y,z)* that is connected to *6* immediate neighbors (*x+1, y, z*), (*x-1, y, z*), (*x, y+1, z*), (*x, y-1, z*), (*x, y, z+1*) and (*x, y, z-1*). Consider a search at the neighbor nodes over the first dimension, i.e., at locations (*x+1, y, z*) and (*x-1, y, z*). When these nodes receive the similarity computation request, the similarity of the UMs stored there is computed and the request is propagated to the next logical neighbors along the relevant dimension, i.e., to nodes (*x+2, y, z*) and (*x-2, y, z*). The same propagation occurs also in the other dimensions of the hypercube. Thus, the retrieval process can be described as a propagating expanding search, where the similarity computations of the UMs and the communication activities required by the retrieval are parallelized. Consider again the above example of the target node mapped to a location *(x, y, z)*. When *6*

immediate logical neighbor nodes receive the similarity computation request, all *6* computations can be conducted in parallel by the respective users. This allows additional optimization of the retrieval process.

For example, consider the target UM c_t=<*gender:male, age:30, residence:USA*> stored in a *3*-dimensional hypercube. Assume that c_t is mapped to a location (*5, 6, 7*) in the hypercube. The retrieval compares c_t with the candidate UMs in the locations (**, 6, 7*), (*5, *, 7*), and (*5, 6, **), where * denotes any possible value in the respective dimension. To accomplish the search for all the UMs stored in (**, 6, 7*), a node (*5, 6, 7*) storing the target UM queries its immediate logical neighbors, i.e., sends the description of c_t to nodes (*4, 6, 7*) and (*6, 6, 7*). Upon receiving this query, these nodes autonomously perform three operations: (i) forward the description of the target UM c_t to their next logical neighbors, i.e., nodes (*3, 6, 7*) and (*7, 6, 7*), (ii) compute the similarity between the target UM c_t and the UMs stored in their nodes, and (iii) back propagate the similarity value to the location of c_t over the same route by which the query was received. As a result, the propagated similarity computation both parallelizes the retrieval of most similar UMs and distributes the required computational overhead among the MLH nodes, i.e., among the users managing these nodes.

The retrieval of top-*K* most similar UMs is performed in a similar manner, except a change that the length of the set of *RETRIEVED-UMs* is limited to *K*. Since at any given point of time *RETRIEVED-UMs* stores *K* best candidates for being the most similar UMs, the UMs stored in the *RETRIEVED-UMs* set are sorted according to their similarity values with the *TARGET-UM*. For this, every *TEST-UM* is inserted into the *RETRIEVED-UMs* in a way that keeps the whole set sorted. Figure 22 presents the pseudo-code of the algorithm for retrieval of *K* most similar UMs.

```
 Retrieve (TARGET-UM, K)
(1)  map TARGET-UM to the hypercube of dimension n
(2)  assume the location of TARGET-UM is (c1,…,cn)
(3)  let RETRIEVED-UMs be a set of size K, initially empty
(4)  for each dimension i from 1 to n
(5)    for each x in the range of values for coordinate i
(6)      let CURRENT be the set of UMs stored in
         the location (c1,…,ci-1,x,ci+1,…,cn)
(7)      for each TEST-UM∈ CURRENT
(8)         compute similarity(TARGET-UM, TEST-UM)
(9)           insert TEST-UM into RETRIEVED-UMs s.t. RETRIEVED-UMs
              is sorted according to the similarities of the UMs
(10) return RETRIEVED-UMs
```

Fig. 22. Algorithm for Retrieving *K* most Similar UMs

Let us denote by Δ the maximal number of coordinates in the location of the candidate UM, whose values are modified with respect to the target UM[13]. The above algorithms in Figures 21 and 22 retrieve an approximated set of the most similar UMs, where the value of up to $\Delta=1$ coordinate is allowed to be modified. As can be seen from the pseudo-code, the loop in line *(4)* iteratively scans the dimensions one by one, whereas the loop in the line *(5)* modifies the values within the given dimension and retrieves the UMs. Intuitively, the decision to limit the retrieval to $\Delta=1$ modified coordinates is motivated by the observation that a higher number of modified coordinates is typically reflected in a lower similarity of UMs[14]. Thus, the probability of discovering highly similar UMs decreases with the number of modified coordinates.

However, in certain conditions the retrieval with $\Delta=1$ modified coordinates may not be sufficient. For example, consider retrieval with a low threshold β, or retrieval aimed at retrieving a large number K of similar UMs. In this case, the retrieval with $\Delta=1$ modified coordinates may not find a sufficient number of UMs. Moreover, in a system with a small number of stored UMs and a large number of dimensions, such retrieval misses many UMs that would be retrieved by the traditional exhaustive retrieval. Hence, in these conditions there is a need for a deeper retrieval, where the values of $\Delta>1$ coordinates are allowed to be modified. In order to adapt the algorithm, the loop in the line *(4)* of the above pseudo-code should be replaced by a nested loop, which allows the values of multiple coordinates to be modified in parallel, and, therefore, retrieves UMs with a higher number of modified coordinates.

For the sake of simplicity, the above algorithms describe the retrieval of similar UMs over a flat hypercube and not over the MLH. However, converting the algorithms to the MLH will have only a minor impact. The only change that should be stressed is that over the MLH, the search for the modified values within certain coordinates partially takes place in other inner hypercubes. For example, in a *2*-layered *3*-dimensional MLH, whose dimensions correspond to features {$feature_1$, $feature_2$, $feature_3$}+{$feature_4$, $feature_5$, $feature_6$}, searches for the modified values within the coordinates of $feature_4$, $feature_5$, and $feature_6$ are handled in the inner hypercube, corresponding to the values of $feature_1$, $feature_2$, and $feature_3$ in the target UM. Conversely, searches for the modi-

[13] Note that the actual number of different feature values may be higher than Δ due to the hashing collisions occurring when the UMs are inserted into the MLH. In other words, two UMs having the same coordinate values in a certain dimension may have different values of the feature in that dimension.

fied values within the coordinates of *feature*$_1$, *feature*$_2$, and *feature*$_3$ require accessing UMs stored in the inner hypercubes, corresponding to other nodes of the outer hypercube, i.e., sibling inner hypercubes.

For a system storing *N* UMs, the number of UM comparisons in the exhaustive retrieval is *O(N)*. Intuitively, the complexity of UNSO-based retrieval is lower, as the target UM is compared with a subset of the stored UMs. The complexity of UNSO-based retrieval for $\Delta=1$ is *O(nk)*, where *n* is the number of dimensions in the MLH storing the target UM, and *k* is the range of coordinates within these dimensions (assuming that it is similar in all the dimensions of the MLH). Hence, for *n=10* Boolean coordinates, UNSO-based retrieval with $\Delta=1$ conducts *10x2=20* comparisons, which is much lower than the potential maximal cardinality of $2^{10}=1024$. It is important to note that in practice a much smaller number of UM comparisons will be performed. Due to the sparsity of the data, not all the possible UMs differing only in $\Delta=1$ coordinate with respect to the target UM are actually present in the MLH. Hence, the proposed algorithm compares the target UM with only those UMs that are actually stored in the nodes having up to $\Delta=1$ coordinates different. In the general case of *n* features mentioned in the unspecified description of the target UM, Δ features whose values are allowed to be modified, and *k* possible values for each one of the features, the number of UMs that are compared is:

$$O(k^{\Delta}C_{\Delta}^{n}) = O(k^{\Delta}\frac{n!}{\Delta!(n-\Delta)!})$$

In any case, the number of similarity comparisons that are actually performed in UNSO-based retrieval is bounded by the number of comparisons in the exhaustive retrieval, and this occurs when Δ is equal to the dimension *n*.

In summary, UNSO facilitates maintenance of the stored UM in a distributed hypercube-like graph with a stable connected structure. Grouping of similar UMs in UNSO facilitates retrieval of similar UMs through a simple expanding search. The proposed algorithm allows efficient and accurate retrieval of similar UMs, while spreading the required computational effort among the users. In the next section we present and analyze the empirical evaluation of the proposed retrieval algorithm.

[14] The similarity values depend only on the values of the features used for the similarity computation. However, a high number of modified coordinate values typically indicates also on a lower similarity of the UMs.

7.4 Experimental Evaluation

Storage of user modeling data in the state-of-the-art recommender systems is typically based either on ontology-based representation [Heckmann et al. 2005] or on a representation dictated by the needs of the system, such as application domain and recommendation technique. Hence, the representation and the terminology of the user modeling data are homogeneous and the proposed UNSO approach is not required. Hence, for the experimental evaluation of the proposed algorithm, we used a set of real-life E-Commerce advertisements. These advertisements can be considered as ephemeral content-based search UMs, as they represent the needs of the users for a single search session. Since the advertisements were inserted by different users and were described in different ways, they mimicked the heterogeneously represented UMs. With respect to the above representation of UMs, the similarity-based search can be considered as a matching functionality provided by the system, i.e., search for other users that offer the requested items.

To validate the proposed UM representation and retrieval algorithm, we collected *5* corpora of such content-based UMs from the domains of *refrigerators*, *cameras*, *televisions*, *printers* and *mobile phones* (in short, *mobiles*). They were downloaded from *http://www.recycler.com* Website and manually converted to the form of an unspecified list. For example, an advertisement "Philips 50FD995 50" plasma television, new in box, $4800" was converted to the unspecified description *<price:4800, manufacturer:Philips, model:50FD995, screen:plasma, size:50, condition:new, in package:true>*. The conversions were performed so that the results would be as close as possible to the original contents of the advertisements and the respective UMs.

Table 10 presents various statistical properties of the corpora and features that appeared in the corpora. These properties include: the total number of UMs in a corpus ('UMs' column in Table 10), the number of unique features from every type that appeared in a corpus ('features' column, and its subcolumns 'Boolean', 'Numeric', 'Free-Text FT', and 'total'), the number of occurrences of various $feature_i$:$value_i$ pairs for every feature type in a corpus ('Number of pairs' column, and its subcolumns 'Boolean', 'Numeric', 'Free-Text FT', and 'total'), and the percentage of the $feature_i$:$value_i$ pairs from every type in a corpus ('% of pairs' column, and its subcolumns 'Boolean', 'Numeric,' 'Free-Text FT', and 'total'). The number of pairs is an important indicator of the corpus incompleteness. For example, if every UM in the corpus of refrigerators was described by all the

available features, then the corpus should contain in total *61*2+61*4+61*4=610 feature$_i$:value$_i$* pairs. However, the corpus contains only *254* pairs.

Table 10. Distribution of Features in the Corpora.

Corpus	**UMs**	**Features**				**Number of pairs**				**% of pairs**		
		Bool.	***Num.***	***FT***	***total***	***Bool.***	***Num.***	***FT***	***total***	***Bool.***	***Num.***	***FT***
refrigerators	*61*	*2*	*4*	*4*	*10*	*22*	*117*	*115*	*254*	*8.6*	*46.1*	*45.3*
cameras	*65*	*2*	*5*	*6*	*13*	*5*	*123*	*132*	*260*	*1.9*	*47.3*	*50.8*
televisions	*76*	*1*	*3*	*7*	*11*	*3*	*148*	*135*	*286*	*1.0*	*51.8*	*47.2*
printers	*94*	*5*	*2*	*4*	*11*	*48*	*100*	*293*	*441*	*10.9*	*22.7*	*66.4*
mobiles	*130*	*3*	*1*	*5*	*9*	*70*	*130*	*383*	*583*	*12.0*	*22.3*	*65.7*

We analyzed statistical properties of the features in the corpora. Table 11 summarizes the types and the names of the features in each corpus, the number of their occurrences in the UMs from the given corpus (denoted in Table 11 by o_f), the number of different values for each feature (denoted in Table 11 by v_f), and the total number of occurrences and values for the features from every type in every corpus (the *total* row in Table 11).

Table 11. Distribution of Features and Values from Different Types in the Corpora.

Corpus	***refrigerators***			***cameras***			***televisions***			***printers***			***mobiles***		
	name$_f$	***o$_f$***	***v$_f$***	***name$_f$***	***o$_f$***	***v$_f$***	***name$_f$***	***o$_f$***	***v$_f$***	***name$_f$***	***o$_f$***	***v$_f$***	***name$_f$***	***o$_f$***	***v$_f$***
Boolean	*delivery*	*6*	*2*	*package*	*3*	*2*	*package*	*3*	*2*	*cable*	*5*	*2*	*charger*	*51*	*2*
	ice-maker	*16*	*2*	*video*	*2*	*2*				*ink*	*19*	*2*	*manual*	*14*	*2*
										manual	*6*	*2*	*SIM*	*5*	*2*
										software	*7*	*2*			
										toner	*11*	*2*			
	total	**22**	**2**	**total**	**5**	**2**	**total**	**3**	**2**	**total**	**48**	**2**	**total**	**70**	**2**
Numeric	*age*	*11*	*6*	*memory*	*8*	*4*	*price*	*76*	*51*	*age*	*6*	*2*	*price*	*130*	*45*
	price	*61*	*28*	*price*	*65*	*37*	*size*	*70*	*20*	*price*	*94*	*38*			
	size	*38*	*25*	*resolution*	*12*	*7*	*year*	*2*	*2*						
	warranty	*7*	*2*	*warranty*	*2*	*2*									
				zoom	*36*	*12*									
	total	**117**	**61**	**total**	**123**	**62**	**total**	**148**	**73**	**total**	**100**	**40**	**total**	**130**	**45**
Free-Text	*color*	*23*	*8*	*body*	*4*	*4*	*color*	*5*	*4*	*condition*	*62*	*14*	*case*	*17*	*4*
	condition	*39*	*9*	*condition*	*28*	*10*	*condition*	*42*	*11*	*manufact.*	*91*	*13*	*condition*	*75*	*13*
	manufact.	*50*	*15*	*flash*	*6*	*4*	*manufact.*	*68*	*21*	*model*	*85*	*79*	*manufact.*	*129*	*16*
	model	*3*	*3*	*focus*	*6*	*2*	*material*	*11*	*1*	*type*	*55*	*4*	*model*	*121*	*63*
				manufact.	*63*	*22*	*model*	*5*	*5*				*network*	*41*	*3*
				type	*25*	*5*	*ratio*	*3*	*2*						
							type	*1*	*1*						
	total	**115**	**35**	**total**	**132**	**47**	**total**	**135**	**45**	**total**	**293**	**110**	**total**	**383**	**99**

An UNSO-based model for storage and retrieval of UMs was implemented using in Java. In the implemented prototype, every corpus was assigned a separate MLH. Version *2.0* of WordNet was

used to standardize the terms in the unspecified features and values. The number of dimensions in the MLHs was not limited, i.e., it was equal to the number of different features mentioned in the UMs, as shown in Table 10. The cardinality of the MLH dimensions, i.e., the range of coordinate values in each dimension, referred to as *card*, was set to *7*.

There is a tradeoff between the values of *card* and the performance of the system. On the one hand, low values of *card* increase probability of hashing collisions, where two different values of the same feature are mapped to the same numeric coordinate in a certain MLH dimension. As a result, retrieval capabilities of the system are hampered, since the system may retrieve UMs that are located in close locations due to the hashing collisions, whereas their contents are actually different. In addition, low values of *card* may impair the distribution of UMs among the MLH nodes and lead to high computational overheads, as the number of comparisons in every node increases. On the other hand, high values of *card* may generate large and sparse MLHs, which are hard to maintain in a pure decentralized P2P environment. The value of *7* was chosen for the purposes of keeping the size of the MLHs reasonably small, while allowing the system to demonstrate reasonable performance capabilities (a conclusions of [Ben-Asher and Berkovsky 2006]).

7.4.1. Grouping of Similar User Models

One of the basic properties of UNSO-based MLH is the property of *grouping*, i.e., the fact that similar UMs are mapped by UNSO to close locations. This experiment was aimed at validating this property for the *5* corpora of UMs. As no explicit distance metric for the locations of UMs was defined, we assess distances between a pair of UMs in UNSO by the number of modified coordinates in their locations.

In every execution of the experiment, one of the UMs was selected to be the target UM. For each one of the other UMs, the number of modified coordinates Δ with respect to the target UM and the similarity of the UMs were computed. The experiment was repeated for each UM in the corpora acting as the target UM. The overall similarity of UMs for a given number of modified coordinates Δ was computed as the average similarity of UMs over the different executions and target UMs. Figures 23 and 24 show, respectively, the average similarity of UMs and the percentage of UMs located at a certain number of modified coordinates Δ for the above five corpora. The figures shows percentage of UMs and average similarity for Δ ranging from $\Delta=1$ to $\Delta=9$.

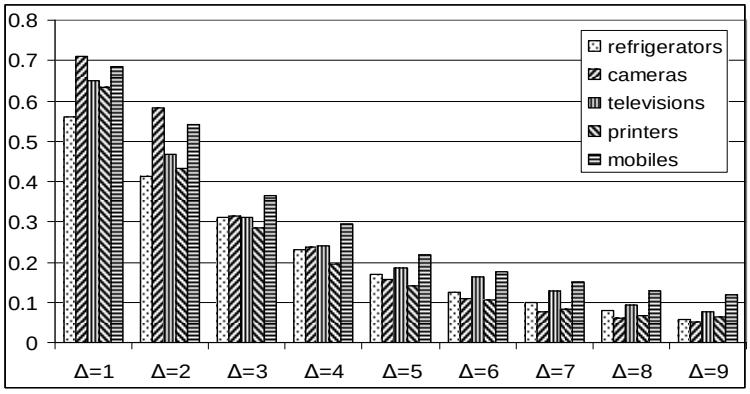


Fig. 23. Average Similarity of UMs for Various Values of Δ.

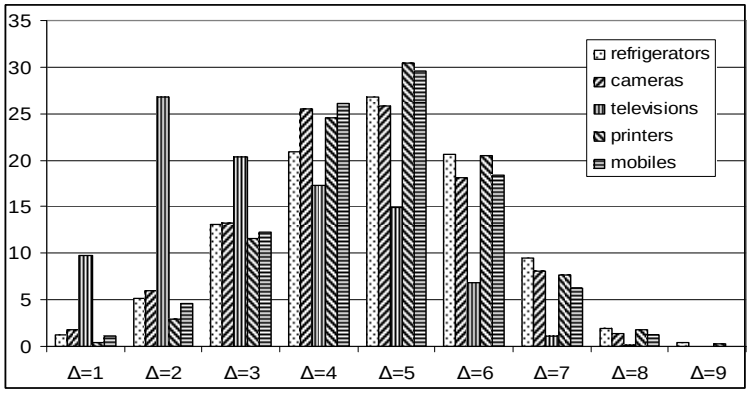


Fig. 24. Percentage of UMs for Various Values of Δ.

Figure 23 shows that the average similarity of UMs monotonically decreases with the number of modified coordinates Δ. This validates our basic assumption regarding the grouping of similar UMs in UNSO-based MLH, as the average similarity of UMs with a small number of modified coordinates is higher than the average similarity of UMs with a large number of modified coordinates. Although there is a certain difference between the similarity values in various corpora (the reasons will be discussed later), the general observation regarding the monotonic decrease of the average similarity of UMs is valid for all the corpora and for all the values of Δ.

Despite this, we should note that this observation is intuitive, since the similarity of UMs depends not only on the number of the modified coordinates Δ, but also on the specific features and values in the UMs. For example, consider two pairs of UMs: c_1=<*age:30*, *residence:NY*> and c_2=<*age:30*, *residence:CA*>, and c_3=<*age:30*, *children:2*> and c_4=<*age:31*, *children:3*>. Assuming that there are no hashing collisions in the mapping of UMs to the MLH, the number of modified coordinates between c_1 and c_2 is $\Delta=1$, while between c_3 and c_4 it is $\Delta=2$. However, computing the similarity of UMs shows that the similarity of c_1 and c_2 is lower than the similarity of c_3

and c_4. This is explained by the different types of feature in the UMs, as the modified feature between c_1 and c_2 is a Free-Text feature, while between c_3 and c_4 they are Numeric. Thus, the value of Δ is only an intuitive indicator regarding the similarity of UMs.

As can be seen in Figure 24, the percentage of UMs located at a certain number of modified coordinates Δ shows a bell-curve behavior in all the corpora. For a low value of Δ, the percentage of UMs is low. Then, the percentage of UMs increases with Δ and finally it decreases, as the percentage of UMs with a high number of modified coordinates Δ is low. Note that there is a difference between the percentages of UMs for a certain Δ in the corpora, as the bell curve of the televisions corpora is shifted towards the low values of Δ in comparison to the other corpora. For example, for $\Delta=1$, $\Delta=2$, and $\Delta=3$ the percentage of UMs in the televisions corpus is significantly higher than in the other corpora. Conversely, for higher values of Δ, it is lower than in the other corpora. This means that the percentage of UMs with a lower number of modified coordinates in the televisions corpus is higher that in the other corpora, i.e., television UMs are *grouped better* and generate a denser structure than the other corpora. The reasons for this behavior will be explained later through an analysis of various statistical properties of the UMs in the corpora. This characteristic of the television UMs will affect the performance of the proposed retrieval method as will be discussed later.

7.4.2. Retrieval Capabilities

This experiment was designed to evaluate the accuracy of the proposed UNSO-based retrieval. In each execution, one UM was selected to be the target UM, and the retrieval was conducted in two ways. First, we retrieved the true set R_e of the most similar UMs using a traditional exhaustive retrieval of UMs, whose similarity was above a given threshold β. Second, we retrieved the approximated set of the most similar UMs R_u using the proposed UNSO-based retrieval algorithm. Finally, two retrieved sets R_e and R_u were compared. This process was repeated for each possible target UM for a number of times equal to the number of UMs in the corpus.

7.4.2.1 Recall Experiments

The accuracy of the proposed algorithm was evaluated using the *recall* metric, adapted from Information Retrieval [Salton and McGill 1983]. Recall is computed as the proportion of the rele-

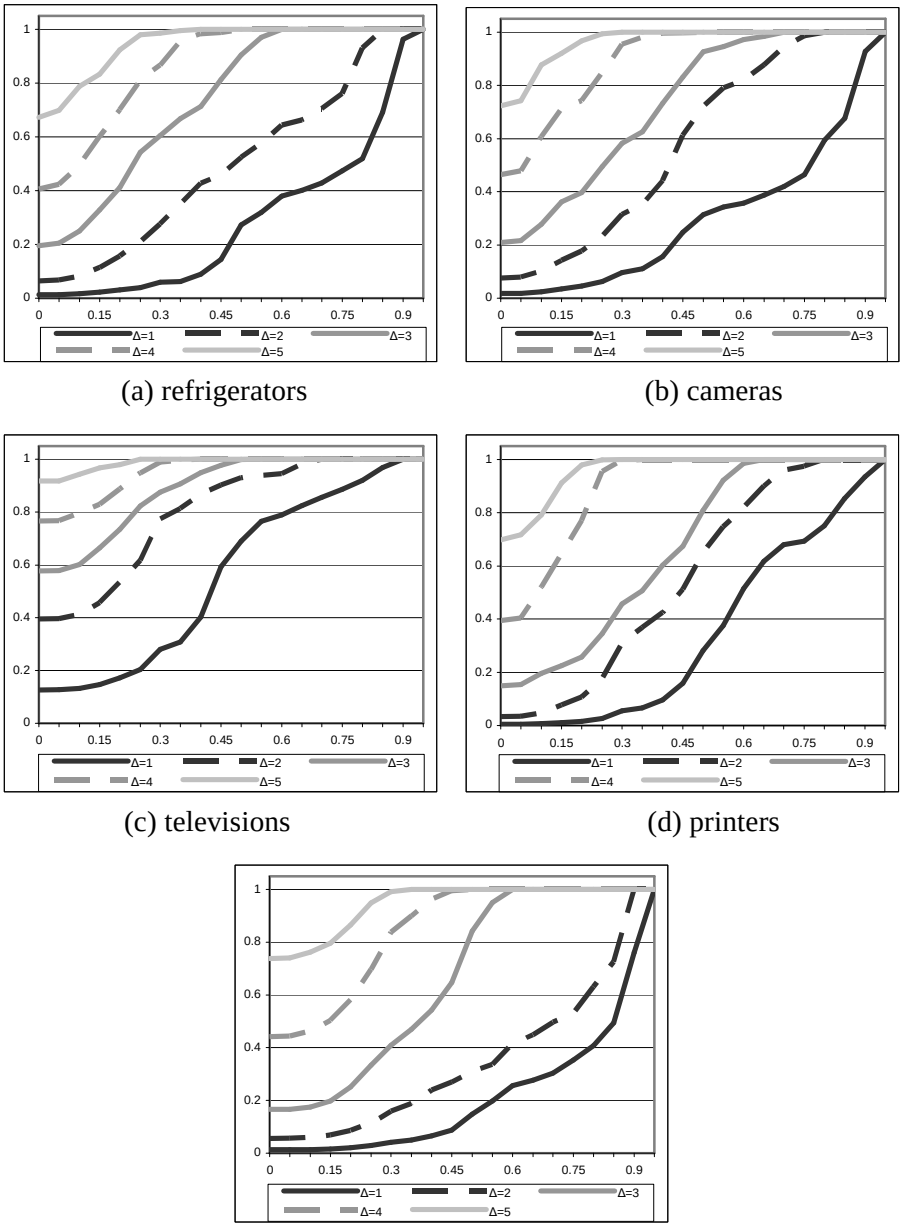
vant documents retrieved by a system as a result of a search query out of all the documents known to be relevant to the query. In other words, the recall was computed by dividing the cardinality of the UM set retrieved using UNSO-based retrieval by the cardinality of the set retrieved by the exhaustive retrieval. As R_e is the real set of UMs with the required similarity, this metrics is denoted as the recall of the retrieval.

$$recall = \frac{|R_u|}{|R_e|}$$

Note that UNSO-based retrieval scans only the UMs stored in close vicinity to the target UM. Thus, the set R_u retrieved by UNSO-based retrieval may be only a subset of the set of the most similar UMs R_e retrieved by the traditional exhaustive retrieval. This can be explained by the observation that in UNSO-based retrieval some UMs may be missed if they differ from the target UM in a number of coordinates greater than Δ, i.e., when mapped into the MLH, the UMs were stored in the nodes having more that Δ different coordinates from the node of the target UM. As a result, the recall values are lower than or equal to *1*.

The experiment was repeated for a number of times equal to the number of UMs in the corpora and the overall recall values were computed by averaging the recall values obtained for every target UM. Figure 25 shows the values of the recall as a function of the similarity threshold β (the higher it is, the better is the required similarity) and the maximal allowed numbers of modified coordinates Δ for different corpora. The horizontal axis represents the values of the similarity threshold β, while the vertical axis represents the level of recall. For each corpus we plot five graphs, for $\Delta=1, 2, 3, 4,$ and 5.

It can be seen that the recall of the retrieval is correlated with the number of allowed modified coordinates Δ. For low values of Δ, the number of UMs considered by UNSO-based retrieval (i.e., UMs differing in at most Δ coordinates from the target UM) is low, and since $|R_e|$ is constant, the recall values are low. Increasing Δ expands the search space, the search compares the target UM with more UMs, R_u grows, and consequently the results of UNSO-based approximated retrieval get closer to the results of the exhaustive retrieval. As a result, the recall increases for higher values of Δ and it converges faster to the maximal value of *1*, i.e., the optimal recall is obtained for lower values of β. This observation is true for all the corpora.



(a) refrigerators (b) cameras

(c) televisions (d) printers

(e) mobile phones

Fig. 25. Recall of the Retrieval vs. the Similarity Threshold β for Various Values of Δ.

For a low similarity threshold β and especially for low values of Δ, the recall values in most corpora are low. This is explained by the observation that the UMs, whose similarity with the target UM is low, may be located not only in close vicinity of the target UM, but also in farther locations and may have many different coordinates. As a result, the approximated UNSO-based retrieval may miss them, whereas the traditional exhaustive retrieval discovers them. Thus, for low values of β the recall is relatively low and UNSO-based retrieval is not appropriate. However, this issue is of a minor importance in real-life systems, where the retrieval is aimed at finding only the UMs with a high similarity to the target UM.

When β increases, which is the most important situation for similarity-based retrieval, the UMs with a low similarity are filtered out by the exhaustive retrieval, and since the set of UMs considered by UNSO-based retrieval does not change (it is always a set of UMs having up to Δ coordinates different from the target UM), the overall recall increases. Thus, for high values of β, both the retrieved sets are very close and the recall converges to *1*. This means that for high β, the set of UMs retrieved by UNSO-based approach is almost identical to the set of UMs retrieved by the traditional exhaustive approach, whereas the required computational effort is smaller.

Note that the precision of the retrieval is always *1*. Precision is computed as the proportion of the relevant documents retrieved by a system as a result of a search query from known corpora out of the all documents retrieved by a system as a result of a search query. Since all the UMs retrieved by UNSO-based approximated retrieval are above the given similarity threshold β, the precision of the retrieval always reaches its maximal value of *1*.

Although the recall increases with β and converges to *1* faster with the increase in Δ, there is some difference between the behaviors of recall in different corpora. The major difference can be observed at the low levels of the threshold β, as for high levels of β the recall is close to *1* in most corpora. For the low values of β, the proposed approximated retrieval considers only a small portion of potentially similar UMs. Hence, many potentially similar UMs are missed by the retrieval and the recall is low. In this order of magnitude for β, the recall values vary significantly across different corpora. To compare them and draw meaningful conclusions, we plot in Figure 26 the recall curves for different corpora as a function of β for a fixed value of $\Delta=3$.

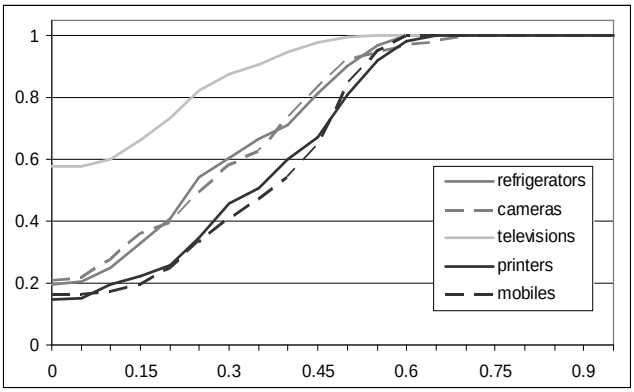


Fig. 26. Recall in the Corpora vs. Similarity Threshold β for $\Delta=3$

As can be seen from Figure 26, the behavior of recall in all the corpora is similar, as the recall values increase with β. Comparison of the recall curves for different corpora shows that there are *3* groups of corpora. The first group includes the corpus of televisions, which demonstrates the best recall and outperforms the other corpora. The second group includes the corpora of refrigerators and cameras, which demonstrate moderate recall values. Finally, the corpora of printers and mobile phones belong to the third group and demonstrate the lowest recall values. To understand the differences in the recall behavior, we will present later an elaborate analysis of the statistics of features and values within the corpora, which were shown in the Tables 10 and 11.

7.4.2.2 Precision Experiments

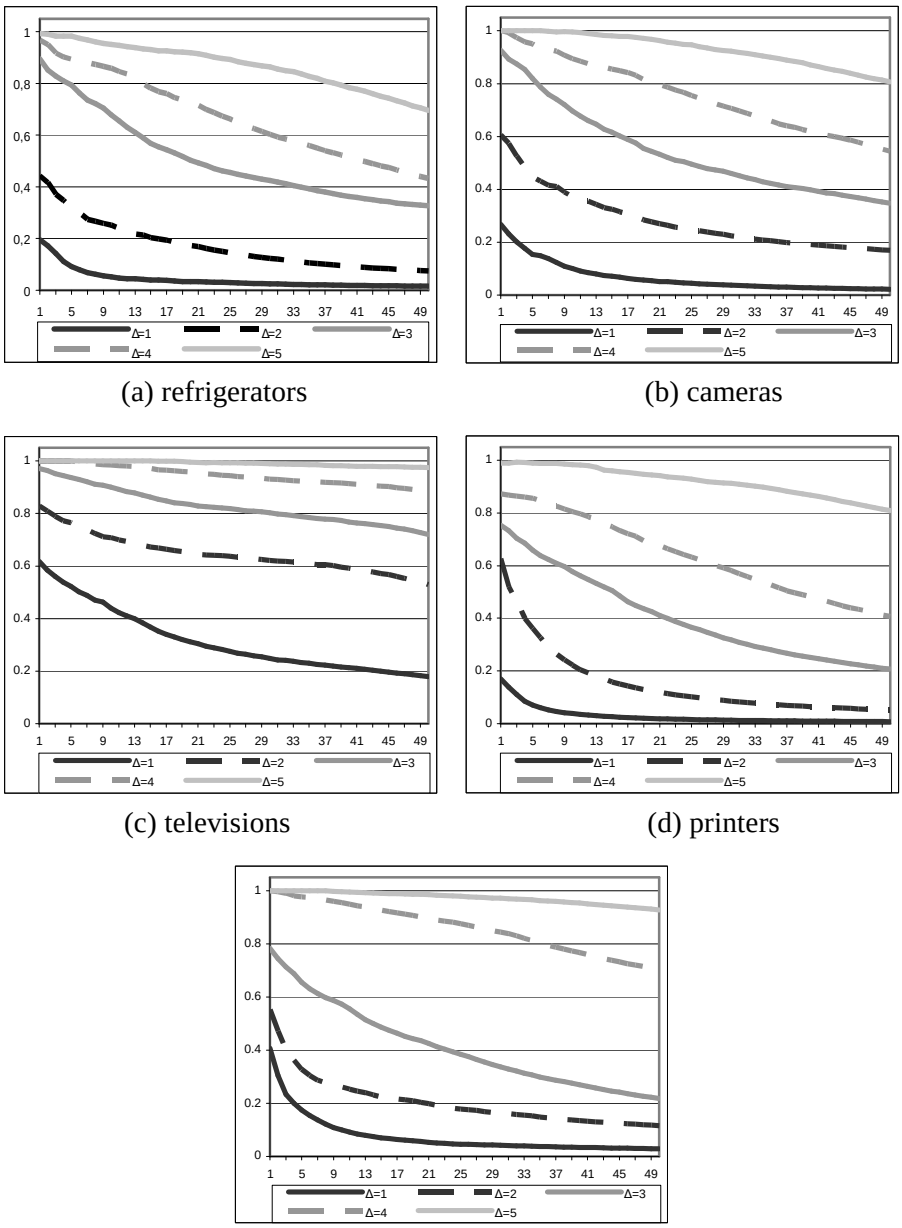
In addition to the recall experiments, we also conducted experiments evaluating the precision of UNSO-based retrieval[15]. Since measuring the precision implies limiting the set of retrieved UMs, this experiment was conducted using the *K* most similar UMs variant of the retrieval. Let us denote by R_e the set of *K* UMs retrieved by the exhaustive retrieval, and by R_u the set of *K* UMs retrieved using UNSO. Given the local nature of UNSO-based retrieval, the two sets are not identical. As a result, UNSO may retrieve *K* most similar UMs that include only part of the real most similar *K* UMs found by the exhaustive search. Thus, the precision metric in this case is defined as the relative part of the real *K* most similar UMs retrieved by UNSO-based retrieval:

$$precision = \frac{|R_u \cap R_e|}{|R_u|} = \frac{|R_u \cap R_e|}{K}$$

In each execution of the experiment, a single target UM was chosen. The overall precision values were computed as an average of the precision values for each target UM, where the number of executions was equal to the size of the corpus. In this experiment we conducted *K* most similar UMs retrieval, i.e., we retrieved *K* UMs, whose similarity to the target UMs was highest and $|R_u|=K$. Figure 27 shows the values of the precision for the above corpora as a function of *K*, for different numbers of the maximal allowed modified coordinates Δ. The horizontal axis stands for *K*, the number of the most similar UMs to retrieve, while the vertical stands for the precision values. For each corpus we plot five graphs for $\Delta=1, 2, 3, 4$, and 5.

[15] In fact, this metric is not a classical precision, but Precision@K [Salton and McGill 1983], as the retrieval is limited to *K* most similar UMs. For the sake of clarity, this metric is referred to as precision.

Similarly to the threshold retrieval, the precision of the retrieval is correlated with the number of modified coordinates *Δ*. For low *Δ*, the number of UMs considered by UNSO-based retrieval is low (note the percentage columns for low *Δ* in Figure 24). In many cases this number may be lower than the number of UMs to retrieve and high values of precision cannot be achieved, regardless of the similarity of UMs. Increasing the number of the number of modified coordinates *Δ* expands the search space and increases the number of UMs that are considered. Thus, for higher values of *Δ* the precision is higher and close to the optimal precision value of *1* for lower values of *K* (i.e., retrieval of highly similar UMs). This observation is true for all the corpora.



(a) refrigerators (b) cameras

(c) televisions (d) printers

(e) mobile phones

Fig. 27. Precision of the Retrieval vs. the Number of Retrieved UMs *K* for Various *Δ*.

The behavior of the precision for all the corpora is similar. For low values of *K*, i.e., retrieval of highly similar UMs, the precision is high and the sets of UNSO-based retrieved UMs and the real set of *K* most-similar UMs found by the exhaustive retrieval are similar. The precision decreases for higher values of *K* and lower threshold of similarity. This is explained by the observation that the set of UMs considered by UNSO-based retrieval for a certain value of Δ is fixed, and the similarity of the UMs in this set may not be sufficient for a large value of *K*. Thus, increasing the number of most similar UMs to retrieve causes the algorithm to retrieve less similar UMs, and the precision of the retrieval is hampered. It should be noted that practical systems typically retrieve a small number of highly similar UMs. Hence, the accuracy of the proposed retrieval should be evaluated at these values of *K*, where the precision of the retrieval is high.

To compare the precision of UNSO-based retrieval in different corpora, we plot in Figure 28 all the available precision curves as a function of a number of retrieved UMs *K* for a fixed value of $\Delta=3$. The behavior of the curves is similar, as the precision values decrease with *K*. Similarly to the recall, also the precision allows partitioning the corpora into *3* groups. The refrigerators corpus belongs to the group, where the precision is highest. It outperforms the group of refrigerators and cameras corpora, whose precision is similar. The worst precision is demonstrated by the group of printers and mobile phones corpora, whose precision values are almost identical. In the following subsection we analyze various statistical properties of the UMs in the corpora for the purpose of understanding the differences between the corpora in terms of precision and recall.

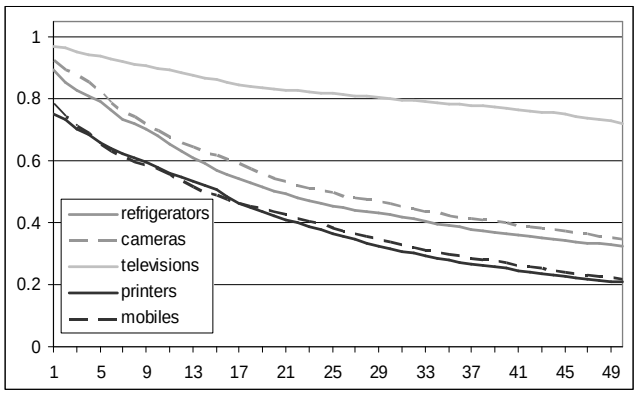


Fig. 28. Precision in Various Corpora vs. Number of UMs to Retrieve *K* for $\Delta=3$

7.4.2.3 Analysis

First, we would like to stress the reason for recall and precision not reaching their optimal value of *1* for low values of β and high values of *K*, i.e., retrieval of not only highly similar UMs. Given a corpus *C* and the target UM, the exhaustive retrieval algorithm compares it with all the UMs in the corpus, i.e., with $|C|$ UMs. Alternatively, the approximated UNSO-based retrieval compares it with the UMs having up to Δ different coordinates only. As a result, the number of UMs considered by UNSO-based retrieval, denoted by $|C_U|$ is smaller than $|C|$. The lower is the ratio $|C_U|/|C|$, the higher is the probability that recall and precision will not reach their optimal values of *1*.

Practically, this may happen for two reasons: (1) not enough UMs are considered, i.e., $|C_U|$ is smaller than the number of UMs with the similarity above the required threshold β, or $|C_U|$ is smaller than *K*, the number of the most similar UMs to be retrieved, and (2) the considered UMs are not sufficiently similar, i.e., among the considered UMs there are not enough UMs with the required similarity, or the considered UMs are not similar to the extent that will allow them to be included in the real set of *K* most similar UMs. Thus, low retrieval performance is expected when the number of UMs that should be retrieved is high, i.e., either for a high value of *K* or for a low similarity threshold β. On the contrary, the proposed UNSO-based retrieval algorithm is expected to be accurate and highly effective for retrieving a small number of highly similar UMs, since the similar UMs are typically located in the locations having a small number of different coordinates Δ, and they will be found and considered by the proposed retrieval algorithm.

Although both the values of recall and precision are high for retrieving only a small set of highly similar UMs, a certain difference between the behaviors of the proposed algorithm in different corpora can be observed. To understand it better, we analyzed the correlation between various statistical properties of the data in the corpora and the performance of the approximated retrieval algorithm. We identified two factors that may affect the retrieval capabilities, both in terms of recall and precision. These factors reflect the distribution of the unspecified pairs and specific values in the corpora:

- Average number of $feature_i$:$value_i$ pairs in an UM, further denoted by $pairs_{UM}$. This number is computed by dividing the total number of pairs that appear in the UMs of a given corpus by the number of UMs in this corpus. Since the approximated retrieval of UMs is conducted through modifying the values of up to Δ coordinates, a low value of $pairs_{UM}$ in a corpus allows the search to discover and consider more potentially similar UMs. In fact, if a corpus has a low value of $pairs_{UM}$, then two arbitrary UMs are described, on average, by a small number of different pairs. Hence, it is more likely that modifying a small number of coordinates in one UM will allow the proposed retrieval algorithm to discover the other UM. As more UMs are discovered by the retrieval, more candidates for being one of the similar UMs are considered and compared, and the retrieval capabilities are improved. Alternatively, in a corpus with a high value of $pairs_{UM}$, for a given number of maximal modified coordinates Δ, a smaller number of UMs having Δ coordinates different from the target UM is discovered, and the retrieval capabilities are hampered. As a result, retrieval capabilities of the proposed UNSO-based retrieval improve with the decrease of $pairs_{UM}$.
- Variability of values within the features in a corpus, which intuitively indicates how different two values of a given feature are expected to be. The computation of the variability should be treated separately for different types of features, as it highly depends on the similarity metric being exploited. For example, for Boolean and Free-Text features, where the similarity is computed through the exact matching check, the variability var_{FT} is correlated with the number of possible values of a feature. Conversely, in the Numeric features, where the similarity is computed by dividing the difference between two values of a feature by the maximal distance within this features, the variability var_{Num} is correlated with the variance of values of a feature. When the variability of a feature is low, retrieval with a fixed number of coordinates that are allowed to be modified considers a larger number of UMs. Since the set of UMs retrieved by the exhaustive search remains unchanged, both the precision and the recall improve. Alternatively, when the variability is high, feature values are distributed across the respective dimension, the organization of UMs is sparse, and the retrieval capabilities are hampered.

In order to show the correlation between the statistical properties of the data and the retrieval capabilities, we computed the values of $pairs_{UM}$, and variability for the Free-Text and Numeric fea-

tures[16]. var_{FT} of a corpus was computed as a weighted average of the numbers of possible values of all the features that appear in a corpus. var_{Num} was computed as a weighted average of the numeric variability, i.e., standard deviation of the corpus features divided by their maximal difference. Table 12 shows the values of $pairs_{UM}$, var_{FT} and var_{Num} in the different corpora.

Table 12. Statistical Properties of Data in the Corpora.

corpus	*refrigerators*	*cameras*	*televisions*	*printers*	*mobiles*
$\boldsymbol{pairs_{UM}}$	*4.164*	*4.000*	*3.763*	*4.692*	*4.485*
$\boldsymbol{var_{FT}}$	*8.750*	*8.250*	*6.429*	*27.500*	*24.750*
$\boldsymbol{var_{Num}}$	*0.262*	*0.262*	*0.235*	*0.292*	*0.308*

We have computed the correlation between these indicators and the recall of the proposed UNSO-based retrieval. The average recall, computed as the average of the recall for various values of β, is strongly negatively correlated with these three indicators. For example, for $\Delta=3$ the average recall has a correlation of *-0.882* with $pairs_{UM}$ (significance $p=0.024$), of *-0.765* with var_{FT} ($p=0.066$), and of *-0.944* with var_{Num} ($p=0.008$). Note that *2* out of *3* correlation values are statistically significant, as significance is $p<0.05$. It should be highlighted that similar correlations can also be observed with the recall of the proposed algorithm for other values of Δ.

It can be seen that both $pairs_{UM}$ and variability of the televisions corpora is the lowest, while the precision and the recall in this corpus are the highest. The next is a group of the refrigerators and the cameras corpora, where $pairs_{UM}$ and variability are higher and the retrieval capabilities are worse. Finally, $pairs_{UM}$ and variability for the group of the printers and mobile phones corpora are significantly higher. Accordingly, their precision and recall are the lowest. These correlations between various characteristics of the data in the corpora and the values of the recall and precision explain the differences in the retrieval capabilities of the proposed algorithm. To understand fully the differences in the performance of the proposed retrieval across different corpora, the distribution of the feature types in the corpora, shown by Table 10 should be also considered.

It also worth noting that when applying the proposed retrieval, one must tune the number of allowed modified coordinates Δ to the average number of pairs in a corpus. For example, consider the approximated K-best retrieval and assume that we are interested in a precision greater than *0.8* for retrieving $K=5$ most similar UMs. Figure 25 shows that for the televisions corpus with a low value of $pairs_{UM}$, a search with $\Delta=2$ modified coordinates will be sufficient. However, for refrig-

[16] The variability of the Boolean features was not computed here, as they have only two values: *true* and *false*.

erators and cameras, having larger values of $pairs_{UM}$, we need to search with $\Delta=3$ different coordinates. Finally, for the printers and mobile phones corpora with even higher value of $pairs_{UM}$, a deeper search with $\Delta=4$ modified features is required to obtain a similar accuracy.

7.4.3. Computational Optimization

The main goal of the UNSO-based retrieval was to decrease the number of comparisons performed during the retrieval process, while maintaining reasonable quality of results. The computational effort is reduced by exploiting the grouping mechanism of UNSO, where the target UM was compared only with the UMs with at most Δ modified coordinates, instead of with the whole set of UMs. Moreover, as the UMs are stored distributively, the comparisons are performed at the connected users, not involving any central processing. This resolves a possible computational bottleneck in the central processing and allows additional spreading of the computational effort.

This experiment was aimed at comparing the number of required comparisons for exhaustive and UNSO-based retrieval. In each execution of the experiment, a single target UM was considered, the sets of the most similar UMs were retrieved using UNSO-based retrieval, and the number of UMs compared during each retrieval process, i.e., the number of UMs considered, was computed. The experiment was repeated for the number of times equal to the number of UMs in the corpora and the overall number of comparisons was computed as an average of the numbers of comparisons for each target UM.

Table 13 shows the average number of comparisons in a single UNSO-based retrieval, denoted by *comp*, in different corpora for different values of the maximal allowed number of modified coordinates $\Delta=1$, *2*, *3*, *4*, and *5*. Note that the number of comparisons is not affected by the value of the threshold β, but only by the allowed number of modified coordinates Δ. This number should be compared to the number of comparisons in the traditional exhaustive retrieval, denoted by *exh*, which is equal to the number of UMs in the corpus minus *1* (the target UM is not compared to itself). To allow easier analysis of the results, we computed the relative number of comparisons that were conducted in UNSO-based retrieval by dividing *comp* by *exh*. It is denoted in the Table 13 by %.

Table 13. Number of Comparisons in the Approximated Retrieval.

corpus	***exh***	***Δ=1***		***Δ=2***		***Δ=3***		***Δ=4***		***Δ=5***	
		comp	*%*	*comp*	*%*	*comp*	*%*	*comp*	*%*	*comp*	*%*
refrigerators	*60*	*0.92*	*1.53*	*4*	*6.67*	*11.84*	*19.73*	*24.39*	*40.66*	*40.46*	*67.43*
cameras	*64*	*1.16*	*1.83*	*4.98*	*7.79*	*13.45*	*21.01*	*29.78*	*46.54*	*46.31*	*72.36*
televisions	*75*	*10.13*	*13.51*	*30.24*	*40.52*	*43.79*	*58.39*	*57.76*	*77.02*	*68.92*	*91.89*
printers	*93*	*0.45*	*0.48*	*3.15*	*3.39*	*13.85*	*14.89*	*36.66*	*39.42*	*64.87*	*69.76*
mobiles	*129*	*1.74*	*1.35*	*7.71*	*5.98*	*21.57*	*16.72*	*57.29*	*44.41*	*95.45*	*73.99*

The results show that in all the corpora the number of comparisons in UNSO-based approximated retrieval is lower than in the traditional exhaustive retrieval. The number of comparisons in UNSO-based retrieval increases with the allowed number of modified coordinates *Δ*. This is explained by the fact that for higher values of *Δ* the retrieval is expanded and more candidate UMs are compared. However, it can be seen that even for *Δ=5*, where the precision and the recall are very high, the number of required comparisons in UNSO-based retrieval is lower than in the traditional exhaustive retrieval.

The results also show that for any value of *Δ*, the relative number of comparisons in the refrigerators, cameras, printers and mobile phones corpora is lower than in the televisions corpora. This is explained by the smaller number of average pairs used in the description of the television UMs. These results also stress the results shown in Figure 24, where for a low number of modified coordinates *Δ*, the UMs from the televisions corpus generate a denser structure than the UMs from the other corpora.

Returning to the above sample retrieval of *K=5* most similar UMs with the precision greater than *0.8*, we note the following computational gain. In the televisions corpus, the required search with *Δ=2* modified coordinates compares the target UM with approximately *40%* of the UMs that would be compared in the exhaustive search. Similarly, search with *Δ=3* modified coordinates in the refrigerators and cameras corpora compares it with approximately *20%* of the UMs, while the search with *Δ=4* in the printers and mobile phones corpora compares approximately *40%-45%* of UMs. Thus, even for these restricting conditions the proposed UNSO-based approximated retrieval improves the computational overhead of the traditional exhaustive retrieval.

The results of this experiment show the trade-off between the retrieval capabilities of UNSO-based retrieval and the computational optimization being achieved. It can be seen that the highest recall and precision for the televisions corpora are obtained at the expense of a larger number of comparisons conducted during the retrieval process. In the other corpora, the values of the precision and the recall are lower, and also the number of comparisons. In summary, combining the results of this experiment with the results of the retrieval experiments allows us to conclude that the proposed UNSO-based retrieval, tuned to the statistical properties of UMs in the corpus (1) decreases the number of comparisons at the retrieval of the most similar UMs, while it (2) keeps reasonably good retrieval capabilities, both in terms of the precision and the recall, with respect to the original set of the most similar UMs retrieved using the traditional exhaustive retrieval.

7.5 Summary

This section presented an approach to pure decentralized P2P storage of UMs over a multi-layered hypercube (MLH) graph built using the UNSpecified Ontology (UNSO). UNSO facilitates relatively free descriptions of user modeling data, as they are described using a flexible list of $feature_i$:$value_i$ pairs, where neither the features nor the respective values are restricted by any a-priori defined ontology. The basic observation that UNSO inherently supports grouping of similar UMs was validated by the initial experiments of this section. The average similarity of UMs decreased with the number of modified coordinates. This observation, in turn, facilitated the development of an approximated algorithm for efficient retrieval of the most similar UMs. This algorithm can be schematically described as a localized search among a subset of UMs, located in a close vicinity of the target UM, as these UMs are supposed to be similar to the target UM.

Retrieval experiments, conducted over five corpora of real-life E-Commerce advertisements mimicking ephemeral content-based search UMs, showed that the approximated retrieval succeeds in retrieving the most similar UMs. The sets of UMs retrieved by the traditional exhaustive and the proposed approximated retrievals are very similar for low values of K nearest neighbors or high values of the similarity threshold β, i.e., for retrieval of highly similar UMs. In addition, the required computational effort measured by the number of comparisons is lower than in the traditional exhaustive retrieval. We would like to stress a practical observation that since real-life ap-

plications are typically aimed at retrieving a low number of highly similar UMs, the good performance demonstrated by the proposed approach for this range of K is especially important.

We also conducted an elaborate analysis of various statistical properties of the user modeling data, aimed at understanding the differences in the performance of the proposed approach. This analysis allowed us to draw conclusions regarding the specific conditions and parameters, such as β, K, and Δ, where the proposed approximated retrieval is highly beneficial, as it will succeed in conducting both accurate (in terms of precision and recall) and efficient (in terms of the number of comparisons) retrieval. We would like to stress that these parameters should be tuned according to the statistical properties and distribution of the available user modeling data stored by the system.

Chapter 8: Privacy Aspects of the Mediation

Managing UMs in recommender systems implies that personal (and possibly sensitive) information about the users is collected, stored and used. Privacy is an important challenge facing the growth and wide acceptance of various E-Commerce services in general, and recommender systems in particular [Brier 1997]. Many systems may easily violate users' privacy by misusing (e.g., selling or exposing) users' private information for their own commercial benefits. As a result, users, who are aware and concerned about such misuse, refrain from using them, to prevent any potential exposure of sensitive private information [Cranor et al. 1999].

Privacy hazards for recommender systems are aggravated by the fact that accurate recommendations require large amounts of personal data. For example, the accuracy of collaborative filtering recommendations is correlated with the number of similar users, number of ratings in their UMs, and the degree of their similarity [Sarwar et al. 2000]. Thus, the more accurate the UMs available to the system (i.e., the higher is the number of ratings stored in the UM), the more reliable the recommendations. Hence, there is a clear trade-off between the accuracy of the recommendations provided to the users and their privacy.

This trade-off is aggravated even more by introducing the mediation of UMs. On the one hand, mediation can improve the UMs available to the system and increase their accuracy; on the other hand, it inherently introduces a severe privacy breach, as the mediation implies that the UMs are to be shared and exchanged between multiple recommender systems. Hence, this poses an important challenge: developing techniques that will improve the privacy-preservation aspects of the mediation, while still allowing the systems to exchange and enrich their UMs.

Several techniques for privacy-enhanced recommendations in general and privacy-enhanced collaborative filtering in particular, were proposed in the past. For example, in [Canny 2002] the authors proposed basing privacy preservation on pure decentralized Peer-to-Peer (P2P) communication between the users. The study suggested forming communities of users, where the overall community reflects the preferences of the underlying users, thus representing the set of users as a whole and not as individual users. Alternatively, in [Polat and Du 2005] the authors suggested preserving users' privacy on a central server by adding uncertainty to the data. This was accom-

plished through using randomized data obfuscation techniques that modified the original content of the UMs. Hence, the data collector or attacker has no reliable knowledge about the true ratings of individual users. That work showed that obfuscation techniques did not considerably reduce the accuracy of the generated recommendations.

This section elaborates on the idea of combining the above two techniques, as initially discussed in [Berkovsky et al. 2005a]. It proposes enhancing the privacy of collaborative filtering through (1) substituting the commonly used centralized collaborative filtering system by a virtual P2P one, while (2) adding a degree of uncertainty to the data through modifying parts of the UMs. This introduces a pure decentralized setting, where individual users participate in the virtual P2P-based collaborative filtering system and exchange their UMs in the following way. The users separately maintain their UMs in the form of ratings vectors containing the ratings of the users on the items. Recommendations are requested by active users through exposing parts of their UMs and sending them as part of the recommendation request. Other users, who actually respond to the request, expose parts of their UMs (i.e., the ratings on the requested items), and send them to the active users, jointly with the degree of similarity between them and the active user. Note that the degree of similarity between the users was computed based on the ratings stored by the users and parts of the UM of the active user, received with the recommendation request. The active users collect the responses from the other users, select a subset of the most similar users as the neighborhood, and aggregate their ratings for the recommendation generation.

In this setting, the users are in full control of their personal sensitive information stored in their UMs. Hence, they can autonomously decide when and how to expose their UMs. In particular, the users may decide which parts of the UMs should be obfuscated before exposing them and may actually modify parts of their UMs to minimize exposure of their personal data. As a result, the proposed approach on the one hand enhances users' privacy, while on the other it still allows them to participate in collaborative filtering and support recommendation generation initiated by other users.

In the experimental part of this section, the accuracy of the proposed privacy-enhanced collaborative filtering is evaluated using three publicly available collaborative filtering datasets: Jester [Goldberg et al. 2001], MovieLens [Herlocker et al. 1999] and EachMovie [McJones 1997].

Thus, the experiments are conducted with both dense (Jester) and very sparse datasets (MovieLens and EachMovie). Initial experimental results for all the datasets demonstrate that relatively large parts of the UMs stored in the datasets can be obfuscated without significantly hampering the accuracy of the recommendations.

The experimental results raise a question regarding the importance of certain ratings for the accuracy of collaborative filtering recommendations. Although collaborative filtering is considered a solid and well-studied recommendation technique, no prior evaluations have tried to understand which ratings are important to the accuracy of the generated recommendations. This is important in the context of privacy, as the users may have different concerns about the potential exposure of their data. This implies that the quantity of the user's personal data, which is exposed to other users, or to the recommender system, must be adapted accordingly. For this, additional experiments aimed at analyzing the impact of data obfuscation on different types of users and ratings have been conducted. The results of the experiments indicate that the accuracy of collaborative filtering recommendations is mostly influenced by extreme ratings, i.e., ratings with extremely positive or negative values that are significantly different from the average rating in the dataset. Hence, these parts of the UMs are the most valuable for generating accurate recommendations, and they should be made available to the system and other users. On the other hand, very little knowledge about the users may be derived from their average ratings and, therefore, usually there is no need to expose these parts of the UMs.

This section also presents the results of an exploratory survey examining the users' attitude towards the above privacy-preserving collaborative filtering using the UMs obfuscation. This was done by correlating the usefulness of a certain kind of rating for the accuracy of the generated recommendations with the users' attitude to exposing such ratings. The results of the survey confirm our conclusion that the extreme ratings, which are more important for the recommendations generation than the moderate ratings, are also considered more sensitive by the users. In some sense, this is a negative result showing that there is no simple way to increase the accuracy of the recommendations without exposing sensitive ratings in the UM and it confirms again how difficult it is to optimize both the accuracy of the recommendations and sense of privacy of the users. Another outcome of the survey is in showing that the users' attitude to exposing their ratings improves as a result of applying data obfuscation.

The rest of the section is organized as follows. Section 8.1 discusses the privacy issues in collaborative filtering and recent works on distributed collaborative filtering. Section 8.2 presents the privacy-enhanced decentralized collaborative filtering using the UMs obfuscation. Section 8.3 presents the experimental results evaluating the proposed obfuscation approach. Section 8.4 presents the users' survey and analyzes its results, and Section 8.5 summarizes this section.

8.1 Privacy-Enhanced and Distributed Collaborative Filtering

Centralized collaborative filtering poses a severe threat to users' privacy, as personal information collected by service providers can potentially be transferred to untrusted parties. Thus, most users will not agree to divulge their private information [Cranor et al. 1999]. These concerns cause many users to refrain from the benefits of personalized services due to the privacy risks. For example, a survey showed that *90%* of people are concerned about protecting themselves from misuse of their personal information and *83%* of people are more than marginally concerned about privacy [Ackerman et al. 1999]. Hence, applying collaborative filtering without compromising the user's privacy is certainly one of the important and challenging issues in collaborative filtering research.

Various security and privacy issues in recommender systems were discussed in [Lam et al. 2006]. In particular, three main threats that may hamper proper functioning of a recommender system were mentioned:

- *Exposure* – undesired access to user's personal information by untrusted parties that are not supposed to access this information leading to user's refraining from using the system.
- *Bias* – manipulation of user's recommendations to change the items that are recommended inappropriately, i.e., to increase (push) and/or decrease (nuke) visibility of certain items in the system.
- *Sabotage* – intentionally reducing the functionality of a recommender system, such as service denial attacks or system malfunctioning.

The latter two threats can be considered as functional threats, as they may hamper the functionality of a recommender system (i.e., not allowing the system to provide a proper service to the users) and benefit some other service provider or company. Conversely, the first threat is a clear

threat to the users' privacy, since as a result of the exposure their personal sensitive information may become publicly available. Hence, this section focuses on the privacy threat only and aims at improving users' privacy in collaborative filtering recommender systems.

This issue was tackled in prior research from several perspectives. In [Polat and Du 2005], the authors proposed a method for preserving users' privacy over a centralized storage of the UMs through adding uncertainty to the data. Before transferring personal data to the server, each user first modified it using randomized data-modification techniques. Therefore, the server (and also the attacker) cannot find out the exact, but only the modified contents of the UMs. Although this method changed the user's original data represented by the UMs, experiments showed that the modified data still allowed relatively accurate recommendations to be generated. This approach enhanced users' privacy, but the users still remained dependent on a centralized storage of the UMs. This constituted a single point of failure, as the data could still be exposed through a series of malicious attacks involving multiple recommendation requests for various items managed by the system.

Storing the UMs in a decentralized manner, i.e., distributed between several locations, reduces the potential privacy breach of having all the data exposed to an attacker, as in this setting the attacker must violate security policies of all the locations, rather than of only one in a centralized setting. Conducting collaborative filtering over a distributed setting of data repositories was initially proposed in [Tveit 2001]. That work presented a Peer-to-Peer (P2P) pure decentralized architecture supporting product recommendations for mobile customers represented by software agents. The communication between the agents exploited an expensive routing mechanism based on network flooding that significantly increased the communication overhead. Following the ideas of [Tveit 2001], PocketLens project [Miller et al. 2004] discussed, implemented and experimentally compared five distributed architectures for collaborative filtering: using a central server, using three different types of P2P discovery mechanism, and using secure encryption communication algorithms. The experimental results showed that the performance of a P2P-based collaborative filtering is close to the performance of a centralized collaborative filtering.

In another technique for a distributed collaborative filtering eliminating the use of central servers [Olsson 1998], the active users create a query by sending parts of their UMs and requesting a rec-

ommendation for specific items. Other users autonomously decide whether they are willing to respond to the query and send their information to the active user. However, this approach requires transferring the UMs of both the active user and the responding users over the network, thus creating potential privacy breaches.

A scheme for privacy-preserving collaborative filtering was proposed in [Canny 2002]. According to it, individual users separately control all of their private data, while they are grouped into communities of users, which represent a public aggregation of their data. This aggregation allows personalized recommendations to be computed for the members of the community or for outsiders by exposing the aggregated community data, but without exposing the data of individual users. In addition, the communication between the communities is implemented using data encryption methods. Although this approach protects users' privacy in a distributed setting, it requires a priori formation of user-communities, which may become a severe limitation in today's dynamic environments.

8.2 Distributed Collaborative Filtering with Data Obfuscation

This section elaborates on the recommendation generation over a distributed set of users possibly obfuscating their data. It should be stressed that this section adopts the pure decentralized P2P organization of users, proposed by [Canny 2002]. Hence, users autonomously keep and maintain their UMs in a pure decentralized manner. Thus, the matrix of user ratings on items, stored by centralized collaborative filtering systems, is substituted by a virtual matrix, where the rows of the matrix, i.e., the ratings vectors of the users, are stored by the users in a distributed manner.

The users are connected using one of the existing P2P communication platforms [Milojicic et al. 2002; Androutsellis-Theotokis and Spinellis 2004]. The underlying platform guarantees connectivity of the users and allows each user to contact any of the other users connected to the system. Note that such setting does not have a single point of management or failure. Figure 29 illustrates the decentralized distribution of initially centralized ratings matrix.

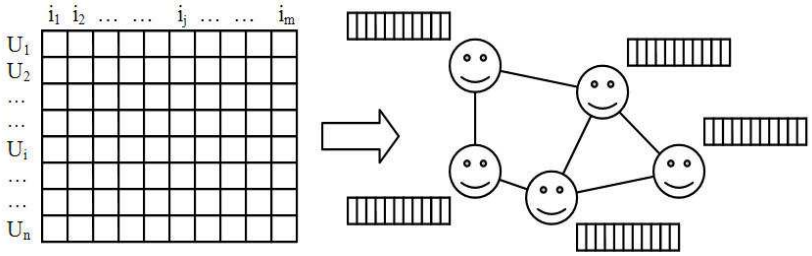


Fig. 29. Centralized vs. Decentralized Storage of the UMs

In this setting, users are the owners of their personal information. This setting introduces a pure decentralized variant of the mediation, where the users, rather than recommender systems, directly share and exchange their UMs. The users communicate with each other during the recommendation generation process and independently decide about the specific ratings and parts of their UMs that should be exposed to other users. The recommendation generation process consists of the following three stages:

- The active user initiates the process through exposing parts of the UM and broadcasting a request for a recommendation for a specific item to other users. Two parameters that should be determined for this stage are:
  1. Which parts of the UM should be exposed? To preserve the privacy of the active user better, the number of ratings that are exposed should be minimized. However, decreasing the number of ratings may hamper the similarity computation, as it will rely on a smaller number of ratings, and, therefore, hamper the accuracy of the generated recommendations. One possible solution to this tradeoff may be basing the similarity computation on a predefined subset of items, e.g., exposing only the ratings on items that are similar to the item, where the recommendation is requested [Sarwar et al. 2002].
  2. To which users should the request be sent? Theoretically, the request should be sent to all the available users, since any connected user in the network can potentially be one of the nearest neighbors of the active user. Practically, this may lead to heavy communication overheads and requires restricting the set of the users to whom the request is sent, e.g., to the a set of users similar to the active user (where the similarity of users was computed offline during a preprocessing stage), or to a set of trusted users with high reputation values (e.g., users, whose opinions were valuable for generating past recommendations for the active user) [Massa and Avesani 2004]. Alternatively, this may be resolved by applying efficient P2P routing mechanisms described in [Milojicic et al. 2002].

- When the request is received, each user autonomously decides whether to respond to it or not. If the user decides to respond, she autonomously computes her similarity degree with the active user based on the received parts of the UM of the active user. The similarity of users basically reflects the correlation of their ratings on various items. It can be computed in several ways, whereas the most popular similarity metrics in collaborative filtering recommender systems are Cosine Similarity [Good et al. 1999; Sarwar et al. 2001] and Pearson Correlation [Pennock et al. 2000a; Sarwar et al. 2000]. After the similarity degree is computed, this value and the user's rating on the requested item are sent back to the active user. Note that in this case two parts of the UM of the responding user are being exposed: (1) the rating on the requested item, and (2) the computed similarity degree, which may allow parts of the UM of the responding user to be inferred.
- Upon collecting the responses, the active user builds a neighborhood of similar users needed for generating the recommendation. This is usually done by selecting *K* users with the highest similarity degree, or selecting all the users whose similarity degree is above a certain threshold. Finally, the active user locally generates a recommendation for the requested item by aggregating the ratings of the users in the neighborhood on this item, e.g., as a weighted average according to the neighbors' similarity degree.

To summarize the recommendation generation process, it should be stressed that this form of collaborative filtering preserves users' privacy (by minimizing the exposure of their UMs), while still allowing them to support recommendations generation initiated by other users.

8.2.1 Data Obfuscation Policies

According to the above distributed collaborative filtering process, the UMs may be exposed in two cases. The first case when the UM of the active user, which is broadcast to other users as part of the recommendation request, is exposed. In this case the exposure is inevitable, as the active user must expose substantial parts of her UM in order to allow a reliable similarity computation by the responding users. The second case is when the other users voluntarily decide to participate in the recommendation generation initiated by the active user and respond to the active user. The exposure of their UMs occurs when the rating on the requested item is sent to the active user for the purpose of using it for the recommendation generation. Although in this case the responding users expose relatively only small parts of their UMs, this constitutes a privacy breach that may

allow large parts of their UMs to be exposed through systematic malicious attacks using multiple recommendation requests.

To mitigate the privacy breaches, the data in the UMs can be partially modified, i.e., part of the ratings stored in the UMs can be substituted with fake values. This section adopts the data obfuscation idea for the purposes of enhancing privacy of the distributed decentralized collaborative filtering, and focuses on modifying the UMs of the responding users only, as modifying the UM of the active user may drastically decrease the accuracy of the similarity computation. Hence, parts of the UMs of the responding users (i.e., certain ratings in the UMs) are substituted with fake values before computing the similarity and responding to the request. Although modifying the UMs does not prevent the initiator of a malicious attack from collecting the ratings of the responding users, the ratings collected by such an attacker in this setting will not necessarily reflect the real contents of the UMs.

Several methods of modifying the data for the purposes of improving privacy preservation of users' sensitive data were discussed in [Ishitani et al. 2003]: encryption [Agrawal et al. 2004], access-control policies [Sandhu et al. 1996], data randomization [Agrawal et al. 2004], anonymization [Klosgen 1995], and *K*-anonymization [Sweeney 2002]. In this section, the term *data obfuscation* [Bakken et al. 2004] is referred to as a generalization of all the approaches that involve modifying the original data for the purposes of better preserving the data privacy.

In this section, three general policies for obfuscating the ratings in the UMs are developed and experimentally compared. These policies are aimed at substituting the original ratings stored in the UMs. Substitution of the original ratings with fake values is performed according to one of the following policies:

- *Default obfuscation(x)* – substitute the real ratings in the UM with a fixed predefined value *x*.
- *Uniform random obfuscation* – substitute the real ratings in the UM with random values chosen uniformly in the range of ratings in the dataset.
- *Bell-curved random obfuscation* – substitute the real ratings in the UM with values chosen using a bell-curve distribution reflecting the distribution of ratings in the dataset.

Clearly, different general policies have different impacts on the preservation of privacy in the UMs. For example, consider the *default* obfuscation policy, which substitutes the real ratings with a predefined fixed value, such that the new ratings in the UM are typically highly dissimilar from the original ratings. As a result, the data that may be exposed by an attacker reflect the modified ratings rather than the real ratings in the UM. This is expected to decrease significantly the probability of exposing user's private and sensitive ratings and to improve the users' privacy. Conversely, the *bell-curved* obfuscation policy substitutes the real ratings in the UMs with values that reflect the distribution of ratings in the dataset. Although in some cases the new rating may be highly dissimilar from the original ratings, overall distribution of the original and modified ratings remains identical. As a result, the probability of private and sensitive ratings not being modified is higher than using the *default* policy, the probability of exposing these ratings is also higher, and the expected users' privacy is lower[17].

Besides hypothesizing and intuitively explaining that applying the above policies may preserve the users' privacy better, this section does not measure the achieved privacy gains. Instead, it focuses on the effect of obfuscating the real ratings on the accuracy of the generated collaborative filtering recommendations. The overall goal of the research is to discover general obfuscation policies and specific obfuscation techniques (i.e., which ratings should be substituted, to what extent, which fake values should substitute the real ratings, and so forth) that facilitate a maximal preservation of users' privacy, while still allowing the generation of accurate collaborative filtering recommendations.

8.2.2 Extreme Ratings and Privacy Preservation

Prior research studies have already shown that the importance of different types of ratings for the collaborative filtering process is different. For example, in [Shardanand and Maes 1995] the authors argue that the accuracy of collaborative filtering is most crucial when predicting extreme, i.e., very high or very low ratings. Intuitively, this can be explained by the observation that achieving high recommendation accuracy for the best and worst items is most important, while poor performance on average items is acceptable. Similarly, [Pennock et al. 200b] focused on evaluating collaborative filtering recommendations of extreme ratings, i.e., ratings which are *0.5*

[17] This section presents a user study, which examines users' attitude towards the above obfuscation policies and does not measure the privacy gains. In the future, it is planned to measure quantitatively the privacy gains actually achieved by applying these policies.

above or *0.5* under the average rating in the dataset (on a scale between *0* and *5*). This is based on a similar assumption that most of the time the user is interested in a recommendation of items she might very much like, or an indication to avoid certain items that she might dislike, but not recommendation of items that she may only like to a certain extent.

Another goal of this section is determining whether the importance of extreme ratings is also higher for privacy-preserving aspects. In particular, this section is aimed at determining whether the amount of private information encapsulated in certain ratings in the UMs is higher than in other ratings. As such, it examines whether the ratings with extremely positive or extremely negative values should be treated differently from ratings with moderate values.

Hence, in this section the above obfuscation policies are applied on two groups of ratings: (1) obfuscating *overall ratings* – all the available ratings, and (2) obfuscating *extreme ratings* – extremely positive or extremely negative ratings only (the exact definition of extreme ratings will be given in the following section). Moreover, this section measures the effect of obfuscating the ratings in each group of ratings on the accuracy of collaborative filtering recommendations of two types of ratings: (1) *overall recommendations* – recommendations for all the available ratings, and (2) *extreme recommendations* – recommendations for extremely positive or extremely negative ratings.

8.3 Experimental Evaluation

This section presents an experimental examination of the impact of the obfuscation policies on the accuracy of the generated recommendations. We start with a description of the implementation and experimental settings, and then proceed to the experiments, their results and analysis. The experimental evaluation may be summarized with a *2x2* table (see Table 14): the rows represent the groups of ratings that are obfuscated and the columns represent the groups of ratings where the recommendations are generated and the effect of obfuscation on the accuracy of the recommendations is measured. The contents of Table 14 are the numbers of the subsections, where the relevant experiments are presented and discussed.

Table 14. Data Obfuscation Experiments

		effect of obfuscation on ratings	
		extreme	**overall**
obfuscated ratings	**extreme**	*4.4*	*4.5*
	overall	*4.3*	*4.2*

8.3.1 Experimental Settings

For the experimental evaluation, a pure decentralized environment was simulated by a multi-threaded implementation. Each user was represented by a thread and recommendations were generated in the following manner. The thread of the active user initiated the recommendation generation process and broadcast the request to the other users. For the sake of simplicity, the request contained all the available ratings of the active user and was sent to all the available users, i.e., to all the threads. Upon receiving the request, each thread locally computed the similarity degree with the active user using the Cosine Similarity metric [Good et al. 1999; Sarwar et al. 2001], and returned the similarity degree jointly with the rating for the requested item, to the active user thread. Finally, the active user thread computed the recommendations as a weighted average of the ratings of $K=10$ most similar users. Hence, the recommendation generation process was performed similarly to a centralized collaborative filtering, except for the similarity computation stage, which was done separately by each user.

To provide solid empirical evidence, the experiments were conducted using three widely-used collaborative filtering datasets: Jester [Goldberg et al. 2001], MovieLens [Herlocker et al. 1999] and EachMovie [McJones 1997]. Table 15 summarizes various statistical parameters of the data in datasets: number of users and items in the dataset, range of ratings, total number of available ratings, average number of items rated by each user, density of the dataset (i.e., the percentage of items with available ratings), average and variance of the ratings, and the MAE of non-personalized recommendations. Since non-personalized recommendations are computed by averaging the available ratings on the required item, their MAE can be computed offline.

Table 15. Properties of the Original Datasets

dataset	**users**	**items**	**range**	**ratings**	**av.rated**	**density**	**average**	**var.**	**MAE_{np}**
Jester	*48483*	*100*	*-10-10*	*3519449*	*72.59*	*0.7259*	*0.817*	*4.400*	*0.220*
ML	*6040*	*3952*	*1-5*	*1000209*	*165.60*	*0.0419*	*3.580*	*0.935*	*0.234*
EM	*74424*	*1649*	*0-1*	*2811718*	*37.78*	*0.0229*	*0.607*	*0.223*	*0.223*

To examine the effect of obfuscating the extreme ratings on the recommendations for extreme ratings, smaller datasets containing a higher percentage of extreme ratings were extracted from the original datasets. To do this, we first defined extreme users as "users more than *33%* of whose ratings in their UMs are more than *50%* farther from the average of their ratings than their variance". For example, if the average rating of a user is *0.6* (on a scale between *0* and *1*), and the variance is *0.2*, then the ratings under *0.3* or above *0.9* are considered as extreme ratings. If the number of ratings in the UM is *200* and more than *66* ratings are extreme, then the user is considered an extreme user. The UMs of all the extreme users, which were found in the original datasets, were extracted to the extreme datasets. Although the selected thresholds of *33%* and *50%* are arbitrary (and may be a basis for future experiments), they leave large enough datasets with a higher percentage of extreme ratings. Table 16 summarizes the characteristics of the extreme datasets (columns are similar to Table 15).

Table 16. Properties of the Extreme Datasets

dataset	users	items	range	ratings	av.rated	density	average	var.	MAE_{np}
Jester	*13946*	*100*	*-10 to 10*	*1007700*	*72.26*	*0.7226*	*0.286*	*6.111*	*0.306*
ML	*1218*	*3952*	*1 to 5*	*175400*	*144.01*	*0.0364*	*3.224*	*1.166*	*0.291*
EM	*12317*	*1649*	*0 to 1*	*491964*	*39.94*	*0.0242*	*0.516*	*0.379*	*0.379*

To validate the assumption regarding the percentage of moderate ratings in the original datasets and the percentage of extreme values in the extreme datasets, the distributions of the ratings over their values were computed. Figure 30 shows the distributions of all three datasets. The horizontal axis denotes the values of the ratings in the datasets, and the vertical denotes the percentage of such ratings. Note that for each dataset, two distributions are shown. The left (light grey) bars show the distribution of ratings in the overall datasets, and the right (dark grey) – in the extreme datasets. As can be seen from the chart, overall datasets demonstrate bell-curve distribution of the ratings, while in the extreme datasets the bell-curve is inversed.

In the implemented setting, the above obfuscation of rating in the UMs was applied. Hence, every user could autonomously decide (1) whether to substitute the ratings stored in her UM, (2) how many ratings, or what percentage of ratings should be substituted (referred to in the rest of the section as the *obfuscation rate*), and (3) which ratings should be substituted. In the experiments, the above three general obfuscation policies were instantiated by five specific policies:

- *Positive* – substitute the real rating by the highest positive rating in the dataset (i.e., *10* for Jester and *5* for MovieLens and EachMovie).

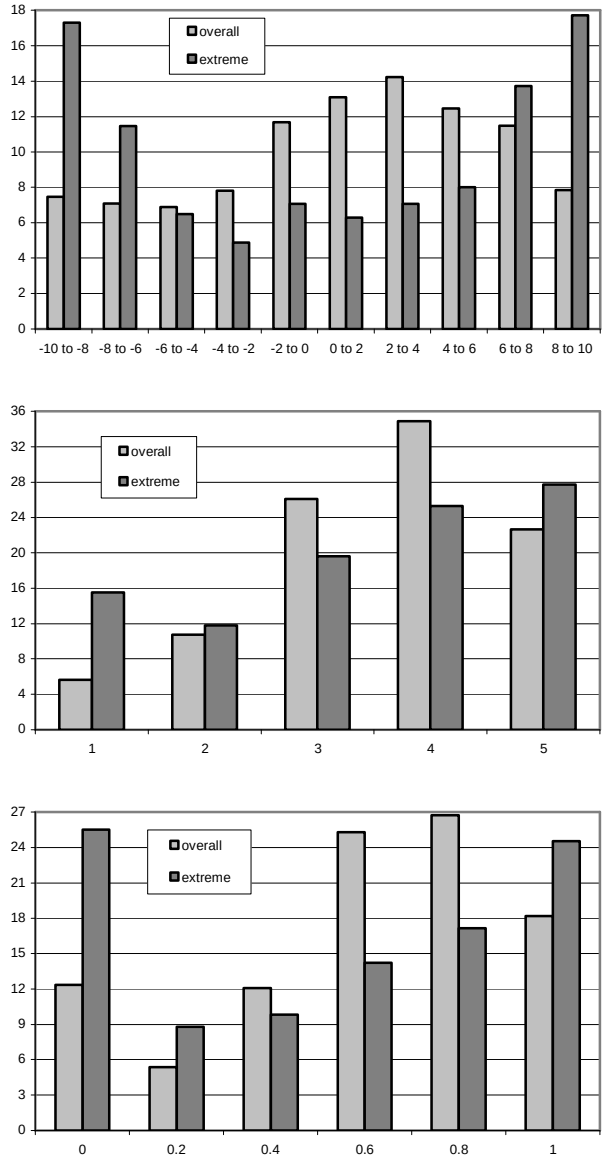


Fig. 30. Distribution of Ratings in the Datasets: Jester (top), MovieLens (middle) and EachMovie (bottom)

- *Negative* – substitute the real rating by the lowest negative rating in the dataset (i.e., *-10* for Jester, *1* for MovieLens, and *0* for EachMovie).
- *Neutral* – substitute the real rating by the neutral rating in the dataset, i.e., an average between the maximal and minimal possible ratings (i.e., *0* for Jester, *3* for MovieLens, and *0.5* for EachMovie).
- *Random* – substitute the real rating by a random value in the range of ratings in the dataset (i.e., from *-10* to *10* for Jester, *1* to *5* for MovieLens, and *0* to *5* for EachMovie).

- *Distribution* – substitute the real rating by a value reflecting the distribution (i.e., average and variance) of ratings in the dataset, as shown in Table 15 and Table 16.

Clearly, the *positive*, *negative* and *neutral* policies are instances of the general *default* policy, since the value that will substitute the original rating is known a-priori. The *random* policy is the instance of the general *uniform* policy, as the substituted values are chosen randomly in the range of dataset ratings and the *distribution* policy is the general *bell-curved* policy, as the substituted values reflect the distribution of the real ratings in the dataset.

The accuracy of the generated recommendations was measured using the Mean Average Error (MAE) metric [Herlocker et al. 2004], a statistical accuracy metric widely-used in recommender systems. The values of the MAE were computed by:

$$MAE = \frac{\sum_{i=1}^{N} | p_i - r_i |}{N}$$

where N denotes the total number of the generated recommendations, p_i is the predicted value of the item i, and r_i is the real rating given by the user on the item i. Note that lower values of MAE reflect high accuracy of the recommendations and vice-versa.

8.3.2 Obfuscation in Original Datasets

The following experiment was designed to examine the impact of obfuscation policies on the accuracy of the generated recommendations. For each dataset, a fixed testing set of *10,000* ratings was selected. These ratings were excluded from the datasets, their values were predicted using the above distributed collaborative filtering, and the MAE value of the recommendations was computed. The *10,000* recommendations experiment was repeated *10* times, for gradually increasing values of the obfuscation rate, i.e., gradually increasing amount of modified data in the UMs. Hence, the obfuscation rate increased from *0* (i.e., the original UMs are unchanged) to *0.9* (i.e., *90%* of the ratings stored in the UMs are modified according to the applied policy). Figure 31 shows the MAE values as a function of the obfuscation rate. The charts refer to Jester (top), MovieLens (middle), and EachMovie (bottom) datasets. The horizontal axis denotes the obfuscation rate, whereas the vertical denotes the MAE values.

The graphs show that in all three datasets the effect of *random*, *neutral* and *distribution* policies is roughly similar, as obfuscating the UMs has a minor impact on the MAE of the generated rec-

ommendations. Although the MAE slightly increases in a roughly linear manner with the obfuscation rate, the change in the MAE values is minor (between *0.02* and *0.07*, for different datasets), and the recommendations are still accurate. This is explained by the observation that, for *random*, *neutral* and *distribution* policies, the modified values (for average users) are relatively similar to the real ratings and the obfuscation does not significantly modify the contents of the UMs. Thus, substituting the actual ratings with similar values, even for high obfuscation rates, creates only a small overall impact on the MAE computed over many users and recommendations.

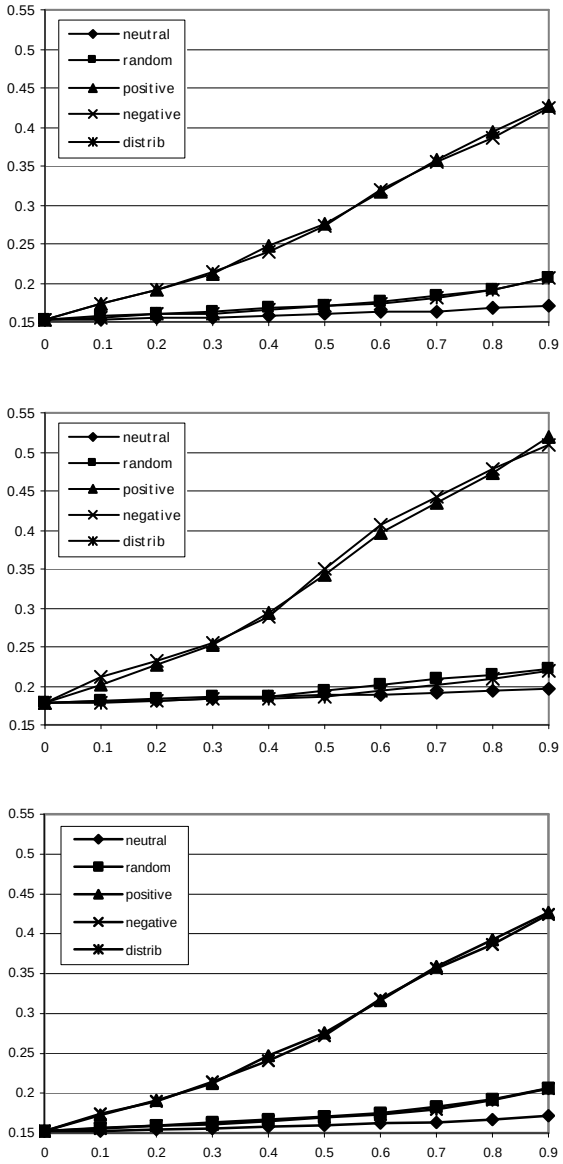


Fig. 31. MAE of the Recommendations vs. Obfuscation Rate: Jester (top), MovieLens (middle) and EachMovie (bottom)

Conversely, for *positive* and *negative* policies, the real ratings are substituted by highly dissimilar values. Thus, replacing the ratings with extremely positive or negative ratings does significantly modify the UMs for all three datasets. As a result, the generated recommendations are inaccurate and the MAE significantly increases (between *0.27* and *0.35*, for different datasets) with obfuscation rate. As can be clearly seen, the slope of the curves in *positive* and *negative* policies is significantly higher than in *random*, *neutral* and *distribution* policies. Hence, for all three datasets the effect of *positive* and *negative* policies on the accuracy of the recommendations is stronger than the effect of *random*, *neutral* and *distribution* policies.

Note that for high obfuscation rates, the MAE of the recommendations in *random*, *neutral* and *distribution* policies is close to the MAE of non-personalized recommendations (taken from Table 15). This observation is true for all three datasets. Hence, the effect of these obfuscation policies on the accuracy of the generated recommendations in the original datasets is quite understandable. The accuracy of the recommendations decreases in a linear manner from the best values, which are obtained when no data is modified, to the worst values that are close to the accuracy of non-personalized recommendations, which are obtained when most of the data are modified.

However, this raises a question regarding the conditions where this observation is true. In other words, for which users or ratings will modifying the real ratings in the UMs with moderate fake values not significantly increase the MAE of the recommendations? In particular, answering this question will allow us to draw a conclusion regarding the applicability of obfuscation on different collaborative filtering data. The data, not affected by obfuscation are not crucial for collaborative filtering recommendation generation process, and can be obfuscated without hampering the accuracy of the recommendations. Conversely, the data, where the MAE increases as a result of the obfuscation, are important for generating accurate collaborative filtering recommendations.

8.3.3 Effect of Overall Data Obfuscation on Extreme Rating Recommendations

To answer the above question, the following experiment, aimed at evaluating the impact of data obfuscation on the recommendations for different types of ratings, was conducted. In this experiment, the ratings in the datasets were partitioned into several groups, according to the values of the ratings. For example, the ratings of Jester dataset are given on a continuous scale between *-10*

to *10*. These ratings were partitioned into *10* groups, according to the ranges of ratings: from *-10* to *-8*, to *8* to *10*. Similarly, the ratings of MovieLens dataset were partitioned according to their discrete values into *5* groups: *1*, *2*, *3*, *4*, and *5*, and the ratings of EachMovie were partitioned according to their discrete values to *6* groups: *0*, 0.*2*, *0.4*, *0.6*, *0.8* and *1*.

For each group, *1,000* ratings were randomly selected and excluded from the dataset. In this experiment, the *distribution* obfuscation policy was applied on all the available ratings, and collaborative filtering recommendations were generated for the excluded ratings. This means that the excluded ratings from various groups served as the test set, whereas the remaining ratings were obfuscated and served as the training set. The MAE of the recommendations was computed for every group of ratings for gradually increasing from *0* to *0.9* values of the obfuscation rate. Figure 32 shows the MAE values for different groups of ratings. The charts refer to Jester (top), MovieLens (middle), and EachMovie (bottom) datasets. The horizontal axis denotes the groups and the ranges of ratings and the vertical denotes the MAE values. For the sake of clarity the chart shows the curves related to four obfuscation rates only: *0, 0.3, 0.6* and *0.9*. For the other obfuscation rates not shown in Figure 32, the behavior of the MAE curve is similar.

As can be clearly seen, in all three datasets the impact of the data obfuscation on different groups of ratings is different. For moderate ratings in the central part of the ratings scale, the impact of the obfuscation is minor as the MAE values roughly remain unchanged, regardless of the obfuscation. Conversely, for extreme ratings in the left and right parts of the ratings scale, the impact of the data obfuscation is stronger and the MAE steadily increases with obfuscation rate. Also, for higher obfuscation rates, a larger increase in the MAE is observed for the recommendations for extreme ratings (can be clearly seen for the extremely positive ratings of MovieLens dataset). Thus, the accuracy of collaborative filtering recommendations for extreme ratings is hampered when the ratings in the UM are obfuscated. Conversely, the accuracy of collaborative filtering recommendations for moderate ratings roughly remains unchanged when the ratings in the UM are obfuscated.

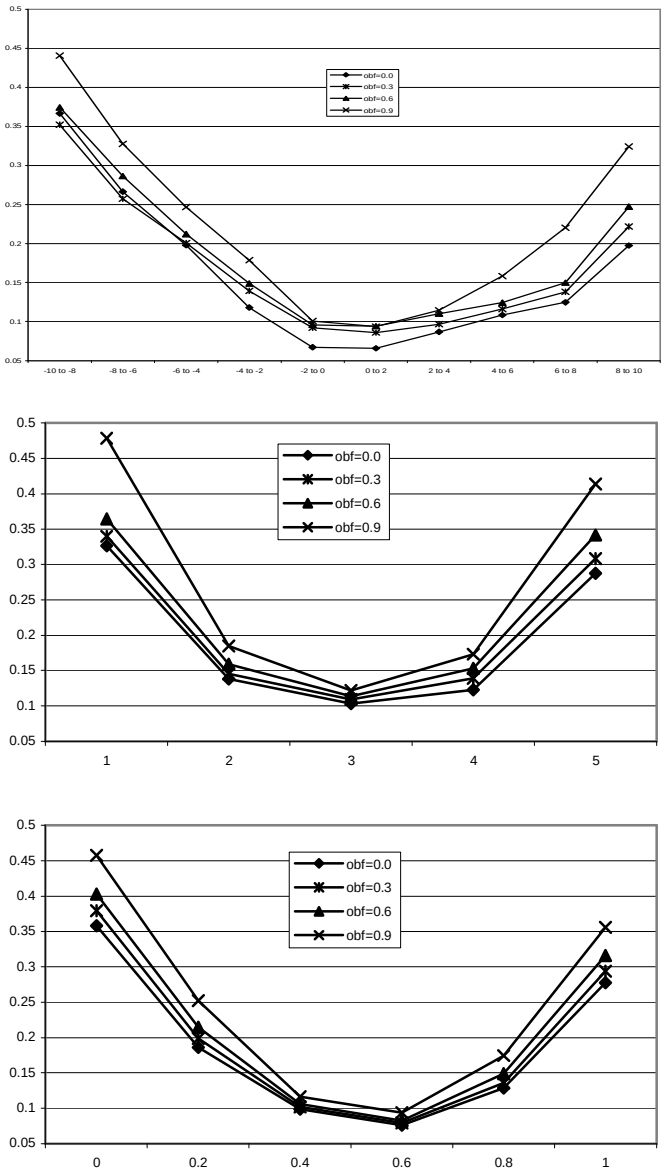


Fig. 32. MAE of the Recommendations for Various Groups of Ratings:
Jester (top), MovieLens (middle) and EachMovie (bottom) datasets

We hypothesize that this can be explained by considering the very nature of the *distribution* obfuscation policy. This policy substitutes the real ratings with fake values reflecting the average and variance of ratings in the dataset. Since the average ratings of the original datasets fall into the groups of the moderate ratings and the variance of ratings is not high (shown in Table 15), applying this policy mostly inserts moderate fake ratings into the datasets. Since collaborative filtering generates recommendations by aggregating the available ratings, the obfuscation has a minor effect on the recommendations for moderate ratings, as the inserted ratings are also moderate. However, it has a stronger effect on the recommendations for extreme ratings, as some of the

existing extreme ratings are substituted with fake moderate ratings and the accuracy of the recommendations is hampered.

8.3.4 Obfuscation in Extreme Datasets

To validate our hypothesis regarding the stronger effect of obfuscating the extreme ratings, the following experiment examined the impact of the obfuscation policies on the accuracy of the recommendations for extreme ratings only. For this, the extreme datasets (discussed in Section 8.3.1 and described in Table 16) were extracted from the original datasets. Then, the obfuscation experiment, similar to the experiment described in Section 8.3.2, was conducted. For each extreme dataset, a fixed testing set of *10,000* ratings was selected, these ratings were excluded from the dataset, their values were predicted and the MAE of the recommendations was computed. Also this experiment was repeated *10* times, for gradually increasing from *0* to *0.9* values of the obfuscation rate. Figure 33 shows the MAE values as a function of the obfuscation rate. The charts refer to Jester (top), MovieLens (middle), and EachMovie (bottom) datasets. The horizontal axis denotes the values of the obfuscation rate, whereas the vertical denotes the MAE.

The experimental results clearly show that the MAE increases with the obfuscation rate. Similarly to the overall obfuscation experiment, for *random*, *neutral* and *distribution* obfuscation policies, the change in the MAE values is linear. The minimal MAE values are observed when no obfuscation is applied, and it increases to the MAE of non-personalized recommendations (the change is between *0.07* and *0.12*, for various datasets). However, for *positive* and *negative* policies, the effect of data obfuscation is stronger than for *random*, *neutral* and *distribution* policies, and the change in the MAE values is significantly higher (between *0.14* and *0.17*). Nevertheless, for *positive* and *negative* policies, the change in the MAE for the extreme datasets is lower than for the overall obfuscation experiment (between *0.27* and *0.35*). This is explained by the observation that most of the ratings in the extreme dataset are originally extreme. Hence, substituting such values with extreme values will not significantly modify the data in many cases and the MAE values will be lower than in the overall experiment.

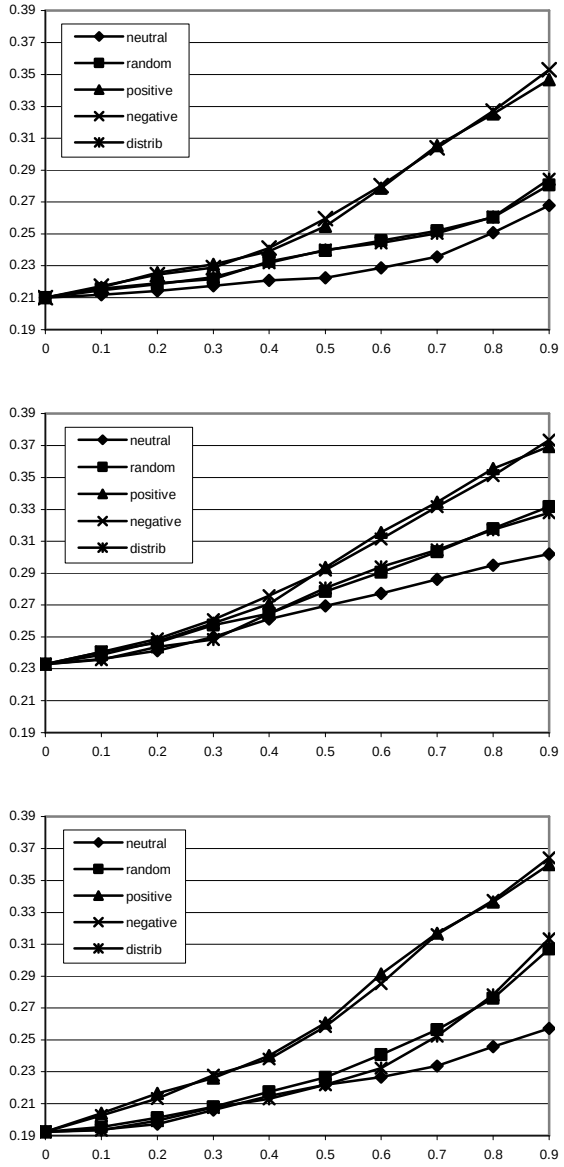


Fig. 33. MAE of the Rrecommendations vs. Obfuscation Rate: Jester (top), MovieLens (middle) and EachMovie (bottom) extreme datasets

In summary, comparison between the obfuscation of extreme and original datasets shows that for the extreme ratings, the MAE values and the slope of the MAE increase are significantly higher. This allows us to conclude that extreme ratings in the UMs are important for the personalized collaborative filtering recommendation generation. Thus, the ratings with moderate values can be obfuscated by the users without hampering the accuracy of the recommendations, whereas the extreme ratings should not be obfuscated, as they are important for the generation of accurate collaborative filtering recommendations.

8.3.5 Effect of Extreme Data Obfuscation on Overall Rating Recommendations

To validate this observation, the following experiment was aimed at evaluating the impact of *localized* data obfuscation (i.e., obfuscation of ratings with certain values) on the recommendations for various types of ratings. In this experiment, similarly to the experiment reported in Section 4.3, the datasets were partitioned into several groups, according to the values of the ratings. Jester dataset was partitioned to *10* groups, from *-10* to *-8*, to *8* to *10*, MovieLens dataset was partitioned to *5* groups: *1*, *2*, *3*, *4*, and *5*, and EachMovie was also partitioned to *6* groups: *0*, 0.*2*, *0.4*, *0.6*, *0.8* and *1*.

For each dataset, a fixed testing set of *10,000* ratings from all the groups of ratings was selected and these ratings were excluded from the dataset. Then, the ratings of one of the groups of ratings were obfuscated (according to the obfuscation rate), the values of the excluded ratings were predicted and the MAE of the recommendations was computed. This means that the remaining datasets with obfuscated ratings from a certain group served as the training set, whereas the excluded ratings served as the test set. This experiment was repeated *10* times, for gradually increasing from *0* to *0.9* values of the obfuscation rate. Note that in each experiment the obfuscation was applied for the ratings of a single group of ratings only, i.e., the ratings within certain range of values (or with a certain discrete value) only were substituted.

It should be stressed that the obfuscation rates in this case do not reliably express the amount of the obfuscated data. Since the number of ratings in every group of ratings is different, obfuscating a certain percentage of ratings in a group results in different number of obfuscated ratings in every group (see Table 15). Hence, we normalized the effect of obfuscating different numbers of ratings in every group by dividing the computed MAE values by the overall number of ratings in the respective group shown in Figure 30. Hence, the results actually show the contribution of every rating substituted in the respective group of ratings to the MAE of the recommendations.

Figure 34 shows the normalized MAE values for obfuscating different groups of ratings. The charts refer to Jester (top), MovieLens (middle), and EachMovie (bottom) datasets. The horizon-

tal axis denotes the groups and the ranges of ratings where the data were substituted, whereas the vertical denotes the MAE values. Also in this experiment, for the sake of clarity the chart shows the curves related to four obfuscation rates only: *0*, *0.3*, *0.6* and *0.9*. For the other obfuscation rates not shown in Figure 34, the behavior of the MAE curve is similar.

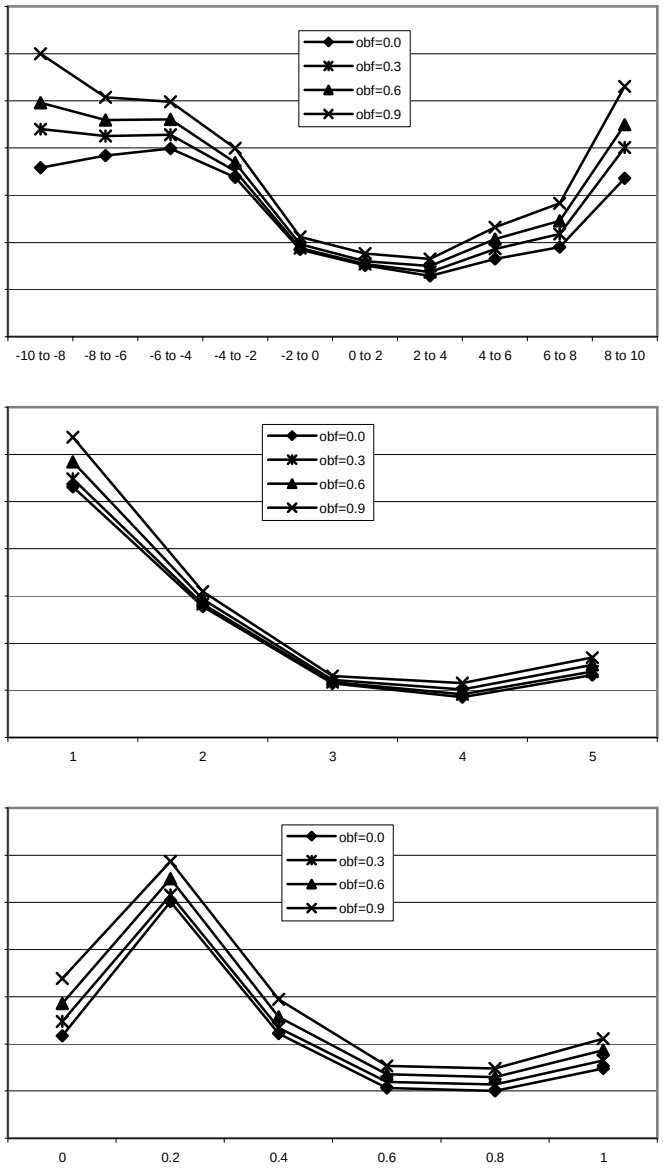


Fig. 34. MAE of the Recommendations for Obfuscation of Various Groups of Ratings in Jester (top), MovieLens (middle) and EachMovie (bottom) datasets

As can be seen from the charts, in all three datasets the effect of obfuscating different data from groups of ratings on the accuracy of the recommendations is different. When moderate ratings are obfuscated, the change of the MAE is minor, regardless of the obfuscation rate. Conversely, obfuscating extreme ratings (both extremely positive and extremely negative ratings) has a stronger impact on the MAE. The charts show that when the extreme ratings are obfuscated, the MAE

steadily increases with the obfuscation rate. This supports our hypothesis regarding the greater importance of extreme ratings for generation of accurate collaborative filtering recommendations.

This behavior is common for all three dataset. This can be seen in the clearest way in the Jester dataset, where the MAE curve is close to an inverted bell-curve. For the MovieLens, the bell-curve is biased toward extremely positive ratings. This can be explained by the fact that the distribution of ratings in MovieLens is also biased toward the positive values (as supported by Figure 30). Hence, very positive ratings in MovieLens are actually not entirely extreme, and the impact of their obfuscation on the MAE is weaker. A similar explanation is valid also for the behavior of extremely negative ratings in EachMovie. The impact of the *0* ratings on the MAE is relatively weak. This abnormal behavior is explained by the skewed distribution of ratings in EachMovie (see Figure 30). Unexpectedly, the number of *0* ratings in EachMovie is approximately *2.3* times higher than the number of *0.2* ratings. Hence, these ratings cannot be considered as really extreme ratings.

In summary, the accuracy of collaborative filtering recommendations is hampered when only the extreme ratings in the UM are obfuscated. Conversely, the accuracy remains roughly unchanged when only the moderate ratings are obfuscated. From a practical point of view, this means that the extreme ratings in the UM should be considered as the user's *representative* data, which is more important for generating accurate and personalized collaborative filtering recommendations than the moderate ratings.

8.4 Attitude of Users towards the Data Obfuscation

The above experiments show that in certain conditions data obfuscation slightly decreases the accuracy of the generated collaborative filtering recommendations, while it supposedly improves the privacy preservation of the UMs. However, the users' privacy improvement was only intuitively described and not measured quantitatively. In addition to measuring the privacy gains, there is a need to evaluate the users' attitude towards the proposed obfuscation policies and their willingness to expose their ratings, as the users may not understand how the privacy is being preserved, or not feel comfortable with applying the policies and exposing their UMs. Hence, the hypothesized privacy improvement may not correlate with the users' perception of privacy. Thus,

it is important to examine the correlation between the need for exposing users' personal data, the proposed obfuscation policies, and the user's sense of privacy.

Privacy attitudes of users towards various types of items, and not different rating values, were studied in [Cranor et al. 1999]. However, we believe that not all rating values within one class of item (e.g., movies etc.) bear the same level of importance. This is explained by the fact that users' extreme ratings express a clearer preference about an item. Thus, it is important to analyze the impact of data obfuscation methods applied on various types and values of ratings on the users' sense of privacy. Also, we aim at studying whether applying the data obfuscation policies increases users' willingness to share their ratings during the collaborative filtering process. To examine these issues, we conducted an exploratory survey of *117* users (researchers and graduate students from the user modeling and adaptive hypermedia research communities and from the Computer Science Department in the University of Haifa) to evaluate their opinions. In the rest of this subsection we present our results.

The survey questions referred to a collaborative filtering system operating numeric ratings given on a scale between *1* and *5*, where *1* means disliking an item and *5* means liking an item. The questions were formulated as statements and the users had to answer them on a discrete scale between *1* and *7*, where *1* means strongly disagreeing and *7* means strongly agreeing with the statements. To analyze the results and neutralize personal dependencies in the answers, we partitioned the answers into three categories: answers *1-2* were treated as disagree, answers *3-5* as *neutral*/*undecided*, and *6-7* as *agree*. The results of the survey are presented in Table 17 showing the average and the standard deviation of answers for each question, and in Figure 35 visually demonstrating the distributions of the answers (the questions and analysis will be presented afterwards)[18].

Table 17. Average Answers to the Survey Questions

question	**Q1**	**Q2**	**Q4**	**Q5**	**Q6**	**Q7**	**Q8**	**Q9**	**Q10**	**Q13**	**Q14**
average	*3.212*	*4.351*	*4.148*	*3.191*	*2.657*	*2.577*	*3.404*	*3.730*	*4.009*	*4.764*	*3.694*
std.dev.	*2.051*	*2.066*	*2.233*	*2.220*	*1.794*	*1.792*	*1.930*	*2.080*	*2.148*	*2.032*	*2.164*

[18] In this section, we present and analyze *11* out of *14* survey questions, which examine four main issues that we were interested in investigating. Other questions included in the survey referred to different issues and will be reported elsewhere.

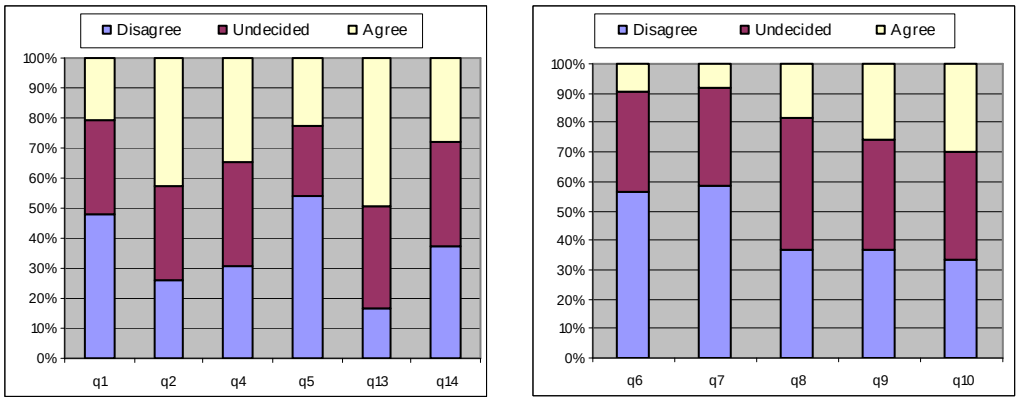


Fig. 35. Distribution of Answers to the Survey Questions: left – questions 1, 2, 4, 5, 13, and 14, right – questions 6, 7, 8, 9 and 10

The first set of questions examines whether different values of ratings within a single type of items are considered of different importance by the users. For this purpose, the following two questions are asked:

Q1: "All my ratings are equally sensitive for me, regardless of their value (*1*, *2*, *3*, *4*, or *5*)".

Q2: "My ratings with extremely positive (equal to *5*) and extremely negative (equal to *1*) values are more sensitive for me than the other ratings (*2*, *3*, or *4*)".

These questions aim to check whether ratings with values that are extremely positive or extremely negative are conceived as more sensitive by users. We defined sensitive ratings as "*ratings the users do not want to make public, such as ratings related to the political, sexual, religious, and health domains*". We hypothesize that *users consider extreme rating as more sensitive*.

We observed that answering to Q1 (Figure 35-left), *47.79%* of users disagree that all the values of their ratings are equally sensitive. Furthermore, in Q2, about *42.98%* of users strongly agree that ratings with extremely positive or extremely negative values are more sensitive than ratings with moderate values. Hence, we can conclude that users really consider their extreme ratings as more sensitive and future privacy-enhancing algorithms should treat such ratings values differently to practically enhance users' personal sense of privacy.

The second set of questions examines to what extent the users are willing to expose their ratings for the purpose of improving the accuracy of the generated recommendations. The following two questions are asked:

Q4: "I agree to make my average (equal to *3*) ratings public, if this can improve the accuracy of the recommendations provided by the system".

Q5: "I agree to make my extremely positive (equal to *5*) and extremely negative (equal to *1*) ratings public, if this can improve the accuracy of the recommendations provided by the system".

As clearly stated, Q4 examines the users' willingness to expose their moderate ratings, while Q5 examines their willingness to expose extreme ratings. We hypothesize that *although the users will generally agree to expose their ratings for the sake of accurate recommendations, they will differentiate between extreme and moderate ratings and will agree to a smaller exposure of their extreme ratings.*

The results in Figure 35-left show that users are polarized towards exposing their average ratings for the purpose of improving the accuracy of the recommendations. In particular, *34.78%* of the users do not agree to this, and *30.44%* of them agree. Hence, this contradicts the first part of our hypothesis that the users generally agree to expose their moderate ratings. Conversely, most of the users do not agree to expose their extreme ratings: only *22.61%* of users agree to expose them, while *53.91%* do not. Also the average answers shown in Table 17 validate these conclusions: the average level of agreement for exposure of moderate ratings is *4.148* and for exposure of extreme ratings is *3.191*. These results are statistically significant, *p=3.61E-09*. Intuitively, they imply that users consider extreme rating as more sensitive, i.e., as more private information, and agree to a smaller exposure of extreme ratings, which validates the second part of our hypothesis.

The third set of questions examines how the users evaluate various obfuscation policies described in the previous section. For this, we define the *positive*, *negative*, *neutral*, *random* and *distribution* obfuscation policies and then ask the users five identical questions regarding the above five policies:

Q6: "I believe that the *positive* is a good policy for preserving my privacy".

Q7: "I believe that the *negative* is a good policy for preserving my privacy".

Q8: "I believe that the *neutral* is a good policy for preserving my privacy".

Q9: "I believe that the *random* is a good policy for preserving my privacy".

Q10: "I believe that the *distribution* is a good policy for preserving my privacy".

Given the distribution of ratings in the datasets, shown in Figure 30, we hypothesize that *the positive and negative obfuscation policies are good privacy-preserving approaches*, as they substitute real ratings in the UMs with fake values that are different from the real ratings. Conversely, *the*

neutral and distribution policies are bad privacy-preserving approaches, as they substitute real ratings with fake values that are similar to the real ratings. Since the distribution of the fake values in the *random* policy is uniform, it is hypothesized to be moderate.

The results show that the users' evaluations on the policies are opposite. The average levels of agreement for *positive* and *negative* obfuscation policies are, respectively, *2.657* and *2.577*. Furthermore, most of the users (*56.48%* for *positive* and *58.56%* for *negative*) do not agree that these policies are good privacy-preserving mechanisms. The evaluations of the other three obfuscation policies are slightly better. The average level of agreement for the *neutral* policy is *3.404*, for the *random* policy it is *3.730*, and for the *distribution* policy it is *4.009*. Similarly, the percentage of users that agree that these policies are good privacy-preserving mechanisms is lower. For the *neutral* policy it is *36.70%*, for the *random* it is *36.94%*, and for the *distribution* it is *33.64%*.

Hence, the *distribution* obfuscation policy is considered by the users as the best privacy preserving policy, the second best is the *random* policy, and the third best is the *neutral* policy. Finally, *positive* and *negative* policies are considered by the users as the worst privacy preserving policies (their results are almost identical). All these results are statistically significant.

We hypothesize that this evaluation of the policies can be described by the effect of the overall evaluation of the policies and not by privacy-related evaluation only. As the *positive* and *negative* policies substitute the real ratings with highly dissimilar fake values, they hamper the accuracy of the recommendations. Hence, their overall evaluation is inferior to the overall evaluation of the *distribution*, *random*, and *neutral* policies, and this bias can be seen also at privacy-related evaluation.

Finally, the fourth set of questions aims at measuring whether the users' willingness to expose their ratings for the purpose of improving the accuracy of the recommendations changed as a result of applying the data obfuscation. The following two questions are asked:

> **Q13:** "I agree to make public my average (equal to *3*) ratings, where part of them is substituted, if this can improve the accuracy of the recommendations provided by the system".

Q14: "I agree to make public my extremely positive (equal to *5*) and extremely negative (equal to *1*) ratings, where part of them is substituted, if this can improve the accuracy of the recommendations provided by the system".

Also here we have different questions for different types of ratings. Q13 examines the users' willingness to expose obfuscated moderate ratings, while Q14 examines their willingness to expose obfuscated extreme ratings. We hypothesize that *in both cases the users will increase their willingness to expose their ratings for the sake of accurate recommendations after applying the data obfuscation*.

The results clearly validate our hypothesis and show that the users' willingness to expose their ratings of both types increased as a result of applying the data obfuscation. The average answer regarding the moderate ratings increased from *4.148* in Q4 to *4.764* in Q13 (statistically significant, *p=6.84E-05*). A similar conclusion is true also for the extreme ratings as the average answer increased from *3.191* in Q5 to *3.694* in Q14 (statistically significant, *p=9.85E-04*). Furthermore, also the distribution of the answers validates our hypothesis. Prior to applying the data obfuscation, *34.78%* of the users agreed to expose their moderate ratings and *22.61%* agreed to expose their extreme ratings. Conversely, after applying it these numbers increased to *49.09%* and *27.78%*, respectively. As already mentioned, users' willingness to expose their ratings improved as a result of the data obfuscation.

8.5 Summary

This section was motivated by the need to enhance the privacy when mediating collaborative filtering UMs in pure decentralized P2P setting. The experimental part focused on improving privacy preservation through UMs data obfuscation and its effect on the accuracy of the generated recommendations. Initial experimental evaluation presented in [Berkovsky et al. 2005a] showed that relatively large parts of the UMs can be obfuscated, without hampering the accuracy of the recommendations.

However, a deeper analysis of the results yielded an interesting behavior. When the experiments were conducted on the original datasets, the accuracy of the recommendations was barely affected. However, when only the extreme ratings were considered, the accuracy of the generated

recommendations decreased as a result of the UM obfuscations. Another experiment showed that obfuscation of extreme ratings had a stronger effect on the accuracy of the recommendations than obfuscation of moderate ratings. This allowed us to conclude that the extreme ratings are important for the accuracy of collaborative filtering recommendations, as they allow the real preferences of the users to be identified. Furthermore, this conclusion was validated by the opinions of the users, as shown by the results of the user survey. The survey demonstrated that users' willingness to expose extreme ratings is less than their willingness to expose moderate ratings.

These results introduce an interesting trade-off. On the one hand, the experiments showed that the extreme ratings are important for generation of accurate collaborative filtering recommendations. Hence, these ratings should be exposed by users to support the recommendation requests of other users, while the moderate ratings are less important. On the other hand, the survey showed the users consider their extreme ratings as more sensitive and prefer not to expose them. In combination, these two conclusions indicate that there is no simple way to optimize both the accuracy of the recommendations and the users' sense of privacy.

Part 5:

Summary and Discussion

Chapter 9: Conclusions and Future Research

9.1 Summary and Conclusions

This research was motivated by the challenges posed by the information overload problem, i.e., the overabundance of information available nowadays on the Web, which has created a need to provide Web users with personalized recommendations regarding products and information items that may interest them. The provision of personalized recommendations requires the availability of accurate User Models (UMs) that encode the needs, preferences, and interests of the users. This work presented and evaluated a general framework for mediation of UMs and user modeling data. The fundamental problem we addressed was the sparseness of user modeling data, i.e., the fact that the data may reside in several repositories and can be modeled using many heterogeneous representation techniques. We resolved this problem and showed that these multiple sources of user modeling data can be integrated, and this can improve the accuracy of the UMs, and ultimately also the quality of the recommendations provided to the users.

Initially, this work presented the user modeling data representation and warehousing in various recommendation techniques and suggested the definition of 'experience' as a fundamental unit of user modeling data. The proposed definition of experience included the representations of three primary dimensions of user modeling data (users, items, and contextual conditions of the experience), which has lead to the definition of a general user modeling data mediation approach. The mediation framework was then introduced and four particular types of mediation were derived and discussed: cross-user mediation, cross-item mediation, cross-context mediation, and cross-representation mediation.

To validate the proposed UM mediation framework, we presented the results of the empiric evaluations of two mediation approaches. The first evaluation referred to cross-technique mediation, as a specific variant of cross-representation mediation. The second evaluation referred to cross-domain mediation as a generalized form of cross-item mediation.

With respect to cross-representation mediation, we showed the mediation from collaborative filtering to content-based UMs. This mediation facilitated the generation of content-based recommendations for users, whose UMs were imported from a collaborative filtering recommender system. This experimental evaluation initially focused on feature selection for determining the data that should be taken into account by the prediction mechanism. The imported and converted UMs were then exploited for the generation of content-based recommendations, and their accuracy was compared to the accuracy of collaborative filtering recommendations, generated using the original collaborative filtering UMs. The experiments showed that for sparse collaborative filtering UMs (typical for the majority of the users), the accuracy of content-based recommendations was superior to the accuracy of collaborative filtering recommendations. Also, the experiments demonstrated the usefulness of feature selection, showing a substantial improvement in the accuracy of the generated recommendations.

Cross-domain mediation demonstrated UM mediation between collaborative filtering recommender systems from different application domains. In particular, it implemented and evaluated the effect of importing the following four types of user modeling data: (1) complete UMs, (2) lists of the nearest-neighbors candidates, (3) degrees of users' similarity, and (4) complete recommendations. The experiments showed that importing user modeling data and then generating collaborative filtering recommendations over the data from multiple systems increased recommendation accuracy with respect to collaborative filtering recommendations built over the data from a single system. The approach that imported complete UMs served as a baseline for the experimental comparisons, as its accuracy is similar to the accuracy of the traditional centralized collaborative filtering. The approaches that imported complete recommendations and that imported degrees of users' similarity outperformed the accuracy of the baseline approach. The approach that imported lists of the nearest-neighbors candidates was inferior to the baseline approach for UMs with few ratings and superior to it for UMs with many ratings.

The above techniques and evaluations practically demonstrated two important observations. First, they demonstrated that the mediation of user modeling data, i.e., the import and integration of data using external domain knowledge, is feasible. Second, they demonstrated that the mediation of user modeling data collected by other recommender systems can be beneficial. Both implementations succeeded in integrating the imported user modeling data. Their evaluations showed that

the mediation improves the accuracy (and quality) of the generated recommendations in comparison to a setting where the recommendations are based on the user modeling data collected only by the target recommender system. This allows us to conclude that the mediation of user modeling data between recommender systems is not only feasible, but also improves the performance of recommender systems and upgrades the accuracy of the recommendations provided to the users.

In addition to demonstrating the feasibility of the mediation and evaluating its contribution to the accuracy of the recommendations, this work highlighted several practical challenges that need to be addressed for a successful completion of the mediation process. Two of these challenges were studied and analyzed in depth. The first challenge referred to a decentralized storage and management of heterogeneously described user modeling data (coming from different recommender systems) in a distributed environment. The second challenge referred to privacy issues raised by the transfer of user data between the recommender systems when the mediation is conducted.

Pure decentralized storage of user modeling data was facilitated by a multi-layered hypercube graph built using the UNSO. UNSO facilitated relatively free descriptions of user modeling data using a list of *feature:value* pairs, where neither the features nor their values were restricted by any predefined ontology. The hypercube graph of UNSO supported grouping of similar UMs, which facilitated the development of an approximated algorithm for efficient retrieval of the most similar UMs. Experimental evaluations in five application domains showed that the approximated retrieval succeeds in accurately retrieving the similar UMs, while significantly decreasing the required computational effort in comparison with the traditional exhaustive retrieval. The approximated retrieval demonstrated good performance when retrieving a low number of highly similar UMs, which is typically required by real-life applications. An elaborate analysis of various statistical properties of the data allowed us to draw some conclusions regarding the specific conditions under which the approximated retrieval is beneficial.

Several privacy issues, raised by the transfer of users' personal data between recommender systems during the mediation process, were studied in the collaborative filtering recommendation approach. This part focused on improving the users' privacy through applying obfuscation policies to the data stored in the UM of a distributed collaborative filtering recommender system. Experimental results showed that the accuracy of the recommendations decreased linearly with the amount of obfuscated user modeling data and approached the accuracy of non-personalized rec-

ommendations when very large parts of the UMs were obfuscated. A further analysis showed that, when extreme ratings were obfuscated, the accuracy of the recommendations decreased faster. These results allowed us to conclude that extreme ratings are the most valuable for accurate recommendations, while average ratings can be obfuscated with only a minor impact on the accuracy of the recommendations. This was validated by the opinions of the users, who indicated in a survey that their willingness to expose the extreme ratings is lower than their willingness to expose the average ratings. In combination, these two conclusions indicate that there is no simple way to optimize both the accuracy of the recommendations and privacy of the users.

These two works referred to some the practical challenges raised by the user modeling data mediation. The proposed solutions and their evaluations practically demonstrated that these challenges can be overcame, and further support our conclusions regarding the feasibility of the mediation.

9.2 Research Contributions and Future Research

The main contribution of this work is the design and development of a general mediation framework for integrating user modeling data collected by various recommender systems in a decentralized distributed environment. This framework definition was followed by a practical mediation mechanism facilitating interoperability of specific types of recommender systems by means of the sharing and exchanging of their user modeling data, and importing and integrating data collected by other systems. As such, the mediation enriched the user modeling data available to the target recommender systems and facilitated the provision to the users of better and more accurate personalized recommendations.

This contribution can be viewed and interpreted from two research perspectives. From the perspective of user modeling research, the mediation established, modeled, and partially evaluated a novel approach to building accurate UMs through importing and integrating user modeling data collected by multiple recommender systems. From the perspective of recommender systems research, the mediation provided a basis for a novel hybrid recommendation approach, where the recommendation generations are based on multiple sources of user modeling data, rather than on only the UMs available to the target recommender system.

We believe that this work has laid the cornerstone to a new vein of research on interoperability of personalization (and, in particular, recommender) systems through sharing and exchange of user modeling data. Although this work presented the general mediation framework and evaluated some mediation scenarios as a proof of concept, several intriguing research topics were left beyond its scope. In the following list we will briefly motivate and discuss several promising topics for future research:

- **Applying machine learning and data mining techniques.** Learning mechanisms used in the implemented mediation scenarios were relatively simplistic. They used intuitive reasoning and inference mechanisms and shallow knowledge bases. For example, cross-representation mediation presented earlier in this work used a simplified assumption, which assigned equal weights to all the feature categories in content-based user modeling data. However, this assumption may hamper the accuracy of the generated UMs, as not all the features categories are of the same importance and weighting may be applied both to the feature categories and to the specific features within these categories. In the future, we plan to investigate which machine learning [Mitchell 1997] and data mining [Witten and Frank 2005] techniques can be applied for the purpose of enhancing the mediation. Particularly for the cross-representation mediation, we plan to apply and compare various machine learning techniques that will infer the weights of the feature categories and of the specific features within these categories. Applying machine learning and data mining techniques may further improve the accuracy of the generated UMs, and as a result, the accuracy of the recommendations provided to the users.
- **Exploiting user modeling ontologies.** All the implementations and experimental evaluations presented in this work exploited semantic domain knowledge for extracting various properties and characteristics of the items. However, they did not exploit any semantically-enhanced representations of the other two components of the experiences: users and context. In the future, we plan to investigate the possibility of exploiting the available user modeling ontologies [Razmerita et al. 2003; Middleton et al. 2004; Heckmann et al. 2005] in the mediation process[19]. These ontologies can be used for (1) bottom-up inference from the available user modeling data to the values of the ontology slots, and (2) the following reverse top-down inference from these inferred values of the ontology slots to the user modeling data required by the tar-

[19] In principle, also the available context ontologies [Dey and Abowd 1999; Strang et al. 2003; Wang et al. 2004] can be exploited in the mediation. In our future plans, we focus at this stage only on the user modeling ontologies.

get recommender system. The use of ontologies may improve the accuracy of the mediation process and of the generated recommendations.

- **Evaluating the mediation in real-life applications.** Although this work demonstrated practical implementations and evaluations of the mediation of user modeling data, they were all conducted on offline data and did not involve studies with real users. Our ongoing efforts focus on practical implementations and evaluations of the mediation in real-life personalization applications. For example, we are working towards implementing and evaluating the mediation from *Trip@dvice* trip planning system [Ricci et al. 2006b] to *PIL* museum visitor's guide [Kuflik et al. 2007]. In *Trip@dvice*, the UMs contain items selected and examined by the users while planning their trips (e.g., attractions, restaurants, and hotels), whereas in *PIL* they contain weighted vectors of terms reflecting the content of the preferred presentations on the museum exhibits. The mediation is performed through extracting the terms from the descriptions of *Trip@dvice* items and projecting them onto the terms representing the museum presentations in *PIL*. In the future, we plan to study more domains and applications, where the evaluation can be implemented and evaluated. Implementing mediation between real-life applications may raise new research challenges. Also, this may allow us to conduct extensive user studies, which will demonstrate users' attitude towards and appreciation of the mediation.
- **Evaluating the cross-context mediation.** None of the implementations and evaluations presented in this work referred to cross-context mediation of user modeling data. This is motivated by the fact that no context-aware datasets, including the contextual conditions of the experiences, are currently available. In the future, we plan to collect extensive context-aware datasets and evaluate cross-context mediation. In particular we are working towards achieving this within two running projects: *SharedLife* [Wahlster et al. 2006] and *Passepartout* [Aroyo et al. 2007]. *SharedLife* deals with a multi-user shopping scenario, where the users are complemented by other everyday activities, such as listening to music, cooking, and so forth. Hence, a user's feedback to a certain activity observed in certain contextual conditions can be used for the purpose of providing personalized recommendations for another activity in other contextual conditions. The *Passepartout* project deals with search, browsing and viewing activities of users with a personalized digital TV guide. The available user modeling data are collected at daily, weekly, monthly and yearly time intervals. which the mediation should be applied. Another practical challenge, where cross-context mediation can be applied, refers to provision of recommendations to individual users Hence, the provision of accurate recom-

mendations implies identification of the right granularity of the data on, or to the same users accompanied by a group of other users. Cross-context mediation may facilitate the provision of accurate context-aware recommendations and the wider acceptance of recommender systems.

- **Distributing storage of user models.** The prototype of the component for distributed P2P storage of user modeling data, which was presented in this work, was actually implemented in a centralized manner. As a result, we were not able to measure reliably the speed-up achieved by the distribution of user modeling data and the computational effort required for extracting the set of the most similar UMs. In the future, we plan to implement a real distributed P2P component for the distributed storage of user modeling data and to evaluate its performance with respect to various P2P data management metrics [Androutsellis-Theotokis and Spinellis 2004]. Moreover, the reported experiments were conducted on relatively small corpora of E-Commerce ads, and not on a real user modeling data. To provide solid experimental evidence, we plan to collect heterogeneous user modeling data (e.g., data originating from different recommender systems) and to conduct an extensive experimental evaluation of a distributed P2P storage of user modeling data.
- **Investigating privacy vs. accuracy trade-off.** This work presented in the section focusing on the privacy of the mediation introduced an interesting trade-off between the accuracy of the recommendations and privacy of users' data. On the one hand, the results of experimental evaluation showed that users' extreme ratings are important for the generation of accurate collaborative filtering recommendations, and, therefore, these ratings should be exposed by the users. On the other hand, the results of the survey showed that most of the users consider their extreme ratings as sensitive and prefer not to expose them. In combination, these two results indicate that there is no straightforward way to optimize both the accuracy of the generated recommendations and the privacy of the users' data [Smyth 2007]. In the future, we plan to investigate this challenge extensively and to develop specific obfuscation policies (i.e., which ratings should be obfuscated, to what extent, ratings of which users, and so forth), which will improve user privacy, while still allow the system to generate reasonably accurate recommendations. This will allow us to develop personalized privacy techniques, which will be adapted to the privacy concerns of individual users.

Bibliography

A.Aamodt, P.Plaza, “Case-Based Reasoning: Foundational Issues, Methodological Variants, and System Approaches”, in Artificial Intelligence Communications, vol. 7(1), pp. 39-59, 1994.

M.S.Ackerman, L.F.Cranor, J.Reagle, “Privacy in E-Commerce: Examining User Scenarios and Privacy Preferences”, in Proceedings of the ACM Conference on Electronic Commerce, Denver, CO, 1999.

E.Adar, B.Huberman, “Free Riding on Gnutella”, Technical Report, Xerox PARC, 2000.

G.Adomavicius, R.Sankaranarayanan, S.Sen, A.Tuzhilin, “Incorporating Contextual Information in Recommender Systems using a Multidimensional Approach”, in ACM Transactions on Information Systems, vol. 23(1), pp. 103-145, 2005a.

G.Adomavicius, A.Tuzhilin, “Personalization Technologies: a Process-Oriented Perspective”, in Communications of the ACM, vol. 48(10), pp. 83-90, 2005b.

G.Adomavicius, A.Tuzhilin, “Towards the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions”, in IEEE Transactions on Knowledge and Data Engineering, vol. 17(6), pp. 734-749, 2005c.

R.Agrawal, S.Ramakrishnan, “Privacy-Preserving Data Mining”, in Proceedings of the Special Interest Group on Management of Data, Dallas, TX, 2000.

R.Agrawal, J.Kiernan, R.Srikant, Y.Xu, “Order Preserving Encryption for Numeric Data”, in Proceedings of the Special Interest Group on Management of Data, Paris, France, 2004.

S.Aguzzoli, P.Avesani, P.Massa, “Collaborative Case-Based Recommender System”, in Proceedings of the European Conference on Advances in Case-Based Reasoning, Aberdeen, UK, 2002.

E.Alfonseca, P.Rodriguez, “Modelling Users' Interests and Needs for an Adaptive Online Information System”, in Proceedings of the International Conference on User Modeling, Pittsburgh, PA, 2003.

S.A.Alvarez, C. Ruiz, T. Kawato, “More Efficient Mining over Heterogeneous Data using Neural Expert Networks”, in G.I.Webb (ed.) 'Multimedia Data Mining and Knowledge Discovery', Springer, 2007

S.Androutsellis-Theotokis, D.Spinellis, “A Survey of Peer-to Peer Content Distribution Technologies”, in ACM Computing Survey, vol.36(4), pp. 335-371, 2004.

L.Aroyo, H.Schut, F.Nack, T.Schiphorst, M.Kauw-A-Tjoe, “Personalized Ambient Media Experience: move.me Case Study”, in Proceedings of the International Conference on Intelligent User Interfaces, Honolulu, HI, 2007.

D.Bakken, R.Parameswaran, D.Blough, “Data Obfuscation: Anonymity and Desensitization of Usable Data Sets”, in IEEE Security and Privacy, vol.2(6), pp. 34-41, 2004.

C.Basu, H.Hirsh, W.Cohen, “Recommendation as Classification: Using Social and Content-Based Information in Recommendation”, in Proceedings of the National Conference on Artificial Intelligence, Madison, WI, 1998.

M.Bawa, T.Condie, P.Ganesan, “LSH Forest: Self-Tuning Indexes for Similarity Search”, in Proceedings of the International Conference on World Wide Web, Chiba, Japan, 2005.

N.J.Belkin, W.B.Croft, “Information filtering and information retrieval: Two sides of the same coin?”, in Communications of the ACM, vol. 35(12), pp. 29-38, 1992.

Y.Ben-Asher, S.Berkovsky, “UNSO: Unspecified Ontologies for Peer-to-Peer E-Commerce Applications”, in Journal on Data Semantics, vol.6, pp. 115-142, 2006.

S.Berkovsky, Y.Eytani, T.Kuflik, F.Ricci, “Privacy-Enhanced Collaborative Filtering”, in Proceedings of the Workshop on Privacy-Enhanced Personalization, Edinburgh, UK, 2005a.

S.Berkovsky, T.Kuflik, F.Ricci, “P2P Case Retrieval with an Unspecified Ontology”, in Proceedings of the International Conference on Case-Based Reasoning, Chicago, 2005b.

S.Berkovsky, “Decentralized Mediation of User Models for a Better Personalization”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 2006a.

S.Berkovsky, T. Kuflik, F. Ricci, “Cross-Technique Mediation of User Models”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 2006b.

S.Berkovsky, L.Aroyo, D.Heckmann, G.J.Houben, A.Kröner, T.Kuflik, F.Ricci, “Predicting User Experiences through Cross-Context Reasoning”, in Proceedings of the Workshop on Adaptivity and User Modeling in Interactive Systems, Hildesheim, Germany, 2006c.

S.Berkovsky, T.Kuflik, F.Ricci, “Enhancing Privacy while Preserving the Accuracy of Collaborative Filtering”, in Proceedings of the Workshop on Recommender Systems, Riva del Garda, Italy, 2006d.

S.Berkovsky, A.Gorfinkel, T.Kuflik, L.Manevitz, “Case-Based to Content-Based User Model Mediation”, in Proceedings of the Workshop on Ubiquitous User Modeling, Riva del Garda, Italy, 2006e.

S.Berkovsky, D.Goldwasser, T.Kuflik, F.Ricci, “Identifying Inter-Domain Similarities through Content-Based Analysis of Hierarchical Web-Directories”, in Proceedings of the European Conference on Artificial Intelligence, Riva del Garda, Italy, August-September 2006g.

S.Berkovsky, T.Kuflik, F.Ricci, “Cross-Domain Mediation of User Models in Collaborative Filtering”, in Proceedings of the International Conference on User Modeling, Corfu, 2007a.

S.Berkovsky, T.Kuflik, F.Ricci, “Distributed Collaborative Filtering with Domain Specialization”, in Proceedings of the Recommender Systems Conference, Minneapolis, October 2007b.

S.Berkovsky, Y.Eytani, T.Kuflik, F.Ricci, “Enhancing Privacy and Preserving Accuracy of a Distributed Collaborative Filtering”, in Proceedings of the Recommender Systems Conference, Minneapolis, 2007c.

S.Berkovsky, T.Kuflik, F.Ricci, “Mediation of User Models for Enhanced Personalization in Recommender Systems”, in User Modeling and User-Adapted Interaction, vol.18 (3), pp.245-286, 2008.

S.Berkovsky, T.Kuflik, F.Ricci, “Cross-Representation Mediation of User Models”, submitted to User Modeling and User-Adapted Interaction, vol.19 (1-2), pp.35-63, 2009a.

S.Berkovsky, T.Kuflik, F.Ricci, “P2P Case Retrieval with an Unspecified Ontology”, Artificial Intelligence Review Journal, to appear, 2009b.

P.A.Bernstein, S.Melnik, “Meta Data Management”, in Proceedings of the Internation Conference on Data Engineering, Boston, MA, 2004.

A.Bernstein, E.Kaufmann, C.Buerki, M.Klein, “How Similar is it? Towards Personalized Similarity Measures in Ontologies”, in Proceedings of the Internationale Tagung Wirtschaftsinformatik, Bamberg, Germany, 2005.

D.Billsus, M.J.Pazzani, “A Hybrid User Model for News Story Classification”, in Proceedings of the International Conference on User Modeling, Banff, Canada, 1999.

D.Billsus, M.J.Pazzani, “User Modeling for Adaptive News Access”, in User Modeling and User-Adapted Interaction, vol.10(2-3), pp.147-180, 2000.

S.Bogaerts, D.Leake, “Facilitating CBR for Incompletely-Described Cases: Distance Metrics for Partial Problem Descriptions”, in Proceedings of the European Conference on Case-Based Reasoning, Madrid, Spain, 2004.

M.Bonifacio, P.Bouquet, G.Mameli, M.Nori, “Peer-Mediated Distributed Knowledge Management”, Agent-Mediated Knowledge Management, Springer, 2003.

K.Branting, D.W.Aha, “Stratified Case-Based Reasoning: Reusing Hierarchical Problem Solving Episodes”, in Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, Cananda, 1995.

G.Brajnik, C.Tasso, “A Shell for Developing Non-Monotonic User Modeling Systems”, in International Journal of Human-Computer Studies, vol. 40, pp. 31-62, 1994.

J.S.Breese, D.Heckerman, C.Kadie, “Empirical Analysis of Predictive Algorithms for Collaborative Filtering”, in Proceedings of the Conference on Uncertainty in Artificial Intelligence, Madison, WI, 1998.

D.Bridge, J.P.Kelly, “Ways of Computing Diverse Collaborative Recommendations”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 2006.

S.Brier, “How to Keep your Privacy: Battle Lines Get Clearer”, The New York Times, 13-Jan-1997.

S.Brin, L.Page, “The Anatomy of a Large Scale Hypertextual Web Search Engine”, in Computer Networks and ISDN Systems, vol. 30(1-7), pp.107-117, 1998.

P.J.Brown, J.D.Bovey, X.Chen, “Context-Aware Applications: From the Laboratory to the Marketplace”, in IEEE Personal Communications, vol. 4(5), pp. 58-64, 1997.

L.Buriano, M.Marchetti, F.Carmagnola, F.Cena, C.Gena, I.Torre, “The Role of Ontologies in Context-Aware Recommender Systems”, in Proceedings of the International Conference On Mobile Data Management, Nara, Japan, 2006.

R.Burke, K.Hammond, E.Cooper, “Knowledge-based Information Retrieval from Semi-structured Text”, in Proceedings of the AAAI Workshop on Internet-based Information Systems, Portland, OR, 1996a.

R.Burke, K.Hammond, E.Cooper, “Knowledge-based Navigation of Complex Information Spaces”, in Proceedings of the National Conference on Artificial Intelligence, Menlo Park, CA, 1996b.

R.Burke, “Knowledge-based Recommender Systems”, in A.Kent (ed.), 'Encyclopedia of Library and Information Systems', vol. 69(32), 2000.

R.Burke, “Hybrid Recommender Systems: Survey and Experiments”, in User Modeling and User-Adapted Interaction, vol. 12(4), pp.331-370, 2002.

P.Brusilovsky, A.Kobsa, W.Nejdl (eds.), “The Adaptive Web: Methods and Strategies of Web Personalization”, Springer, 2007.

A.Caglayan, M.Snorrason, J.Jacoby, J.Mazzu, R.Jones, K.Kumar, “Learn Sesame: A Learning Agent Engine”, in Applied Artificial Intelligence, vol. 11(5), pp.393-412, 1997.

D.J.Carmichael, J.Kay, B.Kummerfeld, “Consistent Modelling of Users, Devices and Sensors in a Ubiquitous Computing Environment”, in User Modeling and User-Adapted Interaction, vol. 15(3-4), pp.197–234, 2005.

J.Canny, “Collaborating Filtering with Privacy”, in Proceedings of IEEE Conference on Security and Privacy, Oakland, CA, 2002.

L.Chen L, K.Sycara, “WebMate: A Personal Agent for Browsing and Searching”, in Proceeding of the International Conference on Autonomous and Multi-Agent Systems, 1998.

P.Clark, “The Development of Net Perceptions Personalization Manager for E-Commerce”, in Proceedings of the International Conference on eXtreme Programming and Agile Processes in Software Engineering, Alghero, Italy, 2002.

I.Clarke, O.Sandberg, B.Wiley, T.Hong, “Freenet: a Distributed Anonymous Information Storage and Retrieval System”, in Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, Berkeley, CA, 2000.

M.Claypool, A.Gokhale, T.Miranda, P.Murnikov, D.Netes, M.Sartin, “Combining Content-Based and Collaborative Filters in an Online Newspaper”, in Proceedings of ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, 1999.

L.Coyle, D.Doyle, P.Cunningham, “Representing Similarity for CBR in XML”, in Proceedings of the European Conference on Advances in Case-Based Reasoning, Madrid, Spain, 2004.

L.F.Cranor, J.Reagle, M.S.Ackerman, “Beyond Concern: Understanding Net Users' Attitudes about Online Privacy”, Technical Report, AT&T Labs-Research, 1999.

W.Dai, R.Cohen, “Dynamic Personalized TV Recommendation System”, in Proceedings of International Conference on User Modeling, Workshop on Personalization in Future TV, Pittsburgh, PA, USA, 2003.

A.Das, M.Datar, A.Garg, S.Rajaram, “Google News Personalization: Scalable Online Collaborative Filtering”, in Proceedings of the International World-Wide Web Conference, Banff, Canada, 2007.

G.Desjardins, R.Godin, “Combining Relevance Feedback and Genetic Algorithms in an Internet Information Filtering Engine”, in Proceedings of Recherche d'Information Assistée par Ordinateur, Paris, France, 2000.

A.K.Dey, G.D. Abowd, “Towards a Better Understanding of Context and Context-Awareness”, in Proceedings of the International Symposium on Handheld and Ubiquitous Computting, Karlsruhe, Germany, 1999.

R.Farzan, P. Brusilovsky, “Social Navigation Support in a Course Recommendation System”, in proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Dublin, Ireland, 2006.

C.Fellbaum (ed.), “WordNet – An Electronic Lexical Database”, the MIT Press, 1998.

T.W.Finin, D.Drager, “A General User Modeling System”, in Proceedings of the Canadian Conference on Artificial Intelligence, Montreal, Canada, 1986.

T.W.Finin, “GUMS: A General User Modeling Shell”, in A.Kobsa, W.Wahlster (eds.) 'User Models in Dialog Systems', Springer, 1989.

J.Fink, A.Kobsa, “A Review and Analysis of Commercial User Modeling Servers for Personalization on the World-Wide Web”, in User Modeling and User-Adapted Interaction, vol. 10(2-3), pp.209-249, 2000.

J.Fink, A.Kobsa, “User Modeling for Personalized City Tours”, in Artificial Intelligence Review, vol. 18(1), pp. 33-74, 2002.

L.Francisco-Revilla, M.Shipman, “Managing Conflict in Multi-Model Adaptive Hypertext”, in Proceedings of the Annual Conference on Hypertext and Hypermedia, Santa Cruz, CA, 2004.

J.H.Friedman, J.H.Bentley, R.A.Finkel, “An Algorithm for Finding Best Matches in Logarithmic Expected Time”, in ACM Transactions in Mathematical Software, vol. 3(3), 1977.

P.J.Gmytrasiewicz, S.Noh, T.Kellog, “Bayesian Update of Recursive Agent Models”, in User Modeling and User-Adapted Interaction, vol. 8(1-2), pp. 49–69, 1998.

A.Goker, H.I.Myrhaug, “User Context and Personalization”, in Procedings of the European Conference on Case-Based Reasoning, Aberdeen, UK, 2002.

A.Gokhale, M.Claypool, “Thresholds for More Accurate Collaborative Filtering”, in Proceedings of the IASTED International Conference on Artificial Intelligence and Soft Computing, Honolulu, Hawaii, 1999.

D.Goldberg, D.Nichols, B.Oki, D.Terry, “Using Collaborative Filtering to Weave an Information Tapestry”, in Communications of the ACM, vol. 35(12), pp. 61-70, 1992.

K.Goldberg, T.Roeder, D.Gupta, C.Perkins, “Eigentaste: A Constant Time Collaborative Filtering Algorithm”, in Information Retrieval Journal, vol. 4(2), pp. 133-151, 2001.

M.C.Golumbic, M.Markovich, U.N.Schild, S.Tsur, “A Knowledge-Based Expert System for Student Advising”, in IEEE Transactions on Education, vol. 29(2), pp. 120-124, 1986.

M.C.Golumbic, R.Feldman, “Optimization Algorithms for Student Scheduling via Constraint Satisfiability”, in the Computer Journal, vol. 33(4), pp. 356-364, 1990.

M.C.Golumbic, M.Markovich, M.Tiomkin, “A Knowledge Representation Language for University Requirements”, in Decision Support Systems, vol. 7(1), pp. 33-45, 1991.

N.Good, J.B.Schafer, J.A.Konstan, A.Borchers, B.Sarwar, J.Herlocker, J.Riedl, “Combining Collaborative Filtering with Personal Agents for Better Recommendations”, in Proceedings of the National Conference of the American Association of Artificial Intelligence, 1999.

T.R.Gruber, “A Translation Approach to Portable Ontology Specifications”, in Knowledge Acquisition Journal, vol. 6(2), pp. 199-220, 1993.

P.Han, B.Xie, F.Yang, R.Shen, “A Scalable P2P Recommender System Based on Distributed Collaborative Filtering”, in Expert Systems with Applications Journal, vol. 27(2), pp. 203-210, 2004.

U.Hanani, B.Shapira, P.Shoval, “Information Filtering: Overview of Issues, Research and Systems”, in User Modeling and User-Adapted Interactions, vol. 11(3), pp. 203-259, 2001.

M.Harren, J.M.Hellerstein, R.Huebsch, B.T.Loo, S.Shenker, I.Stoica, “Complex Queries in DHT-based Peer-to-Peer Networks”, in Proceedings of the International Workshop on Peer-to-Peer Systems, Cambridge, MA, 2002.

C.Hayes, P.Cunningham, “Context Boosting Collaborative Recommendations”, in Proceedings of the International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, 2003.

D.Heckmann,A.Krueger, “A User Modeling Markup Language (UserML) for Ubiquitous Computing”, in Proceedings of the International Conference on User Modeling, Pittsburgh, PA, 2003.

D.Heckmann, “Ubiquitous User Modeling”, Ph.D. Thesis, Saarland University, Germany, 2005.

D.Heckmann, T.Schwartz, B.Brandherm, M.Schmitz, M.von Wiliamowitz-Moellendorff, “GUMO – the General User Model Ontology”, in Proceedings of the International Conference on User Modeling, Edinburgh, UK, 2005.

J.L.Herlocker, J.A.Konstan, A.Borchers, J.Riedl, “An Algorithmic Framework for Performing Collaborative Filtering”, in Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Berkeley, CA, 1999.

J.L.Herlocker, J.A.Konstan, L.G.Terveen, J.Riedl, “Evaluating Collaborative Filtering Recommender systems”, in ACM Transactions on Information Systems, vol. 22(1), pp. 5-53, 2004.

W.Hill, L.Stead, M.Rosenstein, G.Furnas, “Recommending and Evaluating Choices in a Virtual Community of Users”, in Proceedings of CHI Conference on Human Factors in Computing Systems, Denver, CO, 1995.

S.R.Hiltz, M.Turoff, “Structuring computer-mediated communication systems to avoid information overload”, in Communications of the ACM, vol. 28(7), pp. 680-689, 1985.

IMDb, The Internet Movie Database, http://www.imdb.com, accessed in June 2007.

L.Ishitani, V.Almeida, W.Meiru, “Masks: Bringing Anonymity and Personalization Together”, in IEEE Security and Privacy, vol. 1(3), pp.18-23, 2003.

B.J.Jansen, A.Spink, J.Bateman, T.Saracevic, “Real Life Information Retrieval: a Study of User Queries on the Web”, in SIGIR Forum, vol. 32 (1), pp. 5-17, 1998.

T.Joachims, D.Freitag, T.Mitchell, “WebWatcher: A Tour Guide for the World-Wide Web”, in Proceedings of the International Joint Conference on Artificial Intelligence, Nagoya, Japan, 1997.

P.Kalnis, W.S.Ng, B.C.Ooi, K.L.Tan, “Answering Similarity Queries in Peer-to-Peer Networks”, in Information Systems Journal, 31(1), 2006.

R.Kass, “Acquiring a Model of the User's Beliefs from a Cooperative Advisory Dialog”, Ph.D. Thesis, University of Pennsylvania, 1988.

J.Kay, “The um Toolkit for Cooperative User Modelling”, in User Modeling and User-Adapted Interaction, vol. 4(3), pp.149-196, 1995.

J.Kay, “Ontologies for Reusable and Scrutable Student Model”, in Proceedings of the Workshop on Ontologies for Intelligent Educational Systems, Le Mans, France, 1999.

J.Kay, B.Kummerfeld, P.Lauder, “Personis: a Server for User Models”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Malaga, Spain, 2002.

J.Kay, B.Kummerfeld, P.Lauder, “Managing Private User Models and Shared Personas”, in Proceedings of Workshop on User Modelling for Ubiquitous Computing, Pittsburgh, PA, 2003a.

J.Kay, A.Lum, J.Uther, “How can Users Edit and Control their Models in Ubiquitous Computing Environments?”, in Proceedings of Workshop on User Modelling for Ubiquitous Computing, Pittsburgh, PA, 2003b.

J.Kay, “Scrutable Adaptation: Because We Can and Must”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 2006.

J.K.Kim, Y.H.Cho, W.J.Kim, J.R.Kim, J.H.Suh, “A Personalized Recommendation Procedure for Internet Shopping Support”, in Electronic Commerce Research and Applications, vol. 1, pp. 301-311, 2002.

W.Klosgen, “Anonimization Techniques for Knowledge Discovery in Databases", in Proceedings of the International Conference on Knowledge and Discovery in Data Mining, Montreal, Canada, 1995.

A.Kobsa, W.Wahlster (eds.), “User Models in Dialog Systems“, Springler, 1989.

A.Kobsa, “Modeling the User's Conceptual Knowledge in BGP-MS, a User Modeling Shell System”, in Computational Intelligence 6, pp.193-208, 1990.

A.Kobsa, W.Pohl, “The BGP-MS User Modeling System”, in User Modeling and User-Adapted Interaction, vol. 4(2), pp. 59-106, 1995.

A.Kobsa, “Generic User Modeling Systems”, in User Modeling and User-Adapted Interaction, vol. 11(1-2), pp. 49-63, 2001a.

A.Kobsa, “Tailoring Privacy to Users' Needs”, in Proceedings of the International Conference on User Modeling, Sonthofen, Germany, 2001b.

A.Kobsa, J.Koenemann, W.Pohl, “Personalized Hypermedia Presentation Techniques for Improving Online Customer Relationships”, in Knowledge Engineering Review, vol. 16(2), pp. 111–155, 2001c.

A.Kobsa, “Privacy-Enhanced Web Personalization”, in P.Brusilovsky, A.Kobsa, W.Nejdl (eds.) 'The Adaptive Web: Methods and Strategies of Web Personalization', Springer, 2007.

R.Kohavi, G.H.John, “Wrappers for Feature Subset Selection”, in Artificial Intelligence Journal, vol.97(1-2), pp. 273-324, 1997.

B.Krulwich, “Lifestyle Finder: Intelligent User Profiling Using Large-Scale Demographic Data”, in Artificial Intelligence Magazine, vol. 18(2), pp. 37-45, 1997.

T.Kuflik, J.Sheidin, S.Jbara, D.Goren-Bar, P.Soffer, O.Stock, M.Zancanaro, “Supporting Small Groups in the Museum by Context-Aware Communication Services”, in Proceedings of the International Conference on Intelligent User Interfaces, Honolulu, HI, 2007.

S.K.Lam, D.Frankowski, J.Riedl, “Do you Trust your Recommendations? An Exploration of Security and Privacy Issues in Recommender Systems”, in Proceedings of the International Conference on Emerging Trends in Information and Communication Security, Freiburg, Germany, 2006.

D.B.Leake, R.Sooriamurthi, “When Two Case Bases are Better than One: Exploiting Multiple Case Bases”, in Proceedings of the International Conference on Case-Based Reasoning, Vancouver, Canada, 2001.

D.B.Leake, R.Sooriamurthi, “Dispatching Cases versus Merging Case-Bases: When MCBR Matters”, Proc. of the International Florida Artificial Intelligence Research Society Conference, St. Augustine, FL, 2003.

G.Linden, S.Hanks, N.Lesh, “Interactive Assessment of User Preference Models: The Automated Travel Assistant”, in Proceedings of the International Conference on User Modeling, Sardinia, Italy, 1997.

G.Linden, B.Smith, J.York, “Amazon.com Recommendations: Item-to-Item Collaborative Filtering”, in IEEE Internet Computing, vol. 7(1), pp. 76-80, 2003.

N.Littlestone, “Learning Quickly when Irrelevant Attributes Abound: A New Linear-Threshold Algorithm”, in Machine Learning, vol.2(4), pp.285-318, 1988.

A.Lorenz, “A Specification for Agent-Based Distributed User Modeling in Ubiquitous Computing”, in Proceedings of the Workshop on Decentralized, Agent-Based and Social Approaches to User Modeling, Edinburgh, UK, 2005.

A.Lorenz, A.Zimmermann, “Adaptation of Ubiquitous User-Models”, in Proceedings of the Workshop on Ubiquitous User Modeling, Riva del Garda, Italy, 2006.

P.Lyman, H.R.Varian, “How Much Information?” project, available online at http://www.sims.berkeley.edu/research/projects/how-much-info-2003/

P.Maes, “Agents that Reduce Work and Information Overload”, in Communication of the ACM, vol. 37 (7), pp. 31-40, 1994.

T.W.Malone, K.R.Grant, F.A.Turbak, S.A.Brobst, M.D.Cohen, “Intelligent Information Sharing Systems”, in Communications of the ACM, vol. 30(5), pp. 390-402, 1987.

N.Manouselis, D.Sampson, “A Multi-Criteria Model to Support Automatic Recommendation of e-Learning Quality Approaches”, in Proceedings of the World Conference on Educational Multimedia, Hypermedia and Telecommunications, Lugano, Switzerland, 2004.

R.L.Mantaras, D.McSherry, D.Bridge, D.Leake, B.Smyth, S.Craw, B.Faltings, M.L.Maher, M.T.Cox, K.Forbus, M.Keane, A.Aamodt, I.Watson, “Retrieval, Reuse, Revision and Retention in Case-Based Reasoning”, in Knowledge Engineering Review, vol.20(3), 2005.

Manna, “Online Personalization for E-Commerce: The Manna Advantage”, Manna Inc., 2000.

P.Massa, P.Avesani, “Trust-aware Collaborative Filtering for Recommender Systems”, in Proceedings of the International Conference on Cooperative Information Systems, Larnaka, Cyprus, 2004.

L.McGinty, B.Smyth. “Collaborative Case-Based Reasoning: Applications in Personalised Route Planning”, in Proceedings of the International Conference on Case-Based Reasoning, Vancouver, Canada, 2001.

D.L.McGuinness, F.van Harmelen, “OWL Web Ontology Language Overview”, World-Wide Web Consortium, 2004.

P.McJones, “EachMovie Collaborative Filtering Data Set”, HP Research, available online at http://research.compaq.com/SRC/eachmovie/, 1997.

S.M.McNee, S.Lam, J.Konstan, J.Riedl, “Interfaces for Eliciting New User Preferences in Recommender Systems”, in Proceedings of the International Conference on User Modeling, Pittsburgh, PA, 2003.

S.M.McNee, J.Riedl, J.A.Konstan, “Being Accurate is not enough: How Accuracy Metrics have Hurt Recommender Systems”, in Proceedings of the International Conference on Human Factors in Computing Systems, Montreal, Canada, 2006.

M.McTear, Special Issue on User Modeling, in Artificial Intelligence Review, vol. 7(3), 1993.

B.Mehta, C.Niederée, A.Stewart, “Towards Cross-system Personalization”, in Proceedings of the International Conference on Universal Access in Human-Computer Interaction, Las Vegas, NV, 2005a.

B.Mehta, C.Niederée, A.Stewart, M.Degemmis, P.Lops, G.Semeraro, “Ontologically Enriched User Profiling for Cross System Personalization”, in Proceedings of the International Conference on User Modeling, Edinburgh, UK, 2005b.

B.Mehta, “Cross System Personalization by Learning Manifold Alignments”, in Proceedings of the National Conference on Artificial Intelligence, Boston, MA, 2006.

S.E.Middleton, N.Shadbolt, D.De Roure, “Ontological User Profiling in Recommender Systems”, in ACM Transactions on Information Systems, vol. 22(1), pp. 54-88, 2004.

D.S.Milojicic, V.Kalogeraki, R.Lukose, K.Nagaraja, J.Pruyne, B.Richard, S.Rollins, Z.Xu, “Peer-to-Peer Computing”, Technical Report, HP Labs, 2002.

B.N.Miller, J.A.Konstan, J.Riedl, “PocketLens: Toward a Personal Recommender System”, in ACM Transactions on Information Systems, vol.22(3), pp.437-476, 2004.

N.Mirzadeh, F.Ricci, M.Bansal, “Feature Selection Methods for Conversational Recommender Systems”, in Proceedings of IEEE International Conference on eTechnology, eCommerce and eServices, Hong-Kong, 2005.

T.Mitchell, “Machine Learning”, McGraw-Hill, 1997.

M.Montaner, B.Lopez, J.L. de la Rosa, “A Taxonomy of Recommender Agents on the Internet”, in AI Review, vol.19(4), pp. 285-330, 2003.

R.J.Mooney, L. Roy, “Content-Based Book Recommending Using Learning for Text Categorization”, in Proceedings of ACM SIGIR Workshop on Recommender Systems, Berkeley, CA, 1999.

M.Morita, Y.Shinoda, “Information Filtering Based on User Behavior Analysis and Best Match Retrieval”, in Proceedings of the Annual International Conference on Research and Development in Information Retrieval, Dublin, Ireland, 1994.

M.D.Mulvenna, S.S.Anand, A.G.Buchner, “Personalization on the Net Using Web Mining”, in Communications of the ACM, vol.43(8), pp. 123-125, 2000.

M.V.Nagendra-Prasad, V.Lesser, S.Lander, “Retrieval and Reasoning in Distributed Case Bases”, in Journal of Visual Communication and Image Representation, Special Issue on Digital Libraries, vol. 7(1), 1996.

Napster, Napster Free, http://free.napster.com, accessed in June 2007.

M.Nelson, “We Have the Information you Want, but Getting it will Cost you: Being Held Hostage by Information Overload”, in ACM Crossroads, vol. 1(1), 1994.

C.Niederee, A.Stewart, B.Mehta, M.Hemmje, “A Multi-Dimensional, Unified User Model for Cross-System Personalization”, in Proceedings of Workshop On Environments For Personalized Information Access, Galipoli, Italy, 2004.

D.W.Oard, “The State of the Art in Text Filtering”, User Modeling and User-Adapted Interaction, vol. 7(3), pp. 141-178, 1997.

T.Olsson, “Decentralised Social Filtering based on Trust”, in proceedings of AAAI-98 Recommender Systems Workshop, Madison, WI, 1998.

A.Paiva, J.Self, “TAGUS – A User and Learner Modeling Workbench”, in User Modeling and User-Adapted Interaction, vol. 4(3), pp. 197-226, 1995.

J.Pascoe, “Adding Generic Contextual Capabilities to Wearable Computers”, in Proceedings of the International Symposium on Wearable Computers, Pittsburgh, PA, 1998.

M.Pazzani, D.Billsus, “Learning and Revising User Profiles: The Identification of Interesting Web Sites”, in Machine Learning , vol. 27(3), pp. 313–331, 1997.

M.Pazzani, “A Framework for Collaborative, Content-Based and Demographic Filtering”, in Artificial Intelligence Review, vol. 13(5-6), pp. 393-408, 1999.

D.M.Pennock, E.Horvitz, C.L.Giles, “Social Choice Theory and Recommender Systems: Analysis of the Axiomatic Foundations of Collaborative Filtering”, in Proceedings of the National Conference on Artificial Intelligence, Austin, Texas, 2000a.

D.M.Pennock, E. Horvitz, S.Lawrence, C.L.Giles, “Collaborative Filtering by Personality Diagnosis: A Hybrid Memory- and Model-Based Approach”, in Proceedings of the International Conference on Uncertainty in Artificial Intelligence, Stanford, CA, 2000b.

C.R.Perrault, J.F.Allen, P.R.Cohen, “Speech Acts as a Basis for Understanding Dialogue Coherence”, Technical Report 78-5, Department of Computer Science, University of Toronto, Canada, 1978.

D.Petrelli, A.De Angeli, G.Convertino, “A User Centered Approach to User Modelling”, in Proceedings of the International Conference on User modeling, Banff, Canada, 1999.

C.Plaxton, R.Rajaraman, A.Richa, “Accessing Nearby Copies of Replicated Objects in a Distributed Environment”, in Proceedings of the Symposium on Parallel Algorithms and Architectures, Newport, RI, 1997.

E.Plaza, J.L.Arcos, F.Martin, “Cooperative Case-Based Reasoning”, in Proceedings of the Workshop Distributed Artificial Intelligence Meets Machine Learning, Budapest, Hungary, 1996.

E.Plaza, L.McGinty, “Distributed Case-Based Reasoning”, in Knowledge Engineering Review, vol.20(3), 2005.

W.Pohl, “Logic-Based Representation and Reasoning for User Modeling Shell Systems”, in User Modeling and User-Adapted Interaction, vol. 9(3), pp. 217-282, 1999.

H.Polat, W.Du, “Privacy-Preserving Collaborative Filtering”, in the International Journal of Electronic Commerce, vol. 9(4), pp. 9-35, 2005.

O.Potonniee, “Ubiquitous Personalization: A Smart Card Based Approach”, in Proceedings of the Gemplus Developer Conference, Singapore, 2002.

J.Pouwelse, M. Slobbe, J.Wang, M.J.T.Reinders, H.Sips, “P2P-based PVR Recommendation using Friends, Taste Buddies and Superpeers”, in Proceedings of the Beyond Personalization Workshop, San Diego, CA, 2005.

M.Price, “Make it Personal: Using XML to Customize Web-Sites”, in Proceedings of XML Europe Conference and Exposition, Paris, France, 2000.

U.Rabanser, F.Ricci, “Recommender Systems: Do they have a Viable Business Model in E-Tourism?”, in Proceedings of the Annual Conference of the International Federation for IT & Travel and Tourism, Lausanne, Switzerland, 2005.

D.Randall-Wilson, T.R.Martinez, “Improved Heterogeneous Distance Functions”, in Journal of Artificial Intelligence Research, vol. 6, 1997.

S.Ratnasamy, P.Francis, M.Handley, R.Karp, S.Shenker, “A Scalable Content-Addressable Network”, in Proceedings of the Conference of the Special Interest Group on Data Communication, San Diego CA, 2001.

L.Razmerita, A.Angehrn, A.Maedche, “Ontology-Based User Modeling for Knowledge Management Systems”, in Proceedings of the International Conference on User Modeling, Pittsburgh, PA, 2003.

J.Reilly, K.McCarthy, L.McGinty, B.Smyth, “Dynamic Critiquing”, in Proceedings of the European Conference on Case-Based Reasoning, Madrid, Spain, 2004.

P.Resnick, N.Iacovou, M.Suchak, P.Bergstrom, J.Riedl, “GroupLens: an Open Architecture for Collaborative Filtering of Netnews”, in Proceedings of ACM Conference on Computer Supported Cooperative Work, Chapel Hill, NC, 1994.

P.Resnick, H.R.Varian, “Recommender Systems”, in Communications of the ACM, vol. 40(3), pp. 56-58, 1997.

F.Ricci, A.Venturini, D.Cavada, N.Mirzadeh, D.Blaas, M.Nones, “Product Recommendation with Interactive Query Management and Twofold Similarity”, in proceedings of the International Conference on Case-Based Reasoning, Trondheim, Norway, 2003.

F.Ricci, “Recommendations in Context”, in Proceedings of the International Conference on Electronic Commerce and Web Technologies, Krakow, Poland, 2006a.

F.Ricci, D.Cavada, N.Mirzadeh, A.Venturini, “Case-Based Travel Recommendations”, in D.R.Fesenmaier, H.Werthner, K.W.Wober (eds.) 'Destination Recommendation Systems: Behavioural Foundations and Applications', CABI, 2006b.

F.Ricci, D.R.Fesenmaier, N.Mirzadeh, H.Rumetshofer, E.Schaumlechner, A.Venturini, K.Wöber, A.Zins, “DIETORECS: a Case-Based Travel Advisory System”, in D.R.Fesenmaier, H.Werthner, K.W.Wober (eds.) 'Destination Recommendation Systems: Behavioural Foundations and Applications', CABI, 2006c.

E.Rich, “Building and Exploiting User Models”, Ph.D. Dissertation, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA, 1979a.

E.Rich, “User Modeling via Stereotypes”, in Cognitive Science, vol. 3, pp. 329-354, 1979b.

M.M.Richter, “Classification and Learning of Similarity Measure”, in Proceedings of the Annual Conference of the German Society for Classification, Dortmund, Germany, 1992.

A.Rowstron, P.Druschel, “Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems”, in Proceedings of the International Conference on Distributed Systems Platforms, Heidelberg, Germany, 2001.

G.Ruffo, R.Schifanella, “Evaluating Peer-to-Peer Recommender Systems that Exploit Spontaneous Affinities”, in Proceedings of the Symposium on Applied Computing, Seoul, Korea, 2007.

S.Russell, P.Norvig, “Artificial Intelligence: a Modern Approach”, Prentice-Hall, 1995.

H.Sakagami, T.Kamba, Y.Koseki, “Learning Personal Preferences on Online Newspaper articles for User Behaviors”, in Proceedings of the International World-Wide Web Conference, Santa Clara, CA, 1997.

G.Salton, M.McGill, “Introduction to Modern Information Retrieval”, McGraw-Hill, 1983.

R.Sandhu, E.Coyne, H.Feinstein, C.Youman, “Role-Based Access Control Models”, in IEEE Computers, vol.29(2), pp.38-47, 1996.

B.Sarwar, G.Karypis, J.Konstan, J.Riedl, “Analysis of Recommendation Algorithms for E-Commerce”, in Proceedings of the ACM Conference on Electronic Commerce, Minneapolis, MN, 2000.

B.Sarwar, G.Karypis, J.Konstan, J.Riedl, “Item-Based Collaborative Filtering Recommendation Algorithms”, in Proceeding of the International Conference on World-Wide Web, Orlando, FL, 2001.

B.M.Sarwar, G.Karypis, J.Konstan, J.Riedl, “Incremental SVD-Based Algorithms for Highly Scaleable Recommender Systems”, in Proceedings of the International Conference on Computer and Information Technology, Dhaka, Bangladesh, 2002.

J.B.Schafer, J.Konstan, J.Riedl, “Recommender Systems in E-Commerce”, in Proceedings of the ACM Conference on Electronic Commerce, Denver, CO, 1999.

J.B.Schafer, J.A.Konstan, J.Riedl, “Electronic Commerce Recommendation Applications”, in Data Mining and Knowledge Discovery, vol. 5(1-2), pp. 115-152, 2001.

M.Schlosser, M.Sintek, S.Decker, W.Nejdl, “A Scalable and Ontology-Based P2P Infrastructure for Semantic Web services”, in Proceedings of the International Conference on Peer-to-Peer Computing, Linköpings, Sweden, 2002.

A.I.Schein, A.Popescul, L.H.Ungar, D.M.Pennock, “Methods and Metrics for Cold-Start Recommendations”, in Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Tampere, Finland, 2002.

S.Schmitt, R.Bergmann, “Applying Case-Based Reasoning Technology for Product Selection and Customization in Electronic Commerce Environments”, in Proceedings of E-Commerce Conference, Bled, Slovenia, 1999.

U.Shardanand, P.Maes, “Social Information Filtering: Algorithms for Automating Word of Mouth”, in Proceedings of the International Conference on Human Factors in Computing Systems, Denver, CO, 1995.

T.B.Sheridan, W.R.Ferrell, “Man-Machine Systems: Informaiton, Control, and Decision Models of Human Performance”, Cambridge, MA, MIT Press, 1994.

D.Sleeman, “UMFE: A user Modeling Front-End Subsystem”, in International Journal of Man-Machine Studies, vol. 23, pp. 71-88, 1985.

B.Smyth, P.Cotter, “A Personalized TV Listings Service for the Digital TV Age”, in Journal of Knowledge-Based Systems, vol. 13(2-3), pp. 53-59, 2000.

B.Smyth, P.Cunningham, “The Utility Problem Analysed: a Case-Based Reasoning Perspective”, in Proceedings of the European Workshop on Case-Based Reasoning, Lausanne, Switzerland, 1996.

B.Smyth, L.McGinty, J.Reilly, K.McCarthy, “Compound Critiques for Conversational Recommender Systems”, in Proceedings of the International Conference on Web Intelligence, Beijing, China, 2004.

B.Smyth, “Adaptive Information Access and the Quest for the Personalization-Privacy Sweetspot”, in Proceedings of the International Conference on Intelligent User Interfaces, San Diego, CA, 2005.

B.Smyth, “Adaptive Information Access: Personalization and Privacy”, in International Journal of Pattern Recognition and Artificial Intelligence, vol. 21(2), pp. 183-206, 2007.

M.Svensson, J.Laaksolahti, K.Hook, A.Waern, “A Recipe Based On-Line Food Store”, in Proceedings of the International Conference on Intelligent User Interfaces, New Orleans, LA, 2000.

L.Sweeney, “k-Anonymity: A Model for Protecting Privacy”, in International Journal on Uncertainty, Fuzziness and Knowledge-based Systems, vol.10(5), pp. 557-570, 2002.

E.Tanin, D.Nayar, H.Samet, “An Efficient Nearest Neighbor Algorithm for P2P Settings”, in Proceedings of the National Conference on Digital Government Research, Atlanta, GA, 2005.

N.Tintarev, “Explaining Recommendations”, in Proceedings of the International Conference on User Modeling, Corfu, 2007.

B.Towle, C.Quinn, “Knowledge Based Recommender Systems Using Explicit User Models”, in AAAI Workshop on Knowledge-Based Electronic Markets, Menlo Park, CA, 1999.

D.A.Tran, “Hierarchical Semantic Overlay Approach to P2P Similarity Search”, in Proceedings of the USENIX Annual Technical Conference, Anaheim, CA, 2005.

T.Tran, R.Cohen, “Hybrid Recommender Systems for Electronic Commerce”, in Proceedings of the Workshop on Knowledge-Based Electronic Markets, Austin, TX, 2000.

A.Tveit, “Peer-to-Peer Based Recommendations for Mobile Commerce”, in Proceedings of the International Workshop on Mobile Commerce, Rome, Italy, 2001.

M.Vozalis, K.G.Margaritis, “Enhancing Collaborative Filtering with Demographic Data: The Case of Item-Based Filtering”, in Proceedings of the International Conference on Intelligent Systems Design and Applications, Budapest, Hungary, 2004.

W.Wahlster, A.Kröner, D.Heckmann, “SharedLife: Towards Selective Sharing of Augmented Personal Memories”, in O.Stock, M.Schaerf (eds.) 'Reasoning, Action and Interaction in AI Theories and Systems', Springer, 2006.

Y.Wang, A.Kobsa, “Impacts of Privacy Laws and Regulations on Personalized Systems”, in Proceedings of the Workshop on Privacy-Enhanced Personalization, Montreal, Canada, 2006.

P.Waszkiewicz, P.Cunningham, C.Byrne, “Case-Based User Profiling in a Personal Travel Assistant”, in Proceedings of the International Conference on User Modeling, Banff, Canada, 1999.

I.Watson, “Applying Case-Based Reasoning: Techniques for Enterprise Systems”, Morgan Kaufmann, 1997.

I.Watson, D.Gardingen, “A Distributed Case-Based Reasoning Application for Engineering Sales Support”, in Proceedings of the International Joint Conference on Artificial Intelligence, Stockholm, Sweden, 1999.

G.I.Webb, M.J.Pazzani, D.Billsus, “Machine Learning for User Modeling”, in User Modeling and User-Adapted Interaction, vol. 11(1-2), pp. 19-29, 2001.

M.Weiser,”The Computer for the 21st Century”, in Scientific American, vol. 265(3), pp. 94-104, 1991.

S. Wess, K.D. Althoff, G. Derwand, “Using K-d Trees to Improve the Retrieval Step in Case-Based Reasoning”, in Proceedings of the European Workshop on Case-Based Reasoning, Kaiserslautern, Germany, 1993.

I.H.Witten, A.Moffat, T.C.Bell, “Managing Gigabytes”, Morgan Kaufmann, 1999.

I.H.Witten, E.Frank, “Data Mining: Practical Machine Learning Tools and Techniques”. Morgan Kaufmann, 2005.

M.Wooldridge, “An Introduction to Multi-Agent Systems”, John Wiley, 2002.

J.Zhang, P.Pu, “A Comparative Study of Compound Critique Generation in Conversational Recommender Systems”, in Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems, Dublin, Ireland, 2006.

C.N.Ziegler, S.M.McNee, J.A.Konstan, G.Lausen, “Improving Recommendation Lists through Topic Diversification”, in Proceedings of the International World-Wide Web Conference, Chiba, Japan, 2005.