TU WIEN

TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

Ph. D. Thesis

Adaptive Personalization
A multi view personalization approach incorporating contextual information

Conducted for the purpose of receiving the academic title
Doktor der technischen Wissenschaften

Supervisor
Silvia Miksch
Institute of Software Technology & Interactive Systems (E188)

Submitted at Vienna University of Technology
Faculty of Informatics

by

ERICH CHRISTOPH GSTREIN

8226167
Lorenz Mandlgasse 46/9, 1160 Wien
gstrein@acm.org

Vienna, November 10, 2009

(Signature Author)

(First Supervisor)

(Second Supervisor)

Zusammenfassung

Informationsüberflutung ist längst nicht mehr nur ein modernes Reizwort, sondern bereits ein ernstes Problem mit volkswirtschaftlicher Dimension. So veröffentlichte das amerikanische Wirtschaftsforschungsunternehmen Basex[1] 12/2008 eine Studie, in der allein die durch Informationsüberflutung verursachten Verluste der amerikanischen Wirtschaft auf \$900 Milliarden beziffert wurden. Als geeignete Mittel zur Eindämmung oder Entschärfung dieses Problems haben sich in den letzten Jahren vor allem Personalisierungslösungen etabliert, die dem Benutzer individualisierte Filtermechanismen zur Seite stellen. Dabei kommt dem sorgsamen Umgang mit den verfügbaren Daten zentrale Bedeutung zu, da dies die Brauchbarkeit bzw. Qualität dieses Services wesentlich beeinflusst.

In dieser Arbeit wird *Adaptive Personalization*, ein neuartiger Personalisierungsansatz vorgestellt, der die vorhandenen Informationen besser verwertet als bisherige Verfahren. Im Gegensatz zu anderen kontextuellen Verfahren werden zur Berechnung von Vorschlägen nur solche Informationen verwendet, die in jedem Standardszenario verfügbar sind. Die effiziente Nutzung dieser Informationen wird an Hand der Erstellung eines neuartigen Profilmodells sowie verbesserter Algorithmen demonstriert. So wird ein flexibles, mehrschichtiges Profilmodell vorgestellt, das neben der Modellierung von kurz- und langfristigen Vorlieben sowie Vergessen auch den Ursprung von Kundenbewertungen (*context of origin*) berücksichtigt. Zur Berechnung der *k-nächsten Nachbarn* eines Profils wird der D^2-Tree, eine neue Indexstruktur, präsentiert, der den dynamischen Umgang mit (kontextuell) wechselnden Distanzfunktionen im Suchraum unterstützt. Weiters wird ein neuartiger Collaborative-Filtering Algorithmus vorgestellt, der Konzepte aus Information Retrieval und Association Rule Mining zur Steigerung der Vorhersagegüte nutzt und so bessere Ergebnisse erzielt als Standardverfahren. Vorschläge für eine flexible, skalierbare Systemarchitektur sowie für ein Vorgehensmodells zum Systementwurf vervollständigen die Präsentation des *Adaptive Personalization* Ansatzes.

Zur Evaluierung der entwickelten Konzepte wurden unterschiedliche Vorgehensweisen gewählt. So wurde die Brauchbarkeit des Profilmodells empirisch an Hand von Echtdaten untersucht während für den D^2-Tree

[1] `http://www.basex.com`; 2/2009

eine klassische, ausführliche Kostenanalyse erstellt wurde. Die Güte des Collaborative Filtering Algorithmus wurde, zur besseren Vergleichbarkeit mit anderen Techniken, auf Basis von Standard Datensätzen und Metriken evaluiert.

Auf Grund der neuartigen Nutzung vorhandener Informationen, des flexiblen Profilssystems, der effizienten Algorithmen und der robusten Architektur stellt der hier präsentierte Ansatz einen sinnvollen und nützlichen Beitrag zur Bewältigung der Informationsüberflutung dar.

Abstract

Information overload has become a significant problem causing losses in economic relevant dimensions. According to a study of Basex[2] published in 12/2008 only the U.S. economy loses at least $900 billion per year due to lowered employee productivity and reduced innovation. Personalization systems, generating individual suggestions based on user models, are seen as one major concept for solving this urgent problem of information overload. However, providing an appropriate personal assistance requires a diligent usage of the available information.

In this thesis we propose a personalization approach – called *Adaptive Personalization* – utilizing the available, context specific information in a new and efficient way. In addition to other approaches tackling the incorporation of contextual information, we only rely on minimal data available in standard recommendation scenarios. We will show how the knowledge about the origin of user feedback – called *context of origin* – can be used to construct an enhanced multi-view profile model, also incorporating long- and short-term preference aspects as well as neglect. For solving the k-nearest neighbors problem in a high dimensional search space, stretched by the attributes of these profiles, a new index structure – called D^2-Tree – will be presented supporting the dynamic change of (contextual) distance functions. Furthermore, we introduce a new collaborative filtering algorithm where techniques borrowed from information retrieval and association rule mining where used to improve prediction quality, outperforming established approaches especially in contexts with minimal information. Additionally, a robust and flexible architecture is presented suitable for large scale, real world application scenarios together with a proposal for a procedure model concerning system design. While the effectiveness of our profile model was determined on the basis of real world data, the collaborative filtering algorithm was evaluated using popular test data sets and metrics. Furthermore, detailed cost analyzes concerning the different index operations are provided for the D^2-Tree.

Due to this new and efficient utilization of the available information, its light-weight profile model, the efficient algorithms and the flexible architecture, the *Adaptive Personalization* approach is a useful and valuable contribution for solving the problem of information overload.

[2] http://www.basex.com as of 2/2009

Acknowledgements

This thesis is dedicated to my kids **Christoph**, **Lukas** and **Ines** who mostly suffered from the time, I was forced to withhold, for finishing my work. Without the comprehensive support and endless energy of my spouse **Patrizia** in managing the family it would have been impossible for me to write this thesis.

Deep gratitude is owed to my supervisor **Silvia Miksch** for her great support and encouragements throughout the whole project. Furthermore, I would like to thank **Brigitte Krenn** for her discussions, criticisms and motivating feedback. Special thanks go to **Oliver Holle** for providing the economical context for my research activities over many years and for his belief in the results and the people involved.

Moreover I want to thank **Christoph Schmotzer** for his inputs, discussions and his role as the users advocate. I also want to thank **Robert Mayer** for his discussions and criticisms concerning technical topics in early phases of this project. Many thanks to my colleagues **Florian Kleedorfer** and **Roman Cerny** for their inspiring discussions and feedback. Furthermore, I want to thank **Brigitte Krenn**, **Roman Cerny** and **Florian Kleedorfer** for proof reading my thesis.

Finally, I want to thank the *Research Studio Austria Forschungsgesellschaft mbH* for providing the thematical context. This work was partially funded from the *EC Cognitive Systems Project FP6-IST-027596-2004 RASCALLI*.

Contents

1. Introduction

In this chapter the motivation and research questions concerning the thesis are set out together with a brief introduction to personalization.

1.1. Motivation

Large parts of this work were done in a series of research and development (R&D) projects – starting 2002 – preparing and accompanying the implementation of a personalization solution for a mobile download platform, focused on music related content, such as full audio files, wallpapers, etc. The first version of this application went on-line end of 2005 and was deployed in more than 15 countries all over the world.[1]

In the course of these R&D projects many of the existing and well-known personalization/recommendation techniques were analyzed for applicability in the given context with the result that a broader approach is necessary to meet the given constraints. Beside the common problems of recommender/personalization systems such as the cold-start problem, the new item/user problem, etc.[2] the domain of interest – mobile music – bears some specific traps a personalization system must overcome.

Accessability: The access to content via mobile devices suffers from several technical short comings: (i) exhausting interaction when entering data, (ii) limited display area, and (iii) performance. These limitations have strong impacts on the profiling process as well as on the appropriate explanations of recommendations. These are:

Content Meta Data: Although available in principle, provided content meta data is often hardly usable because of its bad quality. Music genres are an often disgraced concept, however indispensable to a music portal. The most serious problem with genres is that they are not standardized and that they tend to be a source of dispute. For example, the AllMu-

[1] *Ericsson's Media Suite - Music* (working title first launch)
[2] See Chapter 3 on page 15

sicGuide[3] offers 531 music styles, Amazon.com[4] 719 and MP3.COM[5] about 430 (Uitdenbogerd and van Schnydel, 2002). Furthermore, in commercial contexts the genre affiliation is often used as a marketing instrument for promoting a specific artist. The results of these practices range from excessive classifications of artists to different genres to obviously personally motivated affiliations insisted by the artist her/himself. Furthermore, content providers often do not or cannot deliver genre information at all or they deliver a standard classification for many content items.

During the projects we were confronted with data sets, provided by global content providers, having artists with more than 130 genre affiliations (e.g., Michael Jackson) or where 70% of all content items were classified to one genre (e.g., "*classic pop*"). Furthermore, one global content provider did not deliver genre meta data for 20% of it's content due to internal differences.

Content volume and life-cycle: Music portals often use huge music archives with rapidly increasing content offering millions of songs to the users (e.g., at the time of writing this document iTunes[6] promised more than 8 million audio tracks). Furthermore, the music industry produces more and more "nine days wonders", such as the annual "summer hits" and their performers. From a recommender's point of view, the complexity of the new item problem[7] (Herlocker et al., 2004; Adomavicius and Tuzhilin, 2005) rises.

User Meta Data: User meta data, stored in the databases of the operators, could often not be used due to legal issues.

Customer retention: As observed in early mobile applications most users tend to be occasional customers and do not return very frequently. As presented in Chapter 8 more than 80% of the users only purchased twice!

Cultural dependency: The cultural background of the music consumers plays an important role (Uitdenbogerd and van Schnydel, 2002), because it has many impacts concerning profiling and recommendation aspects.

Beside these more music specific constraints, a personalization system

[3]`http://www.allmusic.com` as of 08/2008
[4]`http://www.amazon.com` as of 8/2008
[5]`http://www.mp3.com` as of 8/2008
[6]`http://www.itunes.com` as of 8/2009
[7]The *new item problem* is common in recommender systems based on item ratings. The problem is: How to recommend an item which was never rated/bought before?

also has to face a variety of additional challenges, mainly addressing technical or business issues. The most important are:

Performance and scalability: Millions of items together with potentially millions of users define a high-performance scenario which must be handled by the personalization system, concerning the quality of the recommendation algorithms as well as the system response time due to the amount of concurrent users.

Administrability: The person responsible for maintaining the personalization system must be optimally supported to allow effective tuning of the system's performance. This implies, that the system must be as transparent as possible. Furthermore, many operators articulated the need for manipulating recommendations as an instrument of implementing market campaigns.

Cross domain support: Cross- and up-selling are magic words capable of opening doors for personalization solutions on a business level. Although some standard techniques such as association rule mining (also known as shopping cart analysis) (Agrawal and Srikant, 1994) or pure rating based solutions as *Slope ONE* (Lemire and Maclachlan, 2005) can be applied on a domain agnostic level, the performance of such solutions can often be improved significantly by adding domain-specific knowledge.
For example: recommending yet another book or song to a person just having bought such an item (often explained as *users who bought A also bought...*) might be a good idea but will not work well for ringtones or wallpapers in most cases, because of the different usages of these items[8]. Furthermore, applying domain specific techniques for defining item similarity (e.g. sound similarity in the music domain (Aucouturier and Pachet, 2002)) can dramatically improve the performance of the system.

Summing up, the main challenge was to develop a scaleable, domain independent personalization system capable of dealing with a minimum set of available meta data. The concept to be developed was called *Adaptive Personalization*[9] which will be explained throughout this thesis. Although the challenges mentioned above were tackled somehow in the course of the different R&D projects (and most of the solutions found will be presented in this work) this thesis concentrates on how to get better results out of existing information. More precisely: Is there any (still) unused information available in standard scenarios and how can it be used to improve

[8]In many cases ringtones or wallpapers are used to restyle a hand-held device periodically – but most users do not buy them ahead.

[9]This name was chosen due to the adaptive capabilities of this approach concerning the users needs as well as the demands of the operators of a personalization system.

the recommendation process?

1.2. Research Question

As the basis of our research a set of research questions were formulated.

1.2.1. Main Question

Which existing contextual information can be used to improve a personalization system and how can it be applied?

1.2.2. Sub Questions

- What are the existing techniques/approaches and why they are not sufficient for the given task?
- How can contextual information be used to improve user and item models?
- How can contextual information be used to improve rating based recommendation strategies?

1.3. Objectives

The major goal of this work is to provide a theoretical basis for the core concepts of *Adaptive Personalization*, a pragmatic, domain independent and flexible approach for creating scaleable personalization systems, applicable for real world scenarios. We will show how available information, unused by other approaches, can be used to improve user and item models as well as rating based recommendation algorithms. In more detail, the work presents

- how the context of user feedback can be used to improve user models
- how rating based collaborative filtering and k-nearest neighbor algorithms can be improved by adding contextual information
- a flexible architecture for real world applications

1.4. A Brief Survey of Personalization

Although the amount of digital information accessible is increasing dramatically every few months, the usability of this giant digital library or warehouse is getting from bad to worse. This situation is well characterized by the following quote

“We are drowning in information but starved for knowledge”

by John Naisbitt in his book *Megatrends* in 1982. Still valid today – more than 25 years later – information overload is not a modern plague infecting our information society. Barnabe Rich, a British author and soldier, complained

> “One of the diseases of this age is the multiplicity of books; they doth so overcharge the world that it is not able to digest the abundance of idle matter that is every day hatched and brought forth into the world”

in 1613 almost 400 years before Amazon.com[10] went on-line!

The application of personalization systems is seen as one major concept to solve the problem of information overload.

1.4.1. What is Personalization?

Although the word *personalization* is very trendy in e- and m-commerce[11], it should be stressed, that this concept is not restricted to interactive media:

“Personalization involves customizing some feature of a product or service so that the customer enjoys more convenience, lower cost, or some other benefit.”

Following this definition, as found in the *Glossary of Terms* of the Personalization Consortium[12] in 2005, it is obvious, that no particular technology, but a lot of information is required to implement an appropriate personalization strategy. Therefore each retailer serving his/her customers individually is performing personalization – e.g. by targeted recommendations of fitting items such as music, books, etc.

In the context of e- and/or m-commerce, personalization is the combined usage of technology and information about customers to tailor electronic

[10]`http://www.amazon.com` as of 9/2008
[11]e- and m-commerce are abbreviations for *electronic* and *mobile* commerce
[12]`http://www.personalization.org` as of 2005

commerce interactions between a business and each individual customer. The purpose of this information technology combined with marketing practices is to:

- Better serve the customer by anticipating needs
- Make the interaction efficient and satisfying for both parties
- Build a relationship that encourages the customer to return for subsequent purchases

1.4.2. The Benefits of Personalization

From a business point of view the advantages of personalization are obvious: A better, targeted presentation/recommendation of items will lead to more business, increasing the ARPU[13]. But why should a customer use personalization – especially in view of the *paradox of the active user*[14] and the problem of privacy concerns?

A study, conducted by Swearingen and Sinha (2002) in the context of using recommender systems for on-line stores selling books and movies, compared the acceptance of recommendations created by systems to those made by friends. Although users generally preferred recommendations given by friends to those made by a system, they expressed a high level of satisfaction with the on-line recommendations. The reason for this broad acceptance was that the users welcomed the opportunity to explore their taste and learn about new items based on the huge set of data the recommender system operates on.

1.4.3. The Heart of Personalization

As stated above, personalization is the attempt to best serve a user/customer by knowing his/her individual needs and/or preferences. To solve this task the following information is needed:

1. knowledge about the user and his/her needs also including the evolution of personal preferences over time
2. knowledge or at least provision about items and their ability to satisfy these needs

[13]ARPU stands for average revenue per user, a metric often used in the telecom industry.

[14]The *paradox of the active user* – introduced by Carroll and Rosson (1987) – postulates, that users of software never read manuals, but start using it immediately, although they would save time in the long term by spending some time to initialize the system.

3. a process or algorithm for finding the best match between needs and items
4. a process or strategy to gather information about the user – see item 1

Gathering information and processing it in an appropriate way is the core task of personalization, often simplified as *profiling&matching*.

The first and the fourth item refer to the problem of the definition, creation and refinement of user profiles which can be seen as the most critical task in the area of personalization. Item two addresses the availability of special domain knowledge and item three tackles algorithmical challenges.

1.5. Contributions of our Approach

While many available personalization systems are based on some recommending algorithms, the approach at hand tries not only to tackle personalization as a data mining and/or user modeling problem, but also sets it in a broader context where the optimal satisfaction of the user's needs as well as the requirements of the operators are in focus. In contrast to other approaches tackling the incorporation of contextual information (Adomavicius et al., 2005; Kim and Kwon, 2007), we rely on the information available in standard recommendation scenarios consisting of:

1. explicit user actions, such as ratings, buys, etc.
2. information based on the navigation behavior of users
3. explicit data entered by the user (e.g. demographic or preference information during a registration process)
4. meta data from the problem domain (e.g. information about genre similarities)

We will show how this information is used to construct a domain agnostic personalization approach facing the problems mentioned above. The major building blocks of our concept are:

- a light-weight but sophisticated profile system for modeling different views of users and items based on the Johari window, a cognitive psychological tool developed by Luft and Ingham (1955)
- appropriate algorithms for efficiently using the information available

- a well-defined set of recommendation strategies focusing on the demands and needs of the users
- a flexible architecture for real world scenarios

where the profile model (see Chapter 6) together with new recommender algorithms as presented in Chapter 7 are our main contributions to recommender research.

1.6. Dissemination

As mentioned above, parts of this work formed the scientific basis for implementing a personalization engine serving a mobile music portal. This personalization engine was launched in several countries in Asia, Europe, and the USA.

Furthermore, parts of this work were used and further developed in national and international R&D projects such as:

- RASCALLI: Responsive Artificial Situated Cognitive Agents Living and Learning on the Internet ; EU FP6-2004-IST-4
- SEMPRE: SEMantically aware Profiling for REcommenders; FIT-IT 2007

In RASCALLI the user modeling component of the *Adaptive Personalization* approach was extended with the ability of neglect (see Chapter 6). Furthermore, the improved collaborative filtering algorithm, presented in Chapter 7, was partly developed within RASCALLI. In SEMPRE an implementation of the *Adaptive Personalization* approach, called *easyrec*[15], was used as the basis for the development of the demonstrators. While writing this thesis, *easyrec*, is being prepared for going public as an open source project.

Furthermore, some core concepts and results of this work were presented and published at the following conferences:

- Krenn et al. (2009): Adaptive Mind Agent. 9th International Conference on Intelligent Virtual Agents, Amsterdam 14-16 September, 2009.
- Gstrein and Krenn (2006): Mobile Personalization at Work. Workshop on Recommender Systems, ECAI 06, Riva del Garda, 18-19 August , 2006.

[15]`http://www.easyrec.org` as of 7/2009

- Gstrein et al. (2005): Adaptive Personalization: A Multi-Dimensional Approach to Boosting a Large Scale Mobile Music Portal. In Fifth Open Workshop on MUSICNETWORK: Integration of Music in Multimedia Applications, Vienna, Austria.

While in (Gstrein et al., 2005) and (Gstrein and Krenn, 2006) the principles and experiences of the Adaptive Personalization are presented, the integration of the profile model for realizing an adaptive mind component for virtual agents is described by Krenn et al. (2009). Furthermore, first thoughts concerning some topics of the approach at hand were parts of several publications e.g.,

- Krenn and Gstrein (2006): On Female and Male Avatars: Data from a Web-Based Flirting Community. In Proceedings of the AVI 2006 Workshop on Gender and Interaction. Real and virtual women in a male world, 2006.

- Krenn et al. (2004): Lifelike Agents for the Internet: A Cross-Cultural Case Study. In Agent Culture: Human-Agent Interaction in a Multicultural World, 2004.

- Gstrein (2003): The BPnD-Tree: An efficient search structure for high dimensional profile based recommendation systems. Technical report. Austrian Research Centers 2003.

- Krenn et al. (2002): What Can We Learn from Users of Avatars in Net Environments? Proceedings of AAMAS 2002 Workshop:Embodied Conversational Agents – Let's Specify and Evaluate Them! July 15-16 2002.

While in Gstrein (2003) the basics of the D^2-Tree (see Chapter 7) were developed, early versions of the profile models formed the basis of analyses presented in Krenn and Gstrein (2006), Krenn et al. (2002) and Krenn et al. (2004).

Furthermore, the R&D activities related to the approach at hand initiated several master thesis accompanying the implementation of the personalization solution for *Ericsson's Media Suite - Music*. Kleedorfer (2008) developed an algorithm for finding similar songs based on lyrics analysis for extending music recommendations to textual patterns. Schnabel (2007) analyzed the applicability of different information visualization techniques on user actions for supporting the recommender administrator in finding appropriate system settings. In the course of his master thesis Cerny (2008) implemented a generic recommender system, based on the architecture of the *Adaptive Personalization* approach (see Chapter 5) in *Java*.

1.7. Overview

The rest of the document is structured as follows. First, in Chapter 2, definitions, conventions and prerequisites are introduced forming the basis of our discussion. In Chapter 3 a survey of the state-of-the-art in recommender systems is given. Furthermore, a short overview of user modeling servers is provided. A bird's eye view concerning the concepts of the *Adaptive Personalization* approach is presented in Chapter 4, also demonstrating specific considerations concerning the *Ericsson's Media Suite - Music*, the mobile content download platform for which the approach at hand was initially developed.

After these introductory chapters the core concepts of the approach at hand are explained in detail. While the system architecture is presented at a conceptual level in Chapter 5, the multi-view profile model is discussed in detail in Chapter 6. Two new recommender algorithms are presented in Chapter 7.

In Chapter 8 evaluation results based on two real world application scenarios – one from Europe and one from Asia – are presented. Due to several restrictions the evaluation must be seen as an qualitative analysis only. A short summary of the work together with future research directions is presented in Chapter 10. Detailed evaluation results can be found in the Appendix together with screen-shots of the *Ericsson's Media Suite - Music*, which are used to explain the realization of the presented concepts.

2. Conventions & Prerequisites

The definitions, conventions and prerequisites presented in this chapter are used or referred to in the following discussions. While in the first section definitions and conventions are provided, the second section is dedicated to the Johari window (Luft and Ingham, 1955), a cognitive psychological tool, which inspired the development of the multi-view profile model as described in Chapter 6.

2.1. Definitions & Conventions

From the various descriptions of *contextual information* available in the literature (Brown et al., 1997; Pascoe et al., 1998; Schmidt et al., 1999), we follow the definition provided by Dey and Abowd (2000):

> "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."[1]

Most personalization approaches incorporating contextual information (e.g. (Kim and Kwon, 2007; Adomavicius et al., 2005)) rely on narrower definitions and try to consider the user's context when interacting with the system e.g. "at work", "with friends", etc.[2].

In contrast to these approaches, the term *contextual information* is treated in a more differentiated way in the *Adaptive Personalization* approach. Concerning profile adaptation, the context describes the origin or source of the feedback used for refinement. So, for example, the feedback used for sharpening user profiles may come from (i) explicit rating actions of the user (e.g. rating an item, buying an item), (ii) observing the users's navigation behavior by the system and (iii) feedback from other users (see Chapter 6).

In the process of finding similar users, based on collaborative ratings, contextual information refers to different aspects of the relationship of these preference sets, incorporating information such as the number of ratings, the frequency of rated items and the degree of overlap of the rating

[1]Dey and Abowd (2000); p. 3

[2]Some times called *context of use* see (Niederée et al., 2004)

profiles (see Chapter 7).

Chapters are written as self-contained units, each providing a short introduction as well as a summary. If a chapter directly addresses a research question, this summary is named *Contributions to the Research Question*. Furthermore, code snippets used for explaining algorithms are written in pseudo-code similar to the *Java* programming language.

2.2. Johari Window

The Johari window (Luft and Ingham, 1955) was developed by two American psychologists, Joseph Luft and Harry Ingham, in the 1950s during research concerning group dynamics[3]. The Johari window model, also referred as a *feedback/disclosure model of self awareness*, classifies information about a person in relation to a group along the two dimensions *self/others* and *known/unknown* in four different areas, as shown in 2.1.

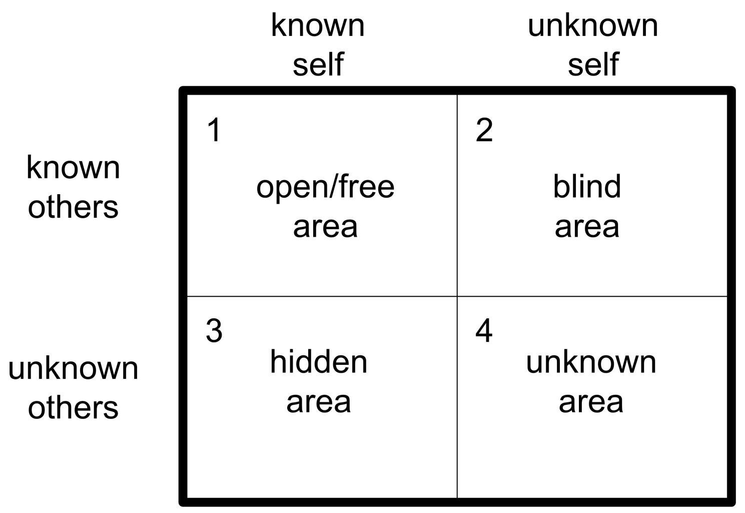


Figure 2.1.: The Four Areas of the Johari Window

Quadrant 1, the *open/free area* or *area of free activity* (Luft and Ingham, 1955), consists of the information known by the individual him/herself and by the community. This quadrant is the public arena where individuals have their personal power and voice. Basically this quadrant is small for team members providing no feedback.

Quadrant 2, the *blind area* or *blind spot*, contains information which the individual about him/herself does not know but which is known by the others. This area can also be referred to as ignorance about oneself or as-

[3]The name *Johari* was derived from a combination of the first names Joseph and Harry

pects one is misguided. Thick-skinned people often have a large blind area.

Quadrant 3, the *avoided/hidden area* or *facade*, contains information a person knows about him/herself but which is hidden from the others. While private information or feelings should remain hidden all team relevant information of this area should be moved to quadrant 1 for improving teamwork.

Quadrant 4, the *unknown area* or *area of unknown activity*, contains information about a user unknown by him/herself and the community. Typically young people or those having little experience or self-confidence have a large unknown area.

For optimizing group performance it is necessary to enlarge the *open/free area* based on community feedback and self-disclosure as presented in Figure 2.2.

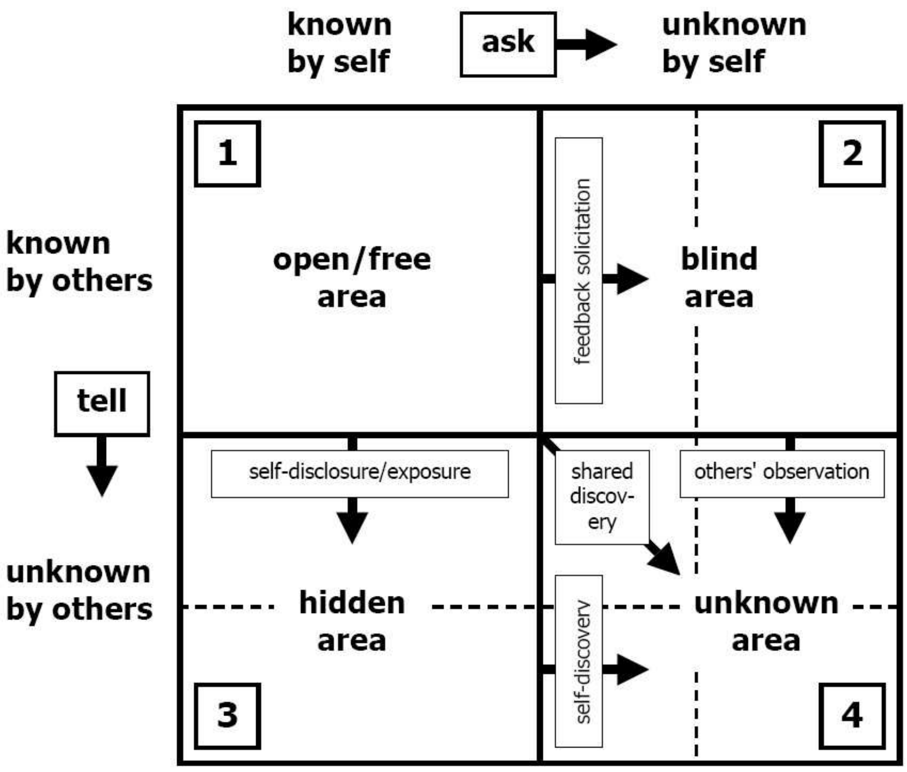


Figure 2.2.: Johari Window Model[4]

[4]©alan chapman 2003; cp. `http://www.businessballs.com/johariwindowmodeldiagram.pdf` as of 11/2008

3. Related Work

Although the term personalization became popular in the course of the rise of e-commerce applications in the mid 90ties, research activities concerning user adaptive systems can be traced back much further. Among others, *generic user modeling* and *recommender systems* can be seen as the most important research areas contributing to personalization. The presentation of the state-of-the-art of these research fields is the goal of this chapter. In the first section we will have a closer look at the history and state-of-the-art in *recommender systems*, because this topic is more related to our *Adaptive Personalization* approach than *generic user modeling* which will be discussed in the second part of this chapter.

3.1. Recommender Systems

Recommender systems are systems that produce individualized suggestions concerning interesting objects the user might like out of a large number of alternatives.

In absence of a common definition of recommender systems in the research community, many authors follow the description given above (Terveen and Hill, 2001; Ujjin and Bentley, 2001; Burke, 2002; Riedl and Dourish, 2005). An earlier definition made by Resnick and Varian (1997) constrict recommender systems to applications aggregating recommendations made by users and directing them to appropriate recipients. Furthermore, the terms *recommender systems* and *collaborative filtering* are often used in the literature interchangeable. Additionally, the term *recommender systems* is often picked to describe specific algorithms or techniques (Terveen and Hill, 2001) such as collaborative filtering or content-based filtering.

In our work and this PhD thesis we use the definition of recommender system given above (following e.g., Burke (2002)) and address the different predicting approaches (e.g., collaborative filtering) as *algorithms* or *techniques* (following e.g., Breese et al. (1998)). So, in our context, a *recommender system* is an application/system generating item recommendations for users by applying a set of different recommending *approaches* or *techniques*.

3.1.1. Historical Survey

Although the origin of recommender systems can be traced back to the late 70ties, research on this topic emerged as an own research area in the mid 90ties. Grundy[1], a book recommender system based on user stereotypes developed Rich (1979), can be seen as the first recommender system. In 1992 Goldberg et al. (1992) coined the term *collaborative filtering* during the development of *Tapestry* a mail system supporting the user in handling the stream of incoming mails, based on manually defined like minded user-user relations.

Based on the findings of Tapestry and the concentration on rating based recommendation approaches, the rise of recommender systems as an independent research area started in the mid 90ties. Systems such as *Ringo* (Shardanand and Maes, 1995), *GroupLens* (Konstan et al., 1997), or *Video Recommender* (Hill et al., 1995) were the first recommender systems using collaborative filtering for automatically generating predictions. Based on the findings of this research many improvements concerning collaborative filtering algorithms were made subsequently (Breese et al., 1998).

However, it was reserved to Amazon.com[2] to make recommender systems popular, even widely beyond the research community. In the late 90ties Amazon.com launched a book recommending service, called *BookMatcher*, on their e-commerce portal. But due to the fact, that users had to provide about 20 to 30 ratings to get recommendations, BookMatcher was hardly used. Based on these experiences Linden et al. (2003) developed a new recommender system based on an *item-based collaborative filtering* approach published by Sarwar et al. (2001). The success of Amazon.com's personalization solution had a great impact on e-commerce and formed the bases for widely used terms such as *long tail*[3]. Furthermore, the phrase "*Amazon like recommendations*" is widely used as a common feature description in e-commerce business.

Based on experience with collaborative filtering approaches some research on hybrid systems was conducted, by combining different techniques to overcome the shortcomings of each approach. While *Fab* – a system recommending web pages to user developed by Balabanović and Shoham (1997) – is an early example of a hybrid recommender combining content-based filtering with collaborative filtering, *EntreeC* – a

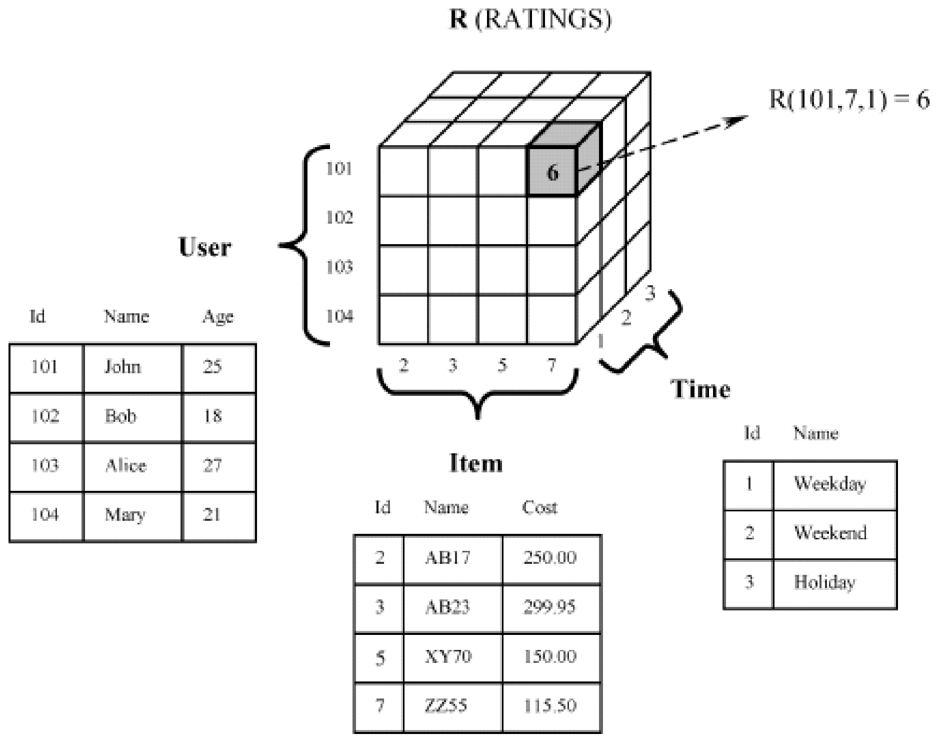
[1]Grundy is also referred as an early generic user modeling system see Kobsa (2001)

[2]`http://www.amazon.com` as of 6/2008

[3]Inspired by an event were Amazon.com's recommender turned a shopkeeper into a best-seller C. Anderson, chief editor of the *Wired Magazine*, coined the term *long tail business* in 2004, `http://www.wired.com/wired/archive/12.10/tail.html`

restaurant recommender system developed by Burke (2002) – combines knowledge-based approaches with collaborative filtering.

Furthermore, contextual information was taken into account to improve recommendation accuracy. Adomavicius et al. (2005) proposed a multi-dimensional approach for calculating recommendations. In Figure 3.1 a 3-dimensional model is presented incorporating time into the recommendation process. The basic idea is, that different contexts call for different recommendations, even within the same domain and preference space. So, for example, the decision which film a young guy want's to see may strongly depend on his companionship and/or day of week – if he attends the cinema with his girl friend on Saturday or with his fellows during the week.



Id	Name	Age
101	John	25
102	Bob	18
103	Alice	27
104	Mary	21

Id	Name	Cost
2	AB17	250.00
3	AB23	299.95
5	XY70	150.00
7	ZZ55	115.50

Id	Name
1	Weekday
2	Weekend
3	Holiday

Figure 3.1.: 3-dimensional model for $User \times Item \times Time$ recommendation space (Adomavicius et al., 2005)[4]

Reduction is one technique to generate contextual recommendations where only the ratings assigned to a given context are considered during computation. So, for example, the cinema visits at week-ends will be calculated separately from those during the week.

[4]cp. Adomavicius et al. (2005), p. 115

Rack et al. (2007) reported about their work on contextual recommendations based on the *AMAYA* recommender (Steglich et al., 2005). AMAYA is a general purpose recommender consisting of four major components – the data adapter, profile manager, a profile broker, and a recommender subsystem (see Figure 3.2) – which form the basis for computing contextual recommendations.

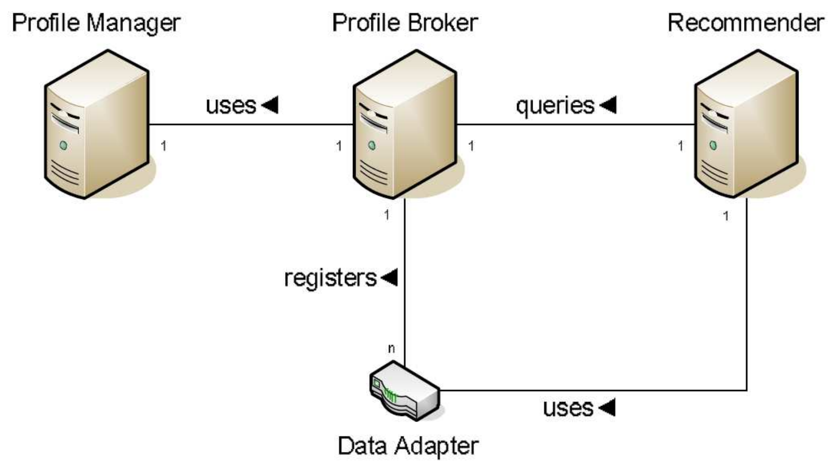


Figure 3.2.: AMAYA recommender system (Rack et al., 2007)[5]

The drawback of AMAYA is, that all contextual situations, like "*being at home*", "*being at work*" have to be modeled explicitly at a profile level. Furthermore, the AMAYA subsystem does not perform the mapping between preferences and situations by itself but leaves this task to well-known techniques like Rocchio's relevance feedback algorithm (Rocchio, 1971).

Another context-aware recommendation approach, developed for a grocery store, is presented by Kim and Kwon (2007). They propose a three step approach based on an ontology containing information about (i) products, (ii) locations, (iii) records, and (iv) consumers. In a first step the preferred items of a user are extracted from the consumer's records. Next, in the second step, the level of information provided to the user is defined. Finally, in the third step, the recommendation of step two is refined according to the consumer's attention.

Beside these attempts to improve the performance of recommendation algorithms research was also done in other important areas, such as security, privacy, trust, or the human-computer interface aspects of recommender systems. Swearingen and Sinha (2001) presented a study concerning HCI aspects of recommender systems as well as suggestions

[5]cp. Rack et al. (2007), p. 446

concerning interaction design for recommender systems (Swearingen and Sinha, 2002). The findings of this study showed that, beside accuracy of recommendations, topics such as trust, transparent and traceable logic of recommendations, detailed information of items, providing not-yet-seen items, etc. had an high impact concerning the acceptance of the tested systems. Although the probands preferred recommendations made by friends, they expressed high satisfaction concerning the performance of recommender systems because of their ability to generate new suggestions out of an enormous set of alternatives. A list of design suggestions for recommender systems, ranging from the usage of specific interactions elements (e.g., continuous rating bars for implementing rating scales) to the support of need driven strategies (e.g., *broaden-my-horizon*[6] recommendation), were provided too.

As recommender systems gained more and more importance in e-commerce, security aspects concerning the prevention of attacks became important, especially in the context of collaborative filtering systems. Some aspects of such attacks, e.g., how to detect attacks, which are the properties of items being attacked, etc. are discussed by Lam and Riedl (2004). The *web-of-trust*, a community driven approach where users explicitly declare others as reliable, was presented by Massa and Avesani (2004). Another approach concerning trust is presented by O'Donovan and Smyth (2005) by introducing trust on an profile and item level.

Due to the fact that many recommender systems are based on user feedback, also *privacy* is an important topic in recommender research. Simply navigating through the Web may result in a massive user profiling where a lot of individual data, ranging from IP address, timing information, browser settings, etc. is collected with far-reaching privacy implications (McSherry and Mironov, 2009). Narayanan and Shmatikov (2008) published a de-anonymization attack where records of the NetFlix Prize[7] dataset where linked with the public profiles of *The Internet Movie Database* (IMDb)[8]. Although this is perhaps not a very disturbing scenario for the video or music domain, privacy concerns may rise when it comes to health records or working contexts. So, for example, a scientist, just writing a paper about a certain topic, may not want to share his currently preferred literature with others.

Baraglia et al. (2006) identified two privacy breaches in common recommender systems. The first breach is based on the fact that recommendations are generated based on similar users. By getting some recommen-

[6]A recommendation strategy supporting the user in exploring his taste was called *broadening-the-user's-horizon* by Swearingen and Sinha (2002).

[7]`http://www.netflixprize.com` as of 11/2008

[8]`http://www.imdb.com` as of 11/2008

dations a malicious user U_{mal} attacking the system knows that there is a group of users having the same preferences as him/herself. Furthermore, this attacking user U_{mal} is able to identify the existence of users with the same preferences by simply stressing his/her profile as long as new recommendations are produced. This detection process is the second breach. To overcome these privacy leaks Baraglia et al. developed $\pi SUGGEST$, a two-tier web recommender system, consisting of a browser plug-in and server component – see Figure 3.3.

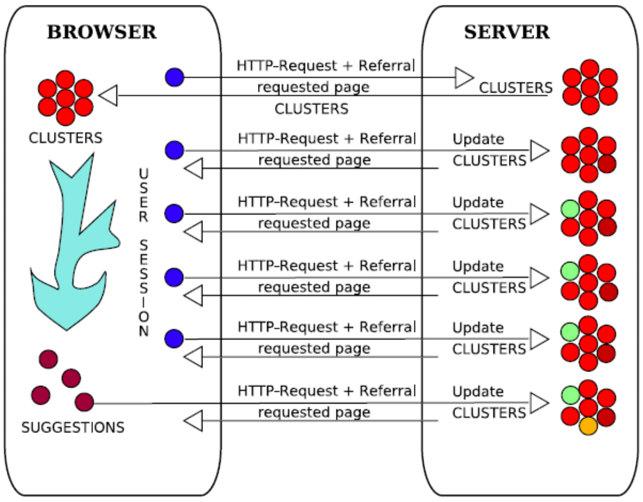


Figure 3.3.: $\pi SUGGEST$ two-tier architecture Baraglia et al. (2006)[9]

While the server side of $\pi SUGGEST$ is updating the knowledge base on-line, the client side browser plug-in is establishing the list of preferred pages.

Other research concerning privacy aspects were published by Ramakrishnan et al. (2001) and Lam and Riedl (2004). The former authors presented a graph-theoretical model demonstrating how information concerning user groups can be derived by observing the system's recommendations. Lam and Riedl suggested the usage of the *value-of-information* (VOI) (Pennock et al., 2000) for deciding which information should be used for personalization or when to stop collecting further user data. The basic idea behind the VOI approach is that different pieces of information may have a different discriminatory power concerning other users. So, for example, personal preferences for a block-buster movie are less useful for a recommender system than preferences for a rare film.

Some suggestions concerning research aspects for developing the next-

[9]cp. Baraglia et al. (2006), p. 560

generation of recommender systems are presented by Adomavicius and Tuzhilin (2005), proposing a deeper understanding of user and items, the support of multi criteria ratings, etc.

3.1.2. Classification

From the several published classifications of recommender algorithms (Resnick and Varian, 1997; Schafer et al., 1999; Terveen and Hill, 2001) the most comprehensive one was presented by Burke (2002) which will be used as our basis for further discussions. Burke identified the following three characteristics for classification:

1. background data: the information that the system has, before producing recommendations
2. input data: the kind of data provided to the recommender systems in order to get recommendations
3. process: an algorithm combing background data with the input to generate recommendations

Based on these definitions Burke identified five different recommendation techniques used in the research community which are presented in Table 3.1.

Additional to these basic recommendation techniques we will also have a closer look to hybrid approaches.

Collaborative Approaches

Collaborative filtering (CF), the most successful technique applied in recommender systems, takes the similarity of users as the basis for generating recommendations. The CF system asks the user to rate presented items – so the knowledge “who likes what” is gathered. When asked for recommendations a list of items, which where high rated by similar users in the past, is generated by the CF system. The similarity between users is calculated based on the rating behavior of common items, by using e.g., the *Pearson correlation coefficient* (Resnick et al., 1994; Shardanand and Maes, 1995; Breese et al., 1998). According to Breese et al. (1998) collaborative approaches can be further divided into *memory-based* and *model-based* techniques, depending on the data and algorithms they use for generating predictions.

[10](cp. from Burke (2002), p. 332)

Technique	**Background**	**Input**	**Process**
Collaborative	user ratings of items	a user is rating an item	finding similar users for a given user and generate an item prediction based on their ratings
Content-based	feature vectors of items	user is rating an item	creation of a classifier that matches the item features to the rating behavior
Demographic	demographic data of users and their item ratings	demographic user information	finding demographically similar users for a given user and generate an item prediction based on their ratings
Utility-based	feature vectors of items	feature based utility function describing the users prefer-ences	defining the rankings of items by applying the util-ity function
Knowledge-based	feature vectors of items and knowledge how these features corresponds to the user's needs	description of the users needs	infer a match between the user's needs and a fea-ture vector

Table 3.1.: Recommender Techniques as classified by Burke (2002)[10]

Memory-based algorithms Memory-based algorithms take all available ratings $r_{v,i}$ – made by user v on an item i – to generate the prediction p for an unknown item j for a given user u. The prediction of an item j for a user u is defined as the weighted sum of the votes of other users (Breese et al., 1998):

$$p_{u,j} = \overline{r_u} + f \sum_{v=1}^{n} w(u,v)(r_{v,j} - \overline{r_v}) \tag{3.1}$$

where $\overline{r_u}$ is the mean rating of user u, f is a factor for doing normalization and $w(u,v)$ is a weight describing the similarity between the two user u and v. As mentioned above, the *Pearson Correlation* is one of the most used similarity measurements based on the common ratings of user u and v. The Pearson correlation is defined as:

$$w(u,v) = \frac{\sum_{i=1}^{n}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i=1}^{n}(r_{u,i} - \overline{r_u})^2 (r_{v,i} - \overline{r_v})^2}} \tag{3.2}$$

where i refers to items, both users u and v have rated in the past.

The *vector similarity*, derived from the field of information retrieval (Heyer et al., 2006), is another widespread function to calculate the similarity between users. In this case the ratings $r_{v,i}$ of users are seen as vectors where the similarity is calculated based on the cosine of the angle, as defined in Definition 3.3 on page 23.

$$w(u,v) = \sum_{i=1}^{n} \frac{r_{u,i} r_{v,i}}{\sqrt{\sum_{k=1}^{o} r_{u,k}^2}\sqrt{\sum_{l=1}^{p} r_{v,l}^2}} \tag{3.3}$$

Using this definition of the vector similarity no distinction between positive and negative ratings is made. All existing ratings are used for similarity calculation and unrated items get a zero value. The dominator in Definition 3.3 is used to normalize the calculated weight and to assure, that the length of a vector is taken into account, avoiding that users with more ratings have a higher similarity by default. (Note: k refers to the ratings of user u and l to the ratings of user v).

The *inverse user frequency* (Breese et al., 1998), another borrowing from information retrieval or data mining (Witten and Frank, 2005), comprises the discriminatory power of an item within the similarity calculation of users. The main idea is, that items being preferred by many users have less significance for similarity relations than rarely chosen ones. This information gain g for discrimination is often defined as $g(i) = \log(\frac{n}{n_i})$, where

n is the number of all users and n_i is the number of users who rated item i. Standard techniques such as *vector similarities* or Pearson correlation can now be improved by applying the *inverse user frequency*. For the vector similarity this is done by transforming the original ratings into adapted ones, by simply multiplying them with their information gain g – see Definition 3.4 – where $r'_{u,i} = g(i) r_{u,i}$.

$$w(u,v) = \sum_{i=1}^{n} \frac{r'_{u,i} r'_{v,i}}{\sqrt{\sum_{k=1}^{o} {r'_{u,k}}^2} \sqrt{\sum_{l=1}^{p} {r'_{v,l}}^2}} \tag{3.4}$$

Another enhancement for the Pearson correlation is called *default voting* where missing ratings are replaced by default values. The main idea behind this strategy is to enlarge the intersection of common ratings – the ratings Pearson is operating on – and so to improve the similarity measurement (Breese et al., 1998; Herlocker et al., 2004).

Case Amplification is another extension to memory-based algorithms where the contributions of more/most similar users are emphasized by amplifying weights $w(u,v)$ – as defined in Definition 3.1 on page 23 – close to 1 and punishing lower values. The adapted weights are calculated as follows:[11]

$$w(u,v)' = \begin{cases} w(u,v)^p & w(u,v) \geq 0 \\ -(-w(u,v)^p) & w(u,v) < 0 \end{cases} \tag{3.5}$$

The techniques discussed above all calculate the similarities of *users* based on their ratings of *items*. In the context of modern recommender systems, with millions of users and items, these approaches suffer from two problem areas: (i) the *sparsity* of data and (ii) *scalability* issues.

- **Sparsity**: Modeling users as vectors of item ratings often leads to very sparse search spaces in many commercial systems.
- **Scalability**: The costs of finding nearest neighbors increases with the growing numbers of users and items.

To face these problems Sarwar et al. (2001) proposed an *item-based collaborative filtering* algorithm where the similarity between two *items* i and j is generated based on the ratings of a set users U having rated both items. Given this set of users U and the target item i – the item, for which a prediction should be generated – the algorithm takes all co-rated items of the users $u \in U$ and calculates the similarities with the target item i.

[11]Breese et al. (1998) used $p = 2.5$ for their experiments.

In the case of *Pearson* correlation the similarity weight w between items i and j can be defined on a user basis having rated i and j as

$$w(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_i})(r_{u,j} - \overline{r_j})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_i})^2 (r_{u,j} - \overline{r_j})^2}} \tag{3.6}$$

where $\overline{r_i}$ denotes the average rating of item i.

When using the cosine-based *vector similarity* an item is seen as a vector of user ratings. The similarity weight between the two items i and j can be adapted as follows:

$$w(i,j) = \frac{\sum_{u \in U} r_{u,i} r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2}\sqrt{\sum_{u \in U} r_{u,j}^2}} \tag{3.7}$$

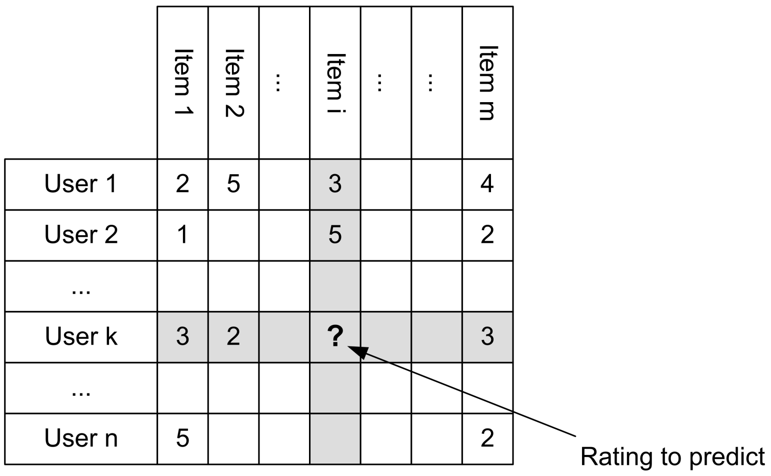
One important drawback of Definition 3.7 is, that the differences in rating scale of the users are not taken into account. This can be avoided by using the adjusted cosine similarity (Sarwar et al., 2001) function which subtracts the average user rating, as defined in Definition 3.8

$$w(i,j) = \frac{\sum_{u \in U}(r_{u,i} - \overline{r_u})(r_{u,j} - \overline{r_u})}{\sqrt{\sum_{u \in U}(r_{u,i} - \overline{r_u})^2}\sqrt{\sum_{u \in U}(r_{u,j} - \overline{r_u})^2}} \tag{3.8}$$

where $\overline{r_u}$ denotes the average rating of user u.

Defining a user-item matrix – see Figure 3.4 – as the basis for generating recommendations, the fundamental difference in the similarity computation between the user-based- and item-based-collaborative approach is, that the user-based technique generates the similarity along the rows of the matrix while the item-based algorithms computes along the columns. By doing so, a similarity relation between items (and not between users) is established, which can be directly used in the on-line recommendation process, once stored in the database. Due to the fact, that in on-line systems the number of users exceed the number of items by far, the scalability of these systems can be improved enormously.

The idea of using items instead of users as the basis for collaborative filtering was picked by Amazon.com for creating their recommender system (Linden et al., 2003) and was further developed by Deshpande and Karypis (2004) creating item-based top-N recommendation algorithms. Empirical comparisons of both approaches were presented by Deshpande and Karypis (2004); Sarwar et al. (2001) where the item-based approaches



	Item 1	Item 2	...	Item i	...	...	Item m
User 1	2	5		3			4
User 2	1			5			2
...							
User k	3	2		?			3
...							
User n	5						2

Rating to predict

Figure 3.4.: User Item Matrix

outperform the best user based techniques concerning computational performance as well as recommendation quality.

Although hardly mentioned in the context of recommender systems, *association rule mining (ARM)* (Agrawal and Srikant, 1994; Park et al., 1995; Witten and Frank, 2005) can also be seen as a memory-based recommender technique, because item-to-item relations (associations) are mined on the bases of historical user actions. Standard *shopping cart* analysis is often performed by applying ARM techniques where items, frequently bought together, are determined. More formally, *association rules* are terms of the form

$$\{X_1, X_2, \ldots, X_n\} \Rightarrow Y \tag{3.9}$$

with the meaning, that if items $X_1, ..., X_n$ (called antecedent) are found in one user transaction (or basket) then there is a certain possibility also to find item Y (called consequent). So, given a huge amount of data consisting of several subsets of items belonging together – often called user transactions or baskets – association rule mining tries to find those item sets which appear at least n times in these subsets.

Sample: In a set of 100.000 baskets the items A and B appear together 2.000 times. Furthermore, within these 2.000 baskets item C can be found 800 times. All in all item C appears in 3.000 baskets. This assumption could be described with the following association rule:

$$\{A, B\} \Rightarrow C \tag{3.10}$$

There are three important figures for controlling association rule mining. The *support* describes the number of baskets containing all items of the rule as defined in Definition 3.11:

$$support = \frac{nrBasketsContaining(antecedent + consequent)}{nrAllBaskets} \tag{3.11}$$

So, for example, the support of the rule $\{A, B\} \Rightarrow C$ is 800 or 0.8% concerning all baskets.

The *confidence* of an association rule describes the probability that the consequent will appear together with a given antecedent and is defined as follows:

$$confidence = \frac{support(antecedent + consequent)}{nrBasketsContaining(antecedent)} \tag{3.12}$$

The confidence of the sample rule $\{A, B\} \Rightarrow C$ defined above is 40% $(0.4 = 800/2.000)$.

The third figure, called *lift*, can be interpreted as the importance of a rule and is defined as the ratio of the *confidence* and the *expected confidence*

$$lift = \frac{confidence}{expectedConfidence} \tag{3.13}$$

where the *expected confidence* is defined as

$$expectedConfidence = \frac{nrBasketsContaining(consequent)}{nrAllBaskets} \tag{3.14}$$

Thus the lift value of the sample rule $\{A, B\} \Rightarrow C$ is 13.33 computed as $0.4/(3.000/100.000) = 13.33$.

In practice association rule mining is a very resource intensive task because the definition of frequent item sets (with different sizes!) is commonly performed on very huge data sources. One of the best known approaches for efficiently solving this problem is the *a-priori algorithm*, developed by Agrawal and Srikant (1994), which works as follows:

1. Starting with a given *support threshold* s (e.g., 1% of all baskets) all items I_k are determined appearing in at least s baskets ($support(I_k) \geq s$). This set of frequent items is called L_1[12].
2. According to the a-priori principle pairs of items of L_1 form candidate set C_2. Those pairs of C_2 having a support value with at least s, form the set L_2

[12] L_i describes a frequent item set with size i

3. The candidate set C_3 is build of those item triples $\{A, B, C\}$ having tuples of the form $\{A, B\}, \{A, C\}, \{B, C\}$ in L_2
4. This procedure can be continued until the sets get empty or a predefined limit (e.g., L_5) is reached.

The idea of using the pairwise occurrence of items as a basis of recommendations was also incorporated in the development of *Slope One*, a family of algorithms for CF, developed by Lemire and Maclachlan (2005). Slope One is a memory-based collaborative algorithm, predicting how a user would rate a given item from other user ratings. Slope One is based on a principle called *popularity differential* between items where the difference of how much better one item is liked than another is defined in a pair-wise manner. This difference, simply computed as the difference of the average ratings of the two items, can further be used to calculate item predictions for users.

Sample: Given two users A and B and two items I and J. Furthermore we assume, that user A rated item I with value 1 and item J with 1.5, while user B only rated item I with 2. Based on this assumptions the prediction of item J should be calculated for user B – see Figure 3.5 on page 28.

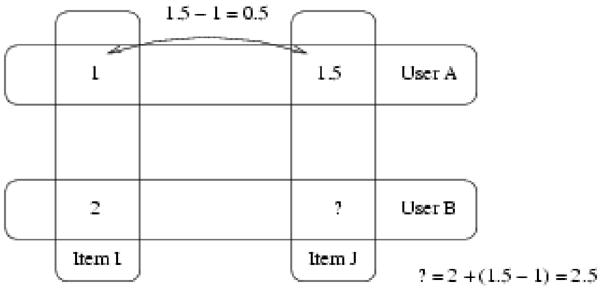


Figure 3.5.: Basis of Slope One Schemes (Lemire and Maclachlan, 2005)[13]

By observing that item J is rated higher than item I by 0.5 points $(0.5 = 1.5 - 1)$, we can predict that user B will give item J a rating of 2.5 $(2 + 0.5 = 2.5)$.

[13]cp. (Lemire and Maclachlan, 2005), p. 2

Model-based algorithms In contrast to memory-based techniques, where all ratings are used to calculate the predictions, model-based algorithms take a subset of these ratings to learn a *model*, used for deriving recommendations (Adomavicius and Tuzhilin, 2005; Breese et al., 1998). Breese et al. (1998) proposes an probabilistic approach for computing the assumed rating for an unobserved item i for a given user u on the basis of what is known about this user u.

$$r_{u,i} = E(r_{u,i}) = \sum_{k=0}^{n} k \cdot Pr(r_{u,i} = k | r_{u,l}, l \in I_u) \tag{3.15}$$

In Definition 3.15 it is assumed, that the ratings have integer values ranging from 0 to n and that the probability expression is the probability that user u will rate item i on the basis of the previously rated items (I_u denotes the set of items user u has just rated). Furthermore, Breese et al. (1998) proposes *cluster models* and *Bayesian networks* as possible approaches for estimating the probability.

In *cluster models* users with the same tastes and preferences are grouped together. So, given a certain cluster/group for a user and assuming that the ratings of the user are independent, the naive Bayesian model can be used. The different parameters of this model, e.g., probabilities of class membership, conditional probabilities of ratings given a class, etc. are learned from the training set of ratings by applying methods capable of learning parameters for models with hidden parameters, e.g., the EM algorithm (Witten and Frank, 2005).

In the case of a *Bayesian network* each item of the domain corresponds to a node in the network and the ratings correspond to the states of these nodes. After applying algorithms for learning Bayesian networks on the training data the resulting networks contains a predecessor-successor relation for each item where the predecessors are used as predictors for a given item.

Seeing collaborative filtering as a classification task is proposed by Billsus and Pazzani (1998) where a complex, machine learning based framework is presented by using different techniques such as neural networks, singular value decomposition etc.

Comparisons of *memory-based* with *model-based* approaches were published by Billsus and Pazzani (1998) and Breese et al. (1998). Both report, that in some cases the model-based approaches outperform the memory-based techniques concerning accuracy[14] of recommendations.

[14]Concerning recommender algorithms, accuracy describes the prediction quality, often measured with metrics such as mean absolute error (MAE).

But it must be mentioned that these results only rely on empirical studies and that no theoretical proof was provided.

There are two main advantages of collaborative filtering compared to other techniques which form the basis of their success in commercial as well as in academic systems: (i) no information about items is necessary, no machine readable profile of items must be provided and (ii) the basic idea of relying on liked items of similar users provide an easy way to implement cross (domain) recommendations. On the one hand, these cross recommendations help the user to explore his/her taste better[15] while on the other hand a more diverse presentation of interesting items may increase the volume of the portal.

The main drawback of the CF approach is known as the *cold start problem* consisting of the *new user* and the *new item* problem (Adomavicius and Tuzhilin, 2005; Schein et al., 2002). New users have no or only a very poor behavior profile (e.g., rated or bought items) thus the definition of similar users is not possible. This problem applies e.g., to the Pearson correlation coefficient, the most popular measure value used in CF systems, which can only be applied to ratings of items, which all concerned users have rated! Further problem areas of CF are the *new item* problem – describing the problem of how to present an item which was not rated before – and the *sparsity* problem, which occurs when the number of rated items is very small in relation to the total item set.

Content-Based Approaches

The concept behind the content-based approach is to suggest items to users which are similar to ones they liked in the past. To compute this similarity a description of the items must be provided in machine readable form of profiles or feature vectors together with the users ratings of these items. Based on this information a variety of machine learning or information filtering techniques (Adomavicius and Tuzhilin, 2005; Pazzani and Billsus, 2007) can be applied to learn the users preferences by identifying the relevant features/attributes of the items provided.

Because *content-based* approaches have their roots in the information retrieval research community (Belkin and Croft, 1992), it is no surprise that most such recommender systems rely on a textual description of items as basis of the items feature vectors. A common technique for creating meaningful item profiles out of text is to create vectors containing the weighted, relevant phrases by applying the *term frequency/inverse document frequency* $(TF \times IDF)$ measure.

[15]named “broaden the user’s horizon” by Swearingen and Sinha (2002)

The weight $w_{t,d}$ in Definition 3.16 describes the importance of a term t for a given document d by using the term's local importance – the *term frequency* (TF) – and it's global discriminatory power, the *inverse document frequency* (IDF). While the TF part of Definition 3.16 describes how important/descriptive a term t is for a given document d by calculating the relative frequency, the IDF defines how often t is used in other documents too:

$$w_{t,d} = TF_{t,d} \times IDF_t = \frac{f_{t,d}}{max_z(f_{z,d})} \log \frac{N}{n_t} \tag{3.16}$$

In Definition 3.16 $f_{t,d}$ describes how often a term t appears in a document d. $max_z(f_{z,d})$ is used for normalization and defines the maximum of all terms in d. Furthermore, N is the number of all documents and n_t is the number of documents containing term t.

Based on such feature vectors of items and the historical ratings of the users several techniques can be applied to learn user profiles concerning preferences of features. Pazzani and Billsus (2007) describe this process of creating a user's preference model as a kind of classification learning, based on the e.g., binary categories *liked/disliked* items.

Decision tree learners such as ID3 or C4.5 (Quinlan, 1986; Witten and Frank, 2005) are well studied, widely used and most successful algorithms to solve this task. Based on a training set of classified feature vectors a decision tree is created by recursively partitioning this set into subgroups until all items of a subgroup belong to one single category (e.g., "liked item"). The appropriate features, with the highest discriminatory power, are identified by calculating the *expected information gain*. The *expected information gain* $E(C; F)$ of a given feature F with respect to the class attribute C can be defined as the reduction of uncertainty about the value of C knowing F, where the uncertainty of C is defined by the *entropy* $H(C)$. Furthermore, the uncertainty of C, knowing F, is defined by the *conditional entropy* of C, $H(C|F)$

$$E(C; F) = H(C) - H(C|F) \tag{3.17}$$

When C and F are discrete variables, having values $\{c_1, ..., c_n\}$, $\{f_1, ..., f_m\}$, the entropy can be defined as

$$H(C) = -\sum_{i=1}^{n} P(C = c_i) \lg_2(P(C = c_i)) \tag{3.18}$$

while the conditional entropy is defined as follows:

$$H(C|F) = -\sum_{j=1}^{m} P(F = f_j)H(C|F = f_j) \tag{3.19}$$

Nearest neighbor methods are also common techniques to solve classification tasks, by defining the n most similar (just labeled) items to a given unlabeled item I. In succession, Item I is associated with the classes of the most similar items. Depending on the kind of data a variety of similarity functions can be applied. While *Euclidean* distance metrics are often used in the case of structured data, the vector space model (based on cosine similarity of vectors, see Definition 3.3 on page 23) is often used in the context of more complex descriptions, e.g., feature vectors based on texts.

Relevance feedback (Rocchio, 1971; Limbu et al., 2006; Pazzani and Billsus, 2007) is an important ability of a user for improving the recommendation results, called *query refinement* in the field of information retrieval. The main idea is to improve the performance of the recommendation/retrieval algorithm incrementally by rating how good the returned items meet the users information need. *Rocchio's* algorithm (Rocchio, 1971) is widely used to implement this feedback driven refinement by recursively adapting an initial query based on relevant and non-relevant feedback. More formally

$$Q_{i+1} = \kappa Q_i + \lambda \sum_{rel} \frac{D_i}{|D_i|} - \mu \sum_{nonrel} \frac{D_i}{|D_i|} \tag{3.20}$$

In this equation Q_i denotes the user's query, at iteration i, D_i the retrieved documents at iteration i, and the weights κ, λ, μ are used to control the different influences, e.g., original query and relevant/non-relevant items. Although empirical experiments demonstrated the power of this algorithm (Rocchio, 1971) no theoretical proof exists (Pazzani and Billsus, 2007) concerning the performance and/or convergence of this approach.

Furthermore, *linear classifiers*, algorithms that learn linear decision boundaries, are appropriate techniques for learning user preferences (Witten and Frank, 2005; Pazzani and Billsus, 2007). For classification purpose a weight vector w, constructed during a training phase, is applied (dot product) to an instance vector to be scored (e.g., vector space model of a document) resulting in a numerical prediction. Algorithms like the *Widrow-Hoff rule* or the *exponentiated gradient (EM)* (Pazzani and Billsus, 2007) can be used to learn vector w. Additionally, also probabilistic methods like the *naive Bayesian classifier* can be applied, especially in the case of text classification.

Like collaborative approaches also content techniques suffer from the *new user problem* because an efficient classifier can only be constructed when having a sufficient number of ratings. Another, and perhaps more important drawback, of the content-based approach is its tendency for *over specialization*. A user, having high rated some items will only get recommendations containing similar ones, a broadening of the user's horizon is not supported. Especially in the context of news recommendation overspecialization becomes a serious problem, because articles telling the same story the user has just read are recommended. The *Daily Learner* (Billsus and Pazzani, 2000), a system recommending daily news, tackles this problem by filtering articles that are too different as well as those being too similar.

Knowledge-Based Approaches

The main characteristic of the *knowledge-based* approach (Towle and Quinn, 2000; Burke, 2000, 2002; Lorenzi, 2007) is the use of functional knowledge – knowledge of how a particular item meets the user's needs – to calculate recommendations. Similar to the *utility-based* approaches no long-term user model is created and no set of historical ratings are required. Any knowledge structure describing the user's needs supporting inference can be used as user profiles, ranging from simple queries (e.g., as used in Google[16]) to more complex descriptions as described by Towle and Quinn (2000). Also the knowledge base used by the recommender system can be provided in many forms. While Google uses links between web pages to determine a popularity and authoritative measure other approaches are using more explicit knowledge or ontologies (Middleton et al., 2002, 2004). *Entree*, a restaurant recommender developed by Burke et al. (1997), uses knowledge of cuisines to determine the similarity of restaurants.

Entree is an example of a FindMe system (Burke et al., 1997; Burke, 1999), which allows a user to navigate through a *web of products*, without being forced to define his/her preferences in advance. FindMe systems have the following characteristics.

1. FindMe systems are example-based supporting a user in easily finding similar products, given an item as a starting point.
2. They support user critiques concerning attributes (or bundles of attributes) of recommended items to find new suggestions.

[16]`http://www.google.com` as of 5/2008

3. Typically the list of recommended items is very comprehensive, containing items ranked according to the expected goal.

So, for example, if a user finds a pretty but too expensive or vanguard restaurant in Entree, he/she can get new suggestions by placing critiques like "Less $" ore "More Traditional".

Beside this *case-based* approach *constraint-based* recommendations form the second well-known technique for implementing knowledge-based recommender systems. Basically, the constraint-based approach tackles the recommendation problem as a constraint satisfaction problem (Felfernig and Burke, 2008). Based on product descriptions, constraints and a set of questions a *recommender knowledge base* is composed containing a pair of feature vectors (U, P) and a set of constraints $(REQ, PROD, FILT)$.

While the features $u_i \in U$ model all possible user requirements (e.g., delivery-period$\leq$4, maximum-price$\leq$100, etc.), the attributes $p_i \in P$ describe the product properties (e.g., price, product-id, etc.). REQ is a set of compatibility constraints req_i describing dependencies between requirements, thus assuring expedient combinations (e.g., $comp_i$: *A VISTA PC requires more than 1 GB memory*). Analogical to REQ, the constraints $prod_i \in PROD$ restrict the number of possible products to a set of useful offers. Filter constraints $filt_i \in FILT$ model relations between requirements and products, often related to marketing strategies (e.g., *PCs with fast CPUs and video cards are reasonable for users who want to play video games*).

Based on the feature vectors and the set of constraints, a constraint-based recommender system computes a solution by assigning the features of (U, P) without violating the given constraints $REQ \cup PROD \cup FILT$.

An advantage of the knowledge-based approach is that no cold start problem exists, because no user ratings are required. On the other hand, an appropriate knowledge base must be provided.

Demographic Approaches

The idea behind the *demographic* approach is using demographic-based user categorizations (e.g., data such as age, gender, occupation, etc.) for recommending items. Similar to the *collaborative* approach “user-to-user” relations are created only based on demographic data – no history of user ratings is needed. Thus, demographic systems rely on user descriptions which must be provided in advance.

The early book recommender system Grundy (Rich, 1979) can be seen as demographic recommender system, because it is based on the usage of predefined *stereotypes*. A stereotype is a collection of frequently occurring characteristics of users, also called *facets*, having a name, a value range and a rating value (between 0 and 1000) describing the degree of certainty. In Figure 3.6 on page 36 some sample stereotypes of Grundy, describing a "sports person" and a "feminist", are presented.

Furthermore, the stereotypes are arranged in an inheritance structure (referred as an directed, acyclic graph having *generalization-of* relations, as presented in Figure 3.7 on page 37, also supporting multiple-inheritance relations.

Stereotypes can be activated by instantiating one of its triggers, an object being associated with a given situation. The information necessary for activating stereotypes is collected throughout an interactive dialog with the user.

The *Lifestyle Finder*, developed by Krulwich (1997), tries to classify a user along a set of predefined clusters, provided by PRIZM[19], using a method called *demographic generalization* as shown in Figure 3.8 on page 37.

In a first step, the available user data is used to determine the best matching clusters. If only one cluster is identified, all the data of that cluster is used as the *broad profile* of the user. If several clusters are appropriate, all similar values of these clusters form a *partial user profile*. For refining profiles subsequently, the best differentiating demographic variable concerning these clusters can be prompted to the user.

In their restaurant recommender Pazzani (1999) avoid the usage of an interactive dialog for gathering demographic data, by using the home pages of the involved users. While the HTML home pages of users who liked a certain restaurant form the positive examples, the negative examples correspond to home pages of users who expressed their disliking. The Winnow algorithm (Littlestone, 1987), a simple text classifier suitable for identifying relevant features in the presence of many attributes, was applied to these samples for determining user profiles.

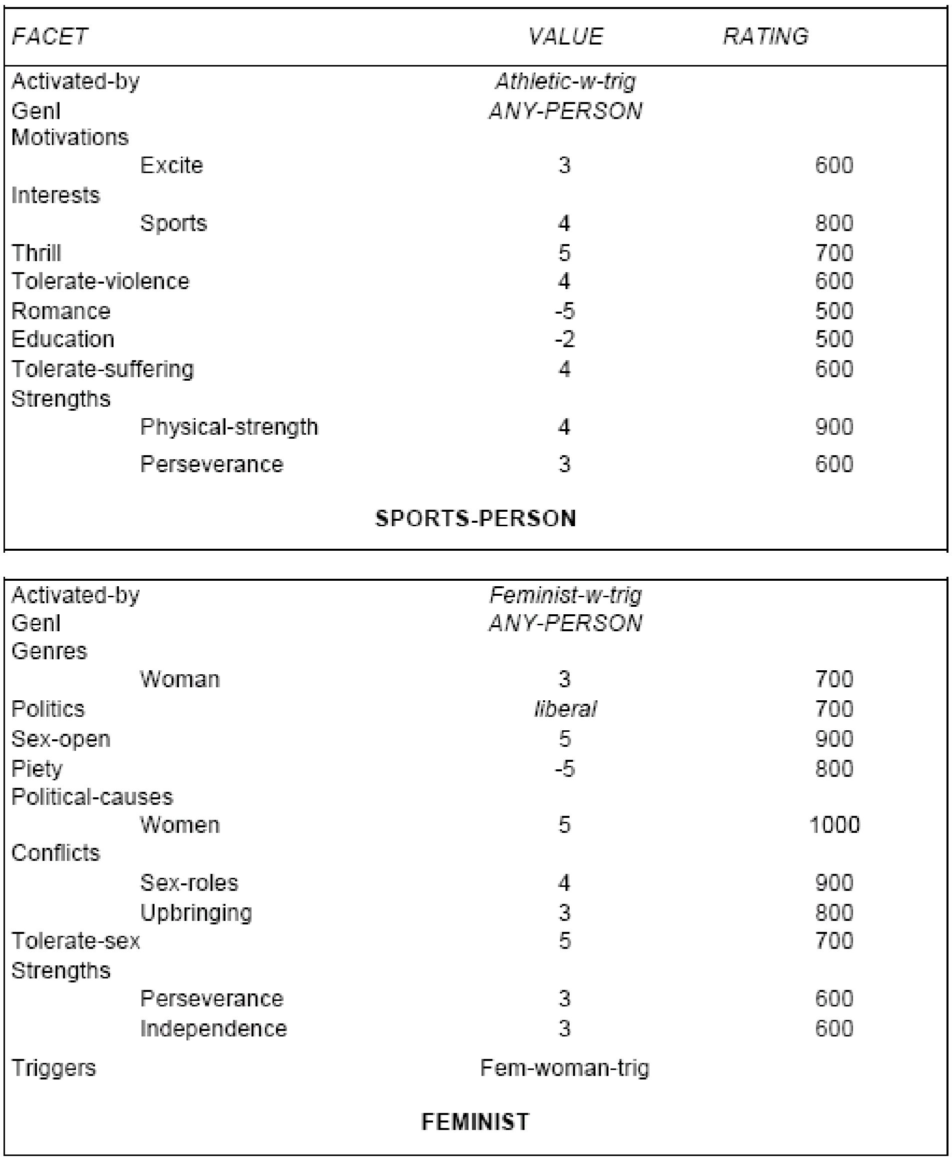
The Winnow algorithm works as follows. Basically, each word x_i is treated as a Boolean feature, thus having the values $0, 1$. Winnow learns

[17]cp. (Rich, 1979), p. 336

[18]cp. (Rich, 1979), p. 337

[19]PRIZM is a marketing research database from Claritas Corporation dividing the population of the United States into 62 clusters based on a set more than 600 variables. Furthermore, each of the demographic clusters provides a mean and deviation value for each variable, indicating the the likelihood of users in this cluster having this attribute.

[20]cp. (Krulwich, 1997), p. 38



FACET		*VALUE*	*RATING*
Activated-by		*Athletic-w-trig*	
Genl		*ANY-PERSON*	
Motivations			
	Excite	3	600
Interests			
	Sports	4	800
Thrill		5	700
Tolerate-violence		4	600
Romance		-5	500
Education		-2	500
Tolerate-suffering		4	600
Strengths			
	Physical-strength	4	900
	Perseverance	3	600

SPORTS-PERSON

Activated-by		*Feminist-w-trig*	
Genl		*ANY-PERSON*	
Genres			
	Woman	3	700
Politics		*liberal*	700
Sex-open		5	900
Piety		-5	800
Political-causes			
	Women	5	1000
Conflicts			
	Sex-roles	4	900
	Upbringing	3	800
Tolerate-sex		5	700
Strengths			
	Perseverance	3	600
	Independence	3	600
Triggers		Fem-woman-trig	

FEMINIST

Figure 3.6.: 2 Sample Stereotypes of Grundy (Rich, 1979)[17]

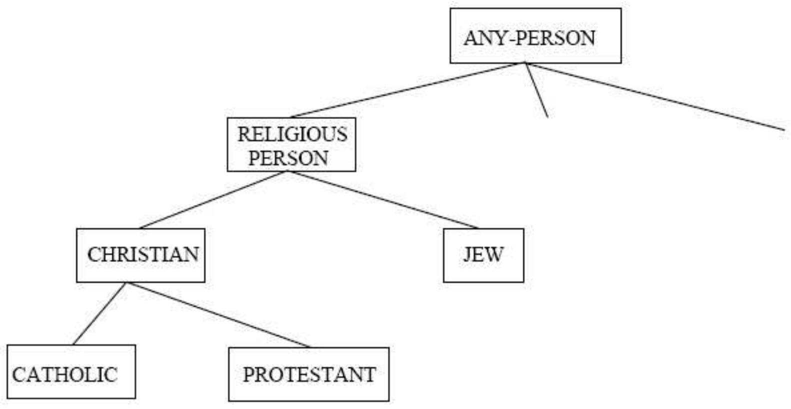


Figure 3.7.: Stereotype Inheritance Tree of Grundy[18]

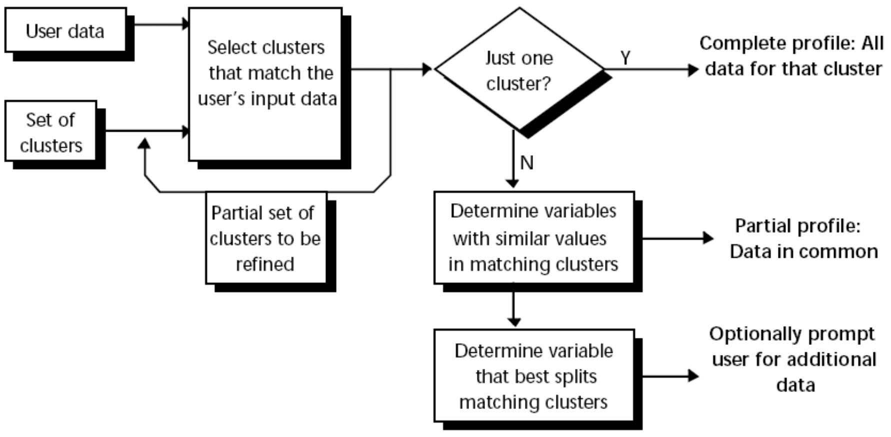

User data
Set of clusters
Select clusters that match the user's input data
Partial set of clusters to be refined
Just one cluster?
Y
Complete profile: All data for that cluster
N
Determine variables with similar values in matching clusters
Partial profile: Data in common
Determine variable that best splits matching clusters
Optionally prompt user for additional data

Figure 3.8.: The Demographic Generalization Process (Krulwich, 1997)[20]

the weight w_i of word x_i to form a linear threshold function, as defined in Definition 3.21, where θ is the threshold.

$$\sum_{i=1}^{n} w_i x_i > \theta \tag{3.21}$$

Initially, all weights w_i are set to 1. Subsequently, each example of the training set is evaluated by finding the sum of the weights w_i of the words x_i that are found in the document (note: if a word is found in the document $x_i = 1$, otherwise $x_i = 0$). If the calculated sum is above the threshold θ and the user liked the document, the example is classified correctly and the calculation stops. Otherwise the weights are adapted in the following way. In the case, that the sum is above the threshold θ but the user disliked the document, the weight associated with each word in the document is divided by a constant value α (commonly α is set to 2). In contrast, if the sum is below the threshold but the document was liked by the user, the weight associated with each word in the document is multiplied by α. Due to the appliance of the factor α, the Winnow algorithm rapidly converges on a set of high weights w_i being assigned to a small number of words.

The evaluations of the different demographic recommender systems, as presented in (Rich, 1979; Krulwich, 1997; Pazzani, 1999), showed good results, although they are commonly outperformed by other approaches like collaborative- or content-based filtering techniques. However, the strength of this approach lies in the minimal information necessary for creating an appropriate user profile forming the basis for generating recommendations.

Utility-Based Approaches

Utility-based techniques generate suggestions based on a utility function matching the (attributes of) items with the user's need. The focus of this approach is not to construct long-term user models, but to filter items in a given context based on an individual utility function. While also non-product attributes such as availability, the vendor's kindness, etc. can be taken into account the main drawback lies in the challenge of how to create an appropriate *utility function*.

The *multi-attribute-utility theory* (MAUT), a major approach in the field of decision analysis (Edwards, 1977), can be used to construct an appropriate utility function. Basically, building and applying such a function consists of the following steps:

1. identify relevant attributes that contribute to a decision
2. define the value ranges of the attributes (e.g., upper- and lower bounds)

3. determine the user's preferences, concerning an attribute
4. evaluate *single-attribute-utility* (SAU) function
5. use all SAUs to construct the multi-attribute utility (MAU) function
6. apply the MAU function to all alternatives and select that with the highest value

More formally, a MAU function can be formulated as follows (Huang, 2008):

$$MAU(u_i, \ldots, u_n) = \sum_{i=1}^{n} w_i u_i \tag{3.22}$$

where n is the number of attributes, w_i is the weight of attribute i with $\sum w_i = 1;\ (0 \leq w_i \leq 1)$ and u_i represents a single attribute utility function for attribute i.

Huang (2008) compared several utility-based recommendation techniques with standard content-based approaches in the context of two different domains, mainly differentiating in the use of nominal and numerical attribute types. While movies, having attributes like genre, actors, company, etc., were chosen to represent the former domain, laptop computers being described by attributes such as CPU speed, memory capacity, etc. were chosen to represent the latter one.

Furthermore, a standard vector space model (VSM), using the cosine correlation (see Definition 3.3 on page 23) as the similarity measure, was used to implement the content-based recommender, the *SMARTER* (Edwards, 1994) method and the *radial basis function networks* (RBFN)(Ghosh and Nag, 2001) approach were used to implement the utility-based recommenders.

SMARTER[21] is an improvement of the *simple multi-attribute rating technique*, provided by (Edwards, 1977), by simplifying the way how weights are determined.

An artificial network, having radial basis functions[22] as it's activation functions, is called a radial basis function networks (RBFN). Such networks are often used for time series prediction and function approximation.

Concerning accuracy, the experiments of Huang (2008) showed, that the VSM approach outperforms the utility-approaches RBFN and SMARTER in the context of nominal attributes, while SMARTER provided the best results in the case of numerical attributes. Also RBFN had an acceptable accuracy (but on a lower level than VSM and RBF), showing only minimal

[21]SMARTER is the abbreviation of simple multi-attribute rating technique exploiting ranks

[22]A function, calculating the distance to some reference point, is called a radial basis function (RBF). The Euclidean distance function is a sample of an RBF.

differences between nominal and numerical attributes.

Schäfer (2001) presented an approach how to apply MAUT for estimating interests of users together with some guidelines on how to create appropriate utility functions were. Tête-à-Tête, an agent-mediated shopping system, presented by Guttman (1998), is a good example of a MAUT based, utility based recommender system.

Hybrid Approaches

Hybrid recommender systems (Burke, 2002) combine two or more recommendation approaches to overcome the specific short-comings of each individual technique. Because of its great success in e-commerce, collaborative approaches are most commonly combined with other techniques (Balabanović and Shoham, 1997; Lemire et al., 2005; Pazzani, 1999).

Burke (2002) identified eight hybridization approaches for implementing a recommender systems which are listed in Table 3.2 on page 41.

In case of the *weighted* approach, the weighted results of different recommendation techniques are used to produce the suggestions for the user. The advantage of such a hybrid is its straight forward process and its simplicity concerning adaptations. However, the implicit assumption of this approach, that the performance of all provided techniques are more or less equal concerning the current domain, is too optimistic and a potential source of problems.

In contrast, the *switching* technique uses different recommendation techniques depending on the current situation. The basic idea is, that different approaches have different performances concerning a given data source and that applying the most suitable technique will lead to the best result. Although a straight forward approach, the definition of an appropriate switching criteria is its pivot adding new complexity to the system.

The *mixed* hybridization technique can be applied in situations, where the simultaneous presentation of the results of all provided recommendation techniques is possible.

In the case of *feature combination* the information of one approach is used as features of a second technique. So, for example, in the context of a collaborative/content-based approach, the collaborative information can be seen as additional features for being used by the content-based approach.

A *cascade* hybrid uses a sequence of recommender strategies to produce a suggestion. While a first recommender creates a crude ranking of the items, the second technique is used to refine this result. One advantage of the cascade approach is, that the second technique can be limited

[23]cp. (Burke, 2002), p. 337

Hybridization Approach	**Description**
Weighted	The suggestions of several recommendation techniques are used are used to construct the recommendation
Switching	A switching between several provided recommendation techniques is performed, based on the current situation
Mixed	The results of the provided recommendation strategies are presented parallel
Feature Combination	Features from different data sources are used together within one recommendation technique
Cascade	Several recommender algorithms are sequenced, so that the successor refines the results of the predecessor
Feature Augmentation	The results of one technique is used as the input of another recommendation approach
Meta-level	The model, learned by one technique, forms the input to another approach

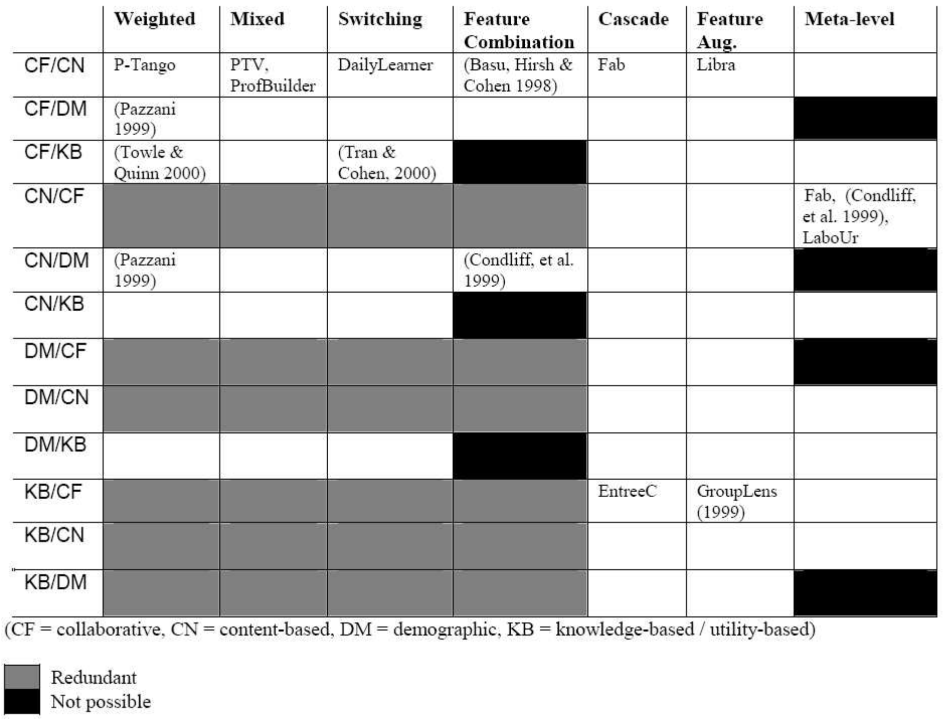
Table 3.2.: Hybridization Techniques (Burke, 2002)[23]

to the insufficiently differentiated items of the first phase.

In case of the *feature augmentation* one recommender technique is used to produce classifications or ratings as the input of the second approach. In contrast to the cascade approach, the second technique is not only applied to the poorly classified items of the first stage. The difference from the feature combination is that not only raw data is used in the second phase.

A *meta-level* hybrid is a recommender system, where the hybridization is implemented on a model layer. A model is learned by a first recommender which forms the input for the second approach. This technique differs from the feature augmentation, where only features, and not the entire model, is used for the second recommender.

Furthermore, Burke (2002) presents an interesting chart, where all possible combinations of hybridizations are shown – see Figure 3.9 on page 42. The white boxes describe possible/useful hybridization techniques filled with examples, if available. While gray areas refer to redundant solutions black boxes are used to define impossible or useless combinations. Where appropriate, also the sequences of the hybridization is taken into account. This is especially true for the cascade, augmentation and the meta-level approach which are all order sensitive.



	Weighted	Mixed	Switching	Feature Combination	Cascade	Feature Aug.	Meta-level
CF/CN	P-Tango	PTV, ProfBuilder	DailyLearner	(Basu, Hirsh & Cohen 1998)	Fab	Libra	
CF/DM	(Pazzani 1999)						■
CF/KB	(Towle & Quinn 2000)		(Tran & Cohen, 2000)	■			
CN/CF	▒	▒	▒	▒			Fab, (Condliff, et al. 1999), LaboUr
CN/DM	(Pazzani 1999)			(Condliff, et al. 1999)			■
CN/KB				■			
DM/CF	▒	▒	▒	▒			■
DM/CN	▒	▒	▒	▒			
DM/KB				■			
KB/CF	▒	▒	▒	▒	EntreeC	GroupLens (1999)	
KB/CN	▒	▒	▒	▒			
KB/DM	▒	▒	▒	▒			■

(CF = collaborative, CN = content-based, DM = demographic, KB = knowledge-based / utility-based)

▒ Redundant
■ Not possible

Figure 3.9.: Possible and Actual Hybrids (Burke, 2002)[24]

Another way of hybridization is to incorporate other techniques in the recommendation process. With the rise of the *Semantic Web*[25] although some research efforts were directed to the combination of *ontologies* and recommender systems.

Having its origin in *philosophy* ontologies were successfully adapted by *Artificial Intelligence* researchers in the mid-1970 for building powerful systems e.g., MYCIN (Shortliffe, 1974). Following the definition of Gruber (1993) an ontology can be defined as an "*explicit specification of a conceptualization*" thus providing a shared vocabulary which can be used for modeling domains. Such a model consists of objects/concepts of certain types, their properties and relations. Having its strength in modeling domain knowledge, ontologies suffer from the *knowledge acquisition*

[24]cp. (Burke, 2002), p. 340
[25]based on concepts of Berners-Lee et al. (2001)

problem, describing the bottleneck of initialization, updating, refining, etc., of the knowledge base.

Middleton et al. (2002) present the combination of a recommender system with an ontology to solve the *cold start* as well as the *knowledge acquisition* problem in the domain of on-line research papers. *Quickstep* (Middleton et al., 2001), a hybrid recommender system for suggesting on-line papers to researchers, was combined with *OntoCoPI* (Alani et al., 2002), an AKT[26] ontology based community of practice identifier. The basic components and their interrelations are shown in Figure 3.10.

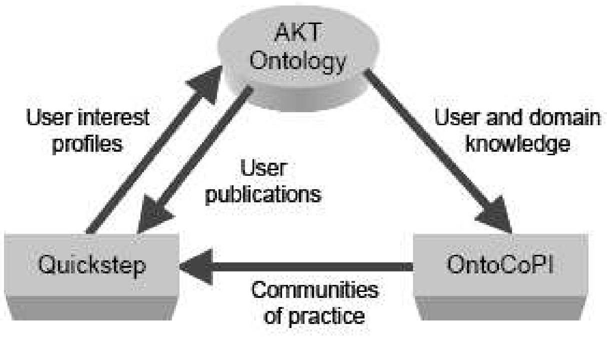


Figure 3.10.: Recommender System and Ontology (Middleton et al., 2002)[27]

In the system's start-up phase the ontology feeds the recommender system a list of publications for each registered user. These papers are then correlated with Quickstep's classified data base compiling historical interest profiles of the users, thus overcoming the cold-start problem of the recommender.

When a new user is joining the system the ontology provides his/her historical publications and OntoCoPI derives a ranked list of similar users. In a next step, the initial profile of the new user is constructed on the basis of his/her publications and the profiles of similar users. This algorithm is called the *new-user* algorithm. The recommender system performs user profile refinement on a daily basis which are asserted into the ontology, thus solving the interest acquisition problem[28].

The *SemTree*, presented by Bouza et al. (2008), uses an ontology to improve the creation process of a decision tree (Quinlan, 1986), by also

[26]`http://www.aktors.org/akt` as of 4/2009

[27]cp. (Middleton et al., 2002), p. 5

[28]The problem gathering the changing interests/preferences of users is called interest acquisition problem – see (Middleton et al., 2002), p. 2

considering superclass relations of item attributes (features). Basically SemTree implements a classical recommender approach, where a user has to rate a couple of items – described by a set of features – in advance. Based on the user's ratings and the feature vectors a decision tree is learned for determining the user's preferences. The new aspect of SemTree is the consideration of superclass relations of features during the construction of the decision tree. As a result, a node of the decision tree can be constructed based on a feature of an item as well as of an appropriate superclass.

In their position paper Buriano et al. (2006) analyze a variety of applications of ontologies in the context of mobile context-aware recommender systems. Although not describing a running system the ideas of applying ontologies are developed along the following features/components of recommender systems:

- context features and candidate items
- output representation
- representation of the recommendation process
- representation of functional modules

Using ontologies for describing contexts and candidate items has the advantage that this data can easily be enriched or augmented with information based on reasoning mechanisms. So, for example, high level categorizations such as "*Important Meeting*", "*Business Trip*" or "*Entertainment Item/Service*" can be generated to enrich the system.

Using a semantic representation of recommended items provides the basis of an unambiguous merging of suggestions, coming from different sources, into a single recommendation list. This is an important scenario in the context of intelligent software agents, where different recommendation services are used to solve a given problem. Ontologies can also be used to describe the recommendation process or algorithm thus supporting the user in better assessing the reliability of recommendations. Furthermore, extending the semantic representation from algorithms to complete recommender systems provides the opportunity that such functional modules can be used in the context of the Semantic Web.

3.1.3. Example Systems

In this section some examples of recommender systems are presented. These systems were chosen because of their importance for recommender research and their impact on the work of the author of this thesis.

Daily Learner: The Daily Learner (Billsus and Pazzani, 2000), is a content-based recommender system suggesting daily news to the reader. A user can browse through a variety of news categories and access the full text by selecting an appropriate headline. The user can give feedback by submitting a positive or negative rating by submitting that an article was *interesting* or *not interesting*. Furthermore, the user can inform the system, that he/she *already knows* something about a given event or that he/she wants to read *more* about it. All these ratings together with the news are taken as the input for a content-based learning algorithm to construct the user's interests profile which can be used as the basis for generating a personalized news program. On an algorithmic level, overspecialization is tackled by filtering articles that are too different as well as those, being too similar to the currently read news story.

Entree/EntreeC: Entree (Burke et al., 1997) is a restaurant recommender using knowledge of cuisines to determine the similarity of restaurants. Entree is an early example of a *FindMe* system (Burke et al., 1997; Burke, 1999), which allows a user to navigate through a *web of products*, without being forced to define his/her preferences in advance. As mentioned before – see Section 3.1.2 on page 33 – FindMe systems are example based which support a user in easily finding similar products. Given an item as a starting point, the user submits critiques concerning specific attributes (or bundles of attributes) of recommended items which are used by the system to find appropriate suggestions. So, for example, if a user finds a pretty but too expensive or vanguard restaurant in Entree, he/she can get new suggestions by placing critiques like "Less $" or "More Traditional". EntreeC (Burke, 2002) is the hybrid version and further development of Entree, combining knowledge-based approaches with collaborative filtering using a cascading hybridization approach.

Fab: Fab (Balabanović and Shoham, 1997) is a hybrid recommender system[29] suggesting web pages to users based on a combination of content-based and collaborative filtering techniques. In Fab recommending pages was performed as a two step process: First appropriate pages corresponding to specific topics were *collected* and stored in a database followed by a *selection* process for particular users. Both tasks were performed by a set of independent agents, referred as *collection agents* and *selection agents* respectively, each having distinct profiles. Pages found by the collection agents are send to a central router which forwards them to those users having appropriate profiles. The personal selection agent

[29]Fab was part of the Stanford University digital library

of a user additionally applies some individual filters, e.g., suppressing already seen pages. Furthermore, the selection agent is responsible for collecting the user's feedback, necessary for refining the user's profiles as well as the topic profiles of the collection agents. Beside this refinement, high rated pages of users are directly forwarded as suggestions to similar users. Especially the topic related nature of the collection agents provide a good basis for scalability, because normally there are much less topics than users. Furthermore, the topic adaptive behavior of the system can also easily be implemented by deleting collection agents with poor or no feedback and by splitting high rated ones. Based on the hybrid nature of the system, the collaborative features can also be extended to users having rated similar pages instead of identical ones.

Grundy: Grundy (Rich, 1979), is a book recommender system based on user stereotypes intended to suggest novels to users/visitors of a library and can be seen as an early sample of a demographic recommender system. The backbone of Grundy is a collection of user stereotypes, arranged as an acyclic directed graph, containing weighted characteristics (also called *facets*) of users being associated with. Typically, a facet of a stereotype is a triple having a *name*, a *value* and a *rating* describing its confidence. So, for example, the stereotype *SPORTS-PERSON* has a facet *Interests-Sports* with a value of 4 (range: -5 to $+5$) and a confidence rating of 800 (range: $0 \cdots 1000$) in Grundy. The different stereotypes are activated by a set of triggers, e.g., during the dialog session with the user required for creating the user model. A specific user model, called *user synopsis* (USS), is established by combining information directly derived from user actions with predications referred from the activated stereotypes. Once a USS is constructed, recommendations are generated by depicting novels best matching USS facets with high values. Beside individual user models, user feedback is also used to adapt the domain model by changing the facet values and ratings of stereotypes according to the positive or negative feedback of users.

GroupLens. GroupLens, developed by Konstan et al. (1997), is a collaborative filtering system for supporting Usenet[30] users, working together, in finding relevant articles. For a given user, GroupLens selects an appropriate group of other users, acting as *personal moderators* for a given category of news (John et al., 1997). These moderators were determined by selecting those users with similar assessments on articles red in the past. Furthermore, also privacy aspects were concerned by allowing a user to submit feedback using a pseudonym. Access to the GroupLens

[30]Usenet is a Internet based discussion system, developed by T. Truscott and J. Ellis in 1979

functionality was gained by providing a specific client application. On the server side, user ratings together with a correlation matrix were stored in a database.

Lifestyle Finder: The Lifestyle Finder, developed by Krulwich (1997), is a demographic recommender system designed for suggesting appropriate Web pages to Internet users. The system is based upon a set of 62 predefined demographic clusters, derived from PRIZM, a commercially available database. This database stores its information in more than 600 variables, each referring to a specific lifestyle characteristic or purchase activity. Furthermore, each of the predefined clusters provide a mean and standard deviation for each variable, describing the probability of people in this cluster having this characteristic. The association of a user to one or more of these clusters is performed by a process, called *demographic generalization*. The basic idea behind this process is to determine either a broad profile based on one (the best matching) cluster or to establish a partial profile, based on the best matching variables of a list of clusters. By determining and prompting the variables best splitting a set of matching clusters, further refinement of the profiling process can be guaranteed on a minimal amount of user involvement. An embodiment of the system as an agent, named *Waldo the Web Wizard*, was chosen for a more appealing and entertaining implementation of the dialog process with the user. Different data areas were abstracted to formulate several high-level question-answer pairs, each providing five to six answer alternatives presented in a graphical way.

Netflix: Netflix, a movie rental platform[31], is mentioned here not as a sample of a recommendation engine, but because of its great impact on collaborative filtering research due to the promised *NetFlix Prize*[32]. In October 2006 Netflix opened a research competition, called Netflix Prize, by promising an award of $1.000.000, to the first recommender development team being able to implement a system that can improve the performance/quality of the currently used system *Cinematch*[33] by at least 10%. As a training set for the algorithms, Netflix released a database containing 100 million ratings, submitted by about 480.000 users over a set of 18.000 movies. 3 million ratings from the same subscribers were withheld, forming the test set. Furthermore, the *root squared mean error* (Herlocker et al., 2004) was chosen as the test metric – see (Bell and Koren, 2007; Bennett and Lanning, 2007). This enormous training set, together with the fact of an extremly sparse user-item matrix (about 99%

[31]`http://www.netflix.com` as of 7/2009
[32]`http://www.netflixprize.com` as of 7/2009
[33]Cinematch is a collaborative filtering system, incorporating a variant of the Pearson correlation.

of all user-movie pairs had no ratings!) and last but not least the awarded prize challanged a large number of researchers and developer. On July 2009, over 5,100 team from more than 185 countries had registered for that competition. On the 26th of July, 2009 the contest was closed, because two teams – *The Ensemble*[34] and *Bellkor's Pragmatic Chaos*[35] – were able to beat the 10% threshhold. Interestingly, the performance improvements of both teams were based on a better understanding how people behave, derived from extensise analysis of the provided data set. While the former team accounted for the different usage of the rating scale by different persons (Potter, 2008), the latter team focussed on temporal effects (Koren, 2009; Koren et al., 2009), such as that people tend to rate older movies higher! In my opinion, the great merit of this award was to guide more research attention to real world scenarios containing vast amounts of data.

PHOAKS: PHOAKS[36], developed by Terveen et al. (1997), is a collaborative-filtering system recommending Web resources on the basis of Usenet Netnews postings. PHOAKS analyzes messages for the occurrence of Web links (URLs) and treats each mentioning as a recommendation if several tests are passed, e.g., by checking if the URL is not part of an advertisement. Remarkable distinction criteria from other systems are the two design principles, *role specialization* and *reuse*, which were realized in PHOAKS. In contrast to other systems where role uniformity of all users is assumed, PHOAKS distinguishes between recommendation providers and recipients. Furthermore, existing recommendations (postings of links) are reused, so no extra effort for generating recommendations is necessary. The generation of recommendation is performed by three main processes: *Search*, *Categorization* and *Disposition*. Messages are *searched* for special patterns like "http://" and their contextual surroundings. During categorization, the patterns found are classified and finally – the disposition phase – are processed in an appropriate way, e.g., storing the information in a database or fetching the content of a Web link.

Tapestry: Tapestry, an early collaborative filtering system created by Goldberg et al. (1992), was designed to support small groups of people, working together, in managing the problem of information overload. Tapestry supports filtering of a variety of information streams including e-mails, Usenet news, etc. Regarding this information, messages can be rated or evaluated by users on a textual basis. Furthermore, users

[34] `http://www.the-ensemble.com/` as of 10/2009

[35] `http://www.research.att.com/~volinsky/netflix/bpc.html` as of 10/2009

[36] PHOAKS is the abbreviation of People Helping One Another Know Stuff; `http://www.cs.indiana.edu/~sithakur/l542\_p3` as of 7/2009

can define a list of preferred evaluators. To form a request in Tapestry, the user can combine keyword with subjective criteria. So, for example, “Give me all news containing the word *computers* that Steve has evaluated containing the word *excellent*” is a valid request in Tapestry. Tapestry showed an appropriate performance in communities with focused interests.

3.2. Generic User Modeling

User modeling is the attempt to construct models of human behavior in a computer environment for exploiting them in assisting the user in certain computing tasks. Furthermore, user modeling is a cross-disciplinary research area not trying to imitate the behavior of a user – as done in many other research fields of Artificial Intelligence – but affording a system to “*understand*” a user’s preferences, goals or even plans. The representation of a user (e.g., his/her desires, preferences, plans, etc.) within a system is called *user model* while the system hosting and exploiting this model is called *user modeling system*.

3.2.1. Historical Survey

User modeling as a separate field of research emerged from the area of natural language dialog systems and can be traced back to the late 70ties where pioneering work concerning user adaptive systems were performed e.g., by Perrault et al. (1978) in their work concerning speech acts for speech recognition among agents. Rich (1979) proposed the evocation of stereotypes – a cluster of characteristics describing (sub)groups of users – as an appropriate mechanism for quickly creating user models only based on a small amount of information. Based on these research results numerous user adaptive applications in many different domains were developed in succession. So in the 1980s a series of task oriented dialog systems appeared assisting the user in different situations, like:

- *Grundy*, a system recommending novels to users of a library (Rich, 1979, 1983)
- *XTRA*, a system supporting users to complete their income tax form (Allgayer et al., 1989)
- *HAM-ANS*, a system simulating a hotel manager trying to rent all available rooms (Jameson et al., 1980; von Hahn et al., 1982)

Although many user modeling systems are available on-line now (in most cases in the form of personalized portals like Amazon.com[37]) the benefit for the user is limited by the fact that these systems do not share a common user model. Many complex tasks, especially in the context of knowledge workers, consist of a series of actions performed on different systems (e.g., writing a paper, preparing a presentation, etc.). Based on the user's need being supported best in all the single actions and boosted by the shift of the computing paradigm toward service architectures (e.g., software as a service) in recent years some research on cross-system personalization were performed. Niederée et al. (2004) present a multi-dimensional, context sensitive user model, called *Unified User Context Model* (UUCM), supporting cross system personalization. Furthermore, in 2004 the $W3C$ published the *Composite Capability/Preference Profiles* (CC/PP) (W3C, 2004a) specification for defining the capabilities and preferences of user agents. CC/PP – an extension of the *Resource Description Framework* (RDF) (W3C, 2004b) – is designed to support the tailoring and adaptation of content to specific end-devices.

Over the years of development, many attributes were identified of being essential for generic user modeling systems. While in (Kobsa, 1995) a list of frequently found services of user modeling shell systems is presented, in (Kobsa, 2001) a set of characteristics of generic user modeling systems is enumerated. Theses required attributes are:

- *Generality*: A user modeling system should be applicable in any domain.
- *Expressiveness*: A user model should be able to express a variety of assumptions of the user, including goals, preferences, beliefs, etc. and to perform appropriate reasoning upon that data. This requirement, derived from the Artificial Intelligence roots of user modeling, lost importance during the rise of e-commerce applications, where often only simple user models (e.g., interaction of the user with the system) were available.
- *Rapid Adaptation*: A quick adaptation of the system to the user is necessary. This is especially essential for commercial systems to convince occasional customers to return again.
- *Open Architecture*: A generic user modeling system should provide interfaces supporting a seamless integration with legacy systems concerning extending the functionality and the exchange of data.
- *Privacy and Security Aspects*: When storing and processing personalized data the prevention of abuse is always a critical issue.

[37]`http://www.amazon.com`as of 5/2009

Furthermore, also common technical requirements such as scalability, performance, etc. should be fulfilled by generic user modeling systems. More information concerning early user adaptive applications are presented by Kobsa and Wahlster (1989) and Kobsa (2001).

The two requirements – generality and expressiveness – were proposed very early in the history of generic user modeling systems and can therefore be seen as their basic characteristics being extended with further attributes (see above) over time. Following Kobsa (1990, 2001) the historical development of generic user modeling systems can be further classified into *user modeling shell systems* and *user modeling servers* due to their architectural differences.

3.2.2. Classification

User Modeling Shell Systems

From (user adaptive) application developer's view a user modeling shell system can be seen as domain independent module being configured/adapted and integrated. The developer formulates and stores the specific domain model in this shell system and integrates it in his/her application. By doing so the user modeling shell systems became an integral part of the application itself – all the available services are only provided to this single application.

The term *shell system* – Kobsa (1990) claims being the first author using this terminology in this context – was borrowed from the research field of expert systems where the experiences and findings of the medical expert system *MYCIN* (Shortliffe, 1976) formed the basis for the development of *EMYCIN*, an domain agnostic, adaptable expert system. Examples of user modeling shell systems are: *GUMS* (Finin and Drager, 1986), *TAGUS* (Paiva and Self, 1995), *BGP-MS*[38] (Kobsa and Pohl, 1995), etc.

User Modeling Servers

In contrast to user modeling shell systems user modeling servers provide the modeling functionality as independent services not being integrated in the application itself. From an (user adaptive) application developer's view a user modeling server acts like a self-contained application (e.g., like a database engine) offering services to a set of clients. Due to that client-server architecture (i) user models can be (re)used by multiple

[38]It is worth being noted that the BGP-MS system is mentioned as a user modeling server example in a later publication by Kobsa himself (Kobsa, 2001)

applications too, (ii) the computational load of handling the model can easily be shifted from the application to dedicated server and (iii) all model relevant tasks like consistency checks, etc. can be performed more easily (Kobsa, 2001).

3.2.3. Example Systems

A short overview of some examples of user modeling systems are presented in this section. A more detailed description is presented by Kobsa (2001).

BGP-MS. BGP-MS (Kobsa, 2001) is a user modeling server where assumptions about users and stereotypical assumptions about user groups are represented in first-order predicates. Different views of the user model can be defined by using hierarchically ordered partitions containing different assumptions as well as stereotypes. Furthermore, BGP-MS can also be used as a network server with multi-user and multi-application capabilities.

CUMULATE: CUMULATE (Brusilovsky and Maybury, 2002) is a user modeling server providing personalization to a student educational system. The activities of students interacting with different systems are collected and used for creating the individual user models by inferring their learning characteristics. Inferences are performed by a set of agents, each responsible for a specific property of the model e.g., the motivation or knowledge level concerning courses. Furthermore, domain ontologies are used as the basis for describing the knowledge-level of students concerning topics.

DOPPELGÄNGER. DOPPELGÄNGER (Orwant, 1994) is a user modeling server gathering data about users from hardware sensors as well as from software systems. User communities are established by applying unsupervised clustering techniques acting as stereotypes. Unlike other modeling servers, DOPPELGÄNGER uses a probabilistic approach to model *is-element-of* relations to stereotypes.

GUMS: Developed by Finin and Drager (1986), GUMS is a user modeling shell system allowing the definition of simple stereotype hierarchies. Prolog clauses are used to describe stereotype memberships as well as rules for reasoning. All predictions concerning users and stereotypes together with the predicates describing the inference processes are stored in one database.

TAGUS: TAGUS (Paiva and Self, 1995) is a user modeling shell system, based on stereotype hierarchies, representing assumptions about users in first-order logic. Beside an inference mechanism TAGUS also provides a truth maintenance system as well as a diagnose subsystem including a library of misconceptions. Furthermore, a user simulation producing predictions based on the current model, together with a diagnoses of unexpected behavior, is provided too.

UMT: Developed by Brajnik and Tasso (1994), UMT is a user modeling shell system supporting the definition of hierarchically ordered user stereotypes. Moreover, rules for user model inferences and the detection of contradictions can be defined. Information about a user, provided by the application, is handled as an invariant premise or assumption. Receiving new information concerning a user may lead to the activation of some stereotypes resulting in the extension of the user profile with the content of these stereotypes. Inference rules are applied to the set of assumptions and premises to determine dependencies. Found contradictions are solved by applying different resolution strategies.

UM. UM (Kay, 1990) is another sample of a user modeling shell system coming as a toolkit for user modeling representing assumptions about the user such as standard characteristics, preferences, knowledge, etc. in form of key-value pairs. Information objects are adorned with a set of evidences (containing information about the data source and a time stamp) concerning its truth or falsehood.

UMS. UMS, developed by (Kobsa and Fink, 2006), is a user modeling server based on the Lightweight Directory Access Protocol (LDAP). Due to the use of LDAP, an international standard well adopted by the industry, the user models in UMS can easily be shared, distributed and synchronized across networks. User modeling modules are designed as internal clients of the directory component, easily plug-able into the system.

3.3. Contributions to the Research Question

In this chapter a variety of different strategies for implementing personalization solutions were presented. Most of these approaches were successfully applied to certain problems thus demonstrating the power of personalization techniques.

However, our main criticism is that these approaches do not use all the information available, even in situations with minimal information[39]. This is perhaps not a big issue in classical e-commerce applications based on standard web interfaces, but a problem in domains with limited access or typical walk-in customers (as it is the case in m-commerce). For example, typical collaborative-approaches such as the Pearson correlation (Breese et al., 1998) or Slope One (Lemire and Maclachlan, 2005) only rely on user ratings. Although improvements such as *inverse user frequency* (Breese et al., 1998; Adomavicius and Tuzhilin, 2005) make better use of the available information (e.g., by considering the popularity of an item) no further information, such as the size and overlap of rating sets, though easily available, is used.

Throughout this thesis, we will show how existing information can be used to improve recommendation algorithms as well as user and item profiles.

[39]consisting of a user identifier, an item identifier and a performed action

4. Adaptive Personalization: A Bird's Eye View

In this chapter an overview of the *Adaptive Personalization* concept is presented, focusing on the big picture. Due to its origin, the creation of a personalization component for a commercial music download platform, many explanations refer to the music domain, although the concept itself is domain independent. Basically, the *Adaptive Personalization* is a hybrid, self adapting personalization approach, combining the advantages of collaborative- and item-based filtering approaches. The cornerstones of this concept are:

- a highly sophisticated profile system, supporting contextual views
- a well defined set of recommendation strategies
- new collaborative-filtering and k-nearest neighbor algorithms for best implementing the strategies
- a flexible architecture supporting large scale, real world scenarios

4.1. Adaptive Profile Model

In this section a short introduction to the profile model of our approach – used for modeling users and items – is presented. A more detailed discussion is presented in Chapter 6.

4.1.1. Modeling the User

Needs can be distinguished in well defined needs where the user is able to characterize an appropriate means of satisfaction, and ill defined needs where the user does not know how to satisfy or even how to define them. Furthermore, users are normally not isolated during the usage of personalization systems; therefore, an effect is created within the relevant community. These effects can be very multifarious, ranging from deliberate interactions such as the placement of ratings or recommendations to be a (passive) example for other users or data mining algorithms. These considerations led to a multi layer model, inspired by the work of Luft and Ingham

(1955), where each user is represented by a profile consisting of three different views:

- self-assessment
- system observations
- community-assessment (assessment by others)

Each view is implemented by a vector of attributes best describing the model to represent.

4.1.2. Modeling the Items

A similar structure is also used for the items, where the affiliation of items to certain clusters (or genres) – being used to group items based on certain characteristics – is modeled with three different views:

- the assessment of the domain expert
- the assessment of the (user) community
- cluster affiliations calculated by appropriate classifier systems, if available.

Genres such as “Rock”, “Jazz”, etc. in the music domain, “Thriller” or “Science-Fiction” in literature are samples of such clusters.

4.1.3. Clustering the Problem Domain

The basic idea behind the clustering of the problem domain is to introduce domain specific knowledge on a conceptual level thus improving the performance of the recommender system. A cluster can be seen as a named container for items sharing commonalities in some respect. Associations can be defined between clusters expressing some special relations such as *is-sub-cluster-of*, *is-similar-to*, etc. The number and kind of associations is mainly guided by the problem domain. Clusters can be created manually by the administrator or can be created by some cluster analyzing programs operating on the item set. The assignment *Object – Cluster* can be made by hand, with the help of some classifier systems or simply by using existing information about domain specific clustering on the item level.

4.1.4. Adaptive Capabilities

The self adapting or *learning* behavior of the *Adaptive Personalization* approach is realized by using data mining and instance-based learning algorithms as well as a simple profile merging strategy. This *adaptive behavior* is implemented on three different levels:

1. The individual level, where profiles of users are permanently refined based on implicit (e.g. navigation observation) and explicit feedback (e.g. ratings, buying behavior, etc.). Beside machine learning approaches such as decision tress (Quinlan, 1986) for learning item attribute preferences, also a user profile refinement, based on merging strategies with item profiles is provided, thus supporting a simple learning strategy also available in contexts with little user feedback.

2. The collaborative level, where the community ratings and/or classifications of items improve the recommendation quality. Furthermore, the ongoing refinement of profiles leads to an improved "similarity" relation among users and items.

3. The statistic level, where data mining algorithms are applied (e.g. association rules) to generate new recommendations. Because these algorithms also operate on data based on user-behavior (e.g. shopping history, compilations of favored items, etc.) the quality will improve over time.

4.2. Recommendation Strategies

Another key factor of an efficient personalization system is to provide an appropriate set of recommendation strategies. Based on the suggestions made by Swearingen and Sinha (2002), the following strategies were incorporated in the *Adaptive Personalization* approach:

1. Reminder recommendations

2. More like this recommendations

3. Hot Item recommendations

4. Broaden my horizon recommendations

5. Similar users like

6. Related Items

Reminder recommendation should help the user not to forget or oversee some important items he/she was willing to use or buy in the past. Theses recommendations are based on a list which is maintained by the user (for example, think of a *black board* feature or a simple *remind-me-later!* feature).

More like this recommendation – probably the most common one – should help the user to find similar items starting from a given one. Hereby similarity is defined by a similarity function based on a set of item attributes. Available user preferences are only used to filter results (e.g. suppressing disliked items). Concerning the approach at hand, common genre affiliations and sound similarity, derived from audio files by music information retrieval methods (Pampalk et al., 2003), were used to implement the similarity function for songs.

Hot item recommendation should support users to be up-to-date within the range of their preferences. This recommendation also helps to satisfy community needs, where a user wants to be best informed within his/her social environment (e.g., *more accurate than friends*). This recommendation is generated based on the user's preferences and the corresponding item attributes.

Broaden my horizon is a recommendation strategy supporting the user to explore his/her taste in a well guided manner. From an operators view, cross-selling capabilities are stressed by this strategy. Starting from well defined preferences (e.g. favored artists, preferred genres, etc.) the user can explore his/her taste by allowing more and more explorative recommendations guided by domain knowledge. The direction of this broadening process is mainly defined by the domain specific structure of the item space and the preferences of the user.

For the music domain the sound similarity of songs together with the genre model and its abstractions defined by STOMP[1] – see Figure 4.1 on page 59 – were intended to guide this broadening process. As shown in Figure 4.1 on page 59, the STOMP model associates 14 music genres to four high level music preference dimensions, defined as: *Reflective & Complex*, *Intense & Rebellious*, *Upbeat & Conventional* and *Energetic & Rhythmic*.

This categorization together with the interrelation of these four dimensions was used to guide the *broaden my horizon* process. So, for example, a user preferring classical music will first receive jazz recommendations before hip-hop or dance.

[1] Short Test Of Musical Preferences, developed by Rentfrow and Gosling (2003)

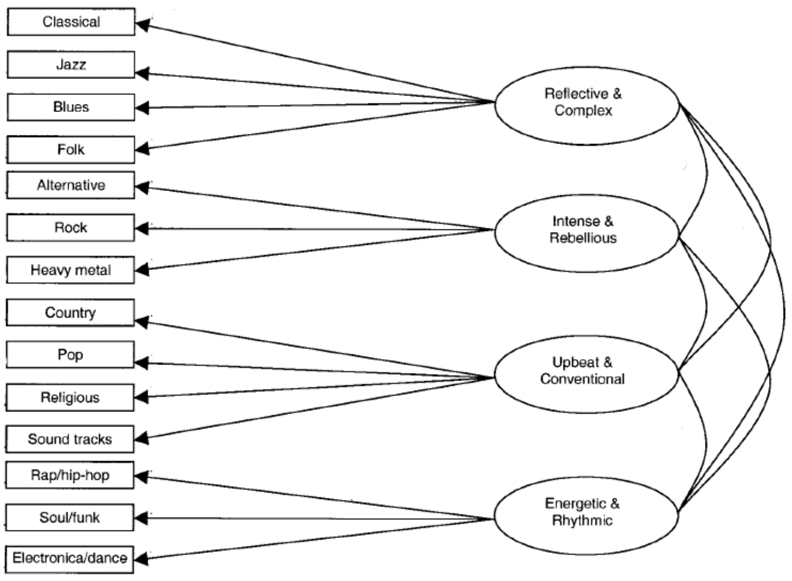


Figure 4.1.: STOMP model: Correlations of Genres and Preferences

The **similar users like** recommendation addresses social aspects: users want to know what others do. The similarity between users is defined by the similarity of their profiles or a subset of profile attributes. Furthermore, several different similarity relations can be defined either based on the user's behavior – for example, buying/rating history – and/or on the model dimensions, e.g., socio-demographic data. So the collaborative aspect can be implemented very efficiently. Recommendations are generated by finding the *k-nearest neighbors* based on the user's profile. Having found these similar users, their favored items are used for assembling recommendations.

The **related Items** recommendation tries to find associations between items which are not part of the item model or cannot be derived from the available item attributes. This recommendation is generated by using information retrieval approaches (e.g. music information retrieval techniques for finding similar sounding tracks to a given one (Pampalk et al., 2003; Aucouturier and Pachet, 2004)) or statistical algorithms e.g., association rule mining (Agrawal and Srikant, 1994; Park et al., 1995).

At least as important as the availability of appropriate algorithms is the knowledge as to where, how, and when to apply them. In real world applications especially user profiles pass through several stages of quality. Often the quality of profiles evolves from *bad* – at the very beginning where often only basic information (e.g. only an identifier) is available – to *ade-*

quate and hopefully to *good* with increasing maturity of the system. Therefore, it is important to know which algorithms should be applied at which state of profile quality.

The following table presents the strategies pursued in our approach concerning

- user and item profiles in low and high quality[2]
- the different recommendation strategies as mentioned above, and the algorithms used for implementation

User Profile Quality	**Item Profile Quality**	**Strategies**
high	high	reminder; broaden my horizon; more like this; similar users like; related items
high	low	reminder; broaden my horizon; similar users like; related items
low	high	reminder; more like this; related items
low	low	reminder; related items

Table 4.1.: Profile Quality and Strategies Applied

[2]low means knowing only the identifier of an item and high implies the availability of a set of meta data attributes

4.3. Architecture

The performance and scalability requirements are met by a very flexible multi layer architecture as presented in Figure 4.2.

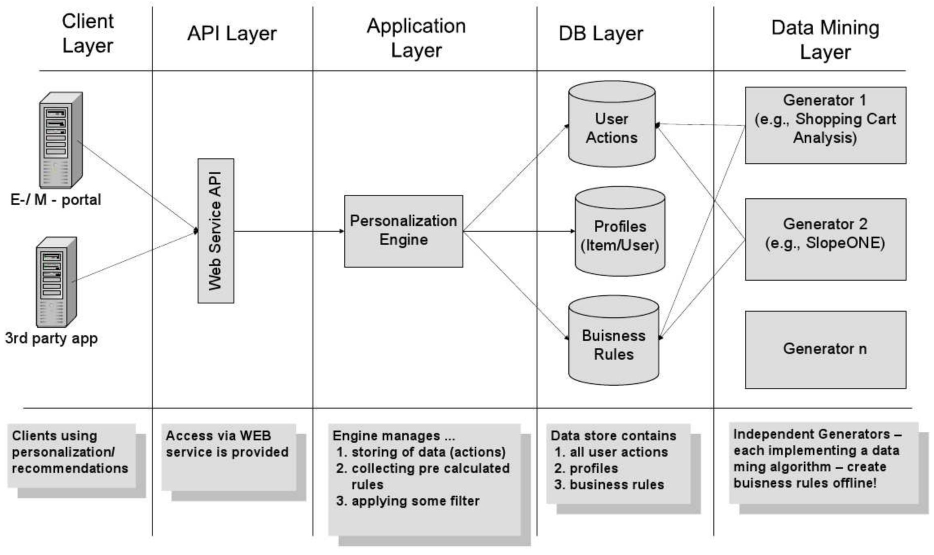

Client Layer
API Layer
Application Layer
DB Layer
Data Mining Layer
E-/ M - portal
3rd party app
Web Service API
Personalization Engine
User Actions
Profiles (Item/User)
Buisness Rules
Generator 1 (e.g., Shopping Cart Analysis)
Generator 2 (e.g., SlopeONE)
Generator n
Clients using personalization/ recommendations
Access via WEB service is provided
Engine manages ... 1. storing of data (actions) 2. collecting pre calculated rules 3. applying some filter
Data store contains 1. all user actions 2. profiles 3. business rules
Independent Generators – each implementing a data ming algorithm – create buisness rules offline!

Figure 4.2.: Architecture Overview

Clients using the system for personalization are accessing the provided functionality via the API layer, a well defined set of APIs (e.g. Web services), forming the standard access interface. The personalization engine – the core component of the application layer – is responsible for assembling recommendations based on pre-calculated business rules, storing user feedback and managing user and item profiles. A set of independent generators (data mining layer) calculates specific business rules off-line based on the provided data such as user actions and profiles. These rules are stored explicitly in the database (database layer) to be accessed by the personalization engine. This approach guarantees scalability and performance also in heavy load environments. A detailed description of the system architecture is provided in Chapter 5.

4.4. Procedure Model

To provide guidance concerning the applicability of the tools and concepts developed so far, a procedure model for applying these techniques was developed during a series of R&D projects. Basically, the design of a personalization system can be divided into 2 phases:

1. domain specific considerations
2. technical design

In the course of our R&D projects we found evidence that each domain has its own characteristics which have to be considered during the design of a personalization solution. A personalization system for a portal mainly serving occasional customers or offering goods with short life cycles has to provide other solutions than a system designed to assist regular customers. So, for example, a user just downloading an item (e.g., wallpaper, ring-tone, etc.) for styling his mobile device, will normally not be interested in recommendations offering more items of the same type, while in the context of books a user, just buying a thriller, will appreciate some more suggestions from this genre. The different suitability of the same strategy, presenting *more-of-the-same* items, is caused by the different needs the two recommendations have to satisfy. In the former case, a user will style his/her mobile device according to current trends and will therefore not be interested in downloading items in advance. In the latter case, a user is buying books according to his/her personal taste, which will not change rapidly. The consideration of domain characteristics as well as customer requirements are, although cornerstones for the development of a successful personalization system, beyond the scope of this thesis and will therefore not be discussed any further.

For creating the technical design the following steps are recommended:

1. **Definition of objectives**: The goals being achieved by the personalization system together with parameters, procedures, etc. for measuring the achievements of objectives should be defined carefully.
2. **Definition of constraints**: The constraints applicable for the system to be developed must be defined/elicited carefully. Basically the quantity structure concerning items and users as well as requirements referring to the performance and/or response time of the system are important constraints.
3. **Definition of user profiles**: The appropriate dimensions are identified/defined together with an questionnaire – if appropriate – the user

should answer during a registration process. Furthermore, the dimensions must be assigned to the different views – self-assessment, system observations, and community-assessment.

4. **Definition of the item profiles**: The appropriate dimensions are identified/defined together with the procedure for their elicitation thus avoiding attributes which cannot be instantiated in real world systems.

5. **Definition of dimensions**: Precise definition of the dimensions used, concerning name, type, value range, and constraints.

6. **Recommendation definition**: Selection of the appropriate strategies/algorithms as defined in Table 4.1 on page 60 and the appropriate parameterization of these algorithms.

7. **Sanity check**: It should be validated that the combination of profiles, dimensions, algorithms, as well as the procedure for measuring the achievements of objectives is sound and that all sources of troubles – as listed below – are eliminated:

   - Avoid unused attributes/dimensions
   - Avoid attributes/dimensions which cannot be ascertained
   - Avoid evaluations which cannot be performed properly when certain dimensions are missing

It is worth mentioning, that the first two items – the definition of goals and constraints – should be seen as binding contracts being agreed upon by the parties involved (e.g., the management and personalization engineering team), most important for achieving a satisfying project result.

4.5. Summary

In this chapter a survey of the core aspects of the *Adaptive Personalization* approach was presented, by introducing a multi-view profile model, a set of needs driven recommendation strategies, a flexible and robust architecture and a procedure model being used. In the following chapters we provide a detailed discussion of the multi-view profile model and the algorithms developed.

5. Adaptive Architecture

In this chapter the system architecture and the underlying data model is presented on a conceptual level. After defining the requirements in more detail in the first section, the system's architecture will be described, focusing on the core components of the system. In the third section a closer look to the definition of the data model is presented, forming the technical basis for the realization of the *Adaptive Profile Model* – see Chapter 6. A short summary concerning the main advantages of the architecture presented so far will be given at the end of this chapter.

5.1. Requirements

As mentioned before, the development of the *Adaptive Personalization Approach* was originated in the course of a series of R&D projects accompanying the realization of the *Ericsson's Media Suite - Music*, a commercial download platform for music related content. This commercial origin is also reflected by the evolutionary history of most of the design principles or requirements chosen so far to realize the current approach.

The project goals, defined at the beginning of the project[1], changed massively every few weeks thus forming a rapidly moving target the designers of the personalization solution had to achieve. So, for example, the requirements concerning the quantity structure of items changed from about 100.000 at the very beginning, to one million after some weeks and ended up with no limitations at all.

By far more severe than these changes was the inability to appoint to a certain application area impacting massively the definition of user and item data models. Thus, starting as a music personalization system, the requirements changed to deal with all kinds of items allowing any number and types of attributes due to cross selling considerations. Furthermore, also the definition of metrics for measuring the project's success could not be appointed in a sufficient way due to ongoing changes of the basic business models[2].

[1]According to the procedure model as presented in Section 4.4 on page 62

[2]The reasons for these moving targets were not an overstrained project management but the complexity and speed of the project.

This situation led to the definition of the following requirements and goals which should be met/supported by an appropriate architecture:

- *Scalability*: The system must be adaptable also to heavy load scenarios.
- *Performance*: The system's response time must be within the range of standard applications supporting real time services.
- *Adaptability*: The system shall support any problem domain. Necessary adaptations should be limited to specific optimizations only.
- *Extensibility*: Adding new data mining algorithms should be possible with minimum efforts.
- *Transparency*: Recommendations shall be transparent to the user as well as to an administrator, the person responsible for maintaining the system.
- *Manageability*: Monitoring and tuning support must be provided. Furthermore, an administrator shall be able to manipulate recommendations, e.g., define recommendations explicitly.
- *Traceability*: All actions concerning the system should be transparent to the administrator.

5.2. System Architecture

For best achieving the requirements defined above, the following major design decisions were made:

1. off-line computation and on-line assembling of recommendations
2. business rules – the results of different recommendation algorithms – should be stored explicitly
3. development of a generic data model for defining item and user profiles
4. user actions and system responses should be logged continuously
5. recommendation/data-ming algorithms should be available as self contained computation units

Furthermore, the personalization system should be accessible through state-of-the-art interfaces (e.g., such as Web services) providing easy integration on the client side thus forming the basis for *software as a* *service*[3] usage scenarios.

These considerations led to a distributed architecture, presented in the deployment/component diagram in Figure 5.1.

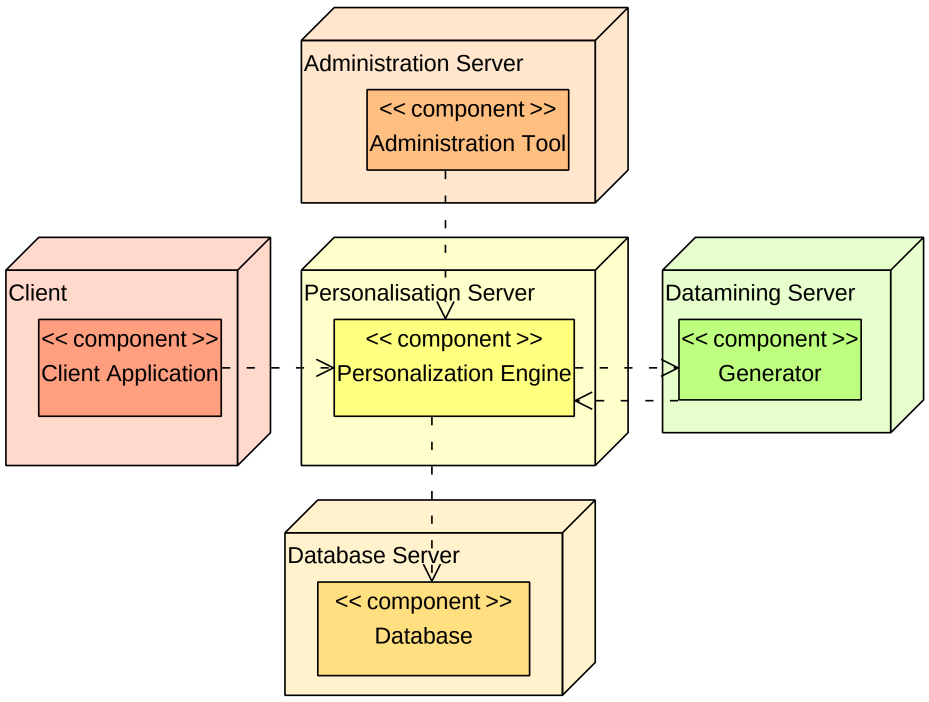


Figure 5.1.: System Overview

Client Applications are accessing the **Personalization Engine** by sending input data such as user actions, profile information, etc. and/or by requesting recommendations. All this information, together with the business rules produced by the **Generators**, is stored in a **Database**. The assembling of requested recommendations is mainly performed on the basis of pre-calculated business rules, produced off-line by different generators. Furthermore, with the help of the provided **Administration Tool** the personalization engine can be monitored, configured, etc. by an administrator. The core modules and their dependencies are illustrated in the component diagram in Figure 5.2 on page 68 which will be the basis for a more detailed discussion.

[3]Following the definition found in `http://www.webopedia.com` (as of 07/2009), software as a service – short SaaS – is a software delivery method that provides access to software and its functionality remotely as a Web-based service.

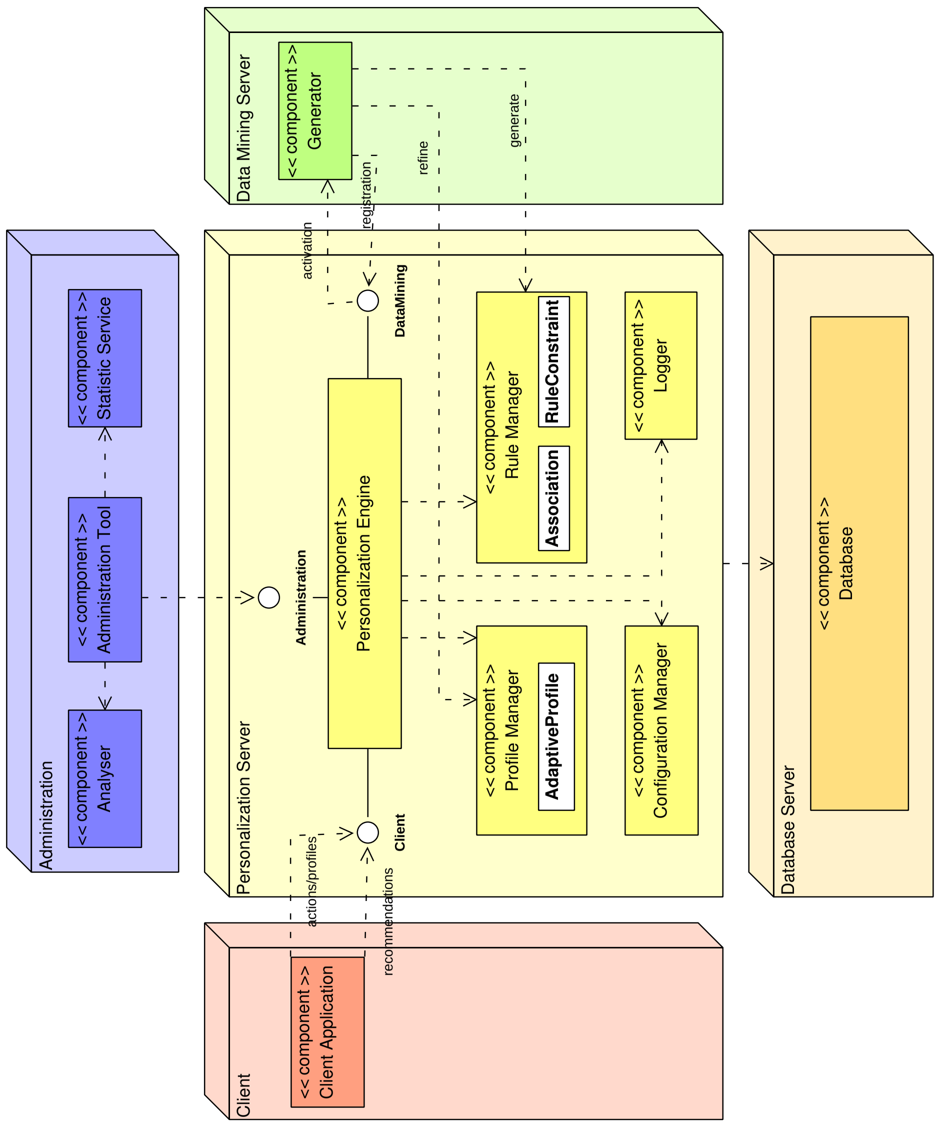


Figure 5.2.: Major Components

As mentioned above, a **Client Application** is communicating with the **Personalization Engine** by using the *client* interface which can further be divided into an *action interface*, a *profile management interface* and a *recommendation interface*. By using the action interface, the client is able to send information about actions a user is performing in the client application such buying an item, voting or rating an item or even only viewing the presentation of an item. Basically, such action information can be described by an action vector – see Definition 5.1 – consisting of (i) a user identifier, (ii) an object identifier, (iii) the kind of action (e.g., buy, rate, etc.) and (iv) a time-stamp describing when the action occurred.

$$Action = \langle\, userId,\ objectId,\ actionKind,\ time \rangle \tag{5.1}$$

Additional to this action information a client application can provide more details about users and items to the personalization system by using the profile management interface. By doing so, the personalization engine is enriched with domain knowledge – e.g., descriptions of users and items – allowing more and even better recommendations. The provided recommendations can be requested using the *recommendation interface* returning a ranked list of objects.

The **Personalization Engine** is mainly responsible for handling the input data, for assembling the requested recommendations and for providing appropriate management interfaces. As shown in Figure 5.2 on page 68 this component is offering interfaces for:

- client applications being personalized
- administration tools supporting the management of the system
- data mining applications – called generators – responsible for calculating the business rules

The received user actions together with all profile relevant input (such as updating, removing, etc. of profiles) is handed over to the profile manager, the component responsible for the creation and refinement of profiles. The assembling of recommendations is mainly based on the information provided by the profile manager, the rule manager and the configuration manager. Furthermore, the personalization engine is informing the *Logger* to track user actions and requested recommendations permanently.

The **Configuration Manager** is responsible for providing and managing the definitions of the supported recommendations strategies. Strategies (like *Hot Items Recommendations* see Section 4.2 on page 57) can be composed of different core data mining patterns or algorithms (e.g., available as pre-calculated rules) in a declarative way. Supporting the administrator in configuring these definitions as well as to provide access to the **Personalization Engine** are the main tasks of this component.

Furthermore, the personalization engine is supporting a registration interface for business rule generators. Such a registration can be seen as a contract between the on-line personalization system and the off-line data mining modules containing the following information:

1. the kind of actions a generator wants to be informed
2. a description of the produced results
3. a description of the parameters an administrator can configure

By receiving a specific action, the personalization engine is informing all generators registered for that action by forwarding the appropriate information.

A **Generator** is a self contained data mining module responsible for generating business rules based on the data provided. By registering at the personalization engine the generator is providing information about the kind of actions being used as an input and a description concerning its result. Receiving an action the generator decides, based on local settings, how to further proceed with this information. Possible strategies are ranging from immediately starting calculations to storing the actions locally being used by a time triggered batch process later. Basically generators can produce two kinds of results – business rules, modeled as associations of objects and refinement information concerning profiles – which are send back to the personalization engine.

The **Profile Manager** is responsible for the creation, maintenance, and refinement of profiles. User actions are collected, condensed, and added to profiles. Furthermore, profile related retrieval methods being used for assembling recommendations are supported. Also profile based algorithms such as the D^2-Tree, as presented in Chapter 7, are also provided by that module.

The **Rule Manager** is responsible for the maintenance and selection of appropriate business rules requested by the personalization engine. The core items this component is operating on are associations between objects, called business rules, and constraints concerning the applicability of these rules. Basically a business rule describes an association between two objects having the form

$$Rule = \langle\ object1_{id,type},\ associatedTo_{name,value},\ object2_{id,type},\ time\rangle \quad (5.2)$$

where the objects are described by an *identifier* and *type* information and the association is specified by a *name* such as *isSimilarTo*, *soundsLike*, *boughtTogether*, etc. and a *value* quantifying its strength.

The *RuleConstraints* are logical expressions being used as filters for the appliance of the pre-defined rules. With these constraints an administrator is easily able to define restrictions due to current business, legal or other requirements without loosing the basic information provided by the generators.

Samples for such constraints are: *Suppress rules containing items of type 'Christmas Gift' in summer*, *Suppress rules which are older than 1 year*, etc.

For managing and monitoring the personalization system an administration tool is provided supporting the following functionality:

- statistics concerning the number and kinds of user actions, recommendations and business rules
- statistics/analysis dedicated to measure the performance of the system (business goals)
- statistics concerning the provided profiles
- viewing and manipulation of profiles, business rules, constraints and clusters
- configuration and managing of generators
- overview concerning the system status (system health)

Although a very important tool in real world applications, the administration tool is no core component of the personalization system and will therefore not being discussed any further in this thesis.

5.3. Data Model

As imposed by the requirements defined above, flexibility and adaptability were major concerns for the definition of the data model. Particular attention was turned to the creation of an object model based on the following design criteria:

1. the set of dimensions/attributes of a profile must be able to change over time

2. the redefinition of profiles must be possible without programming, so that domain experts as well as machine learning algorithms can adapt profiles

3. dimensions/attributes must be able to represent more complex data structures than just flat values e.g., hierarchical dimensions

The presentation of the data model developed for the *Adaptive Personalization* approach is divided into two diagrams, see Figure 5.3 and Figure 5.4 on page 75, by reason of clarity. While the former diagram is showing the core or first class objects of the data model, the latter is focusing on the profile details.

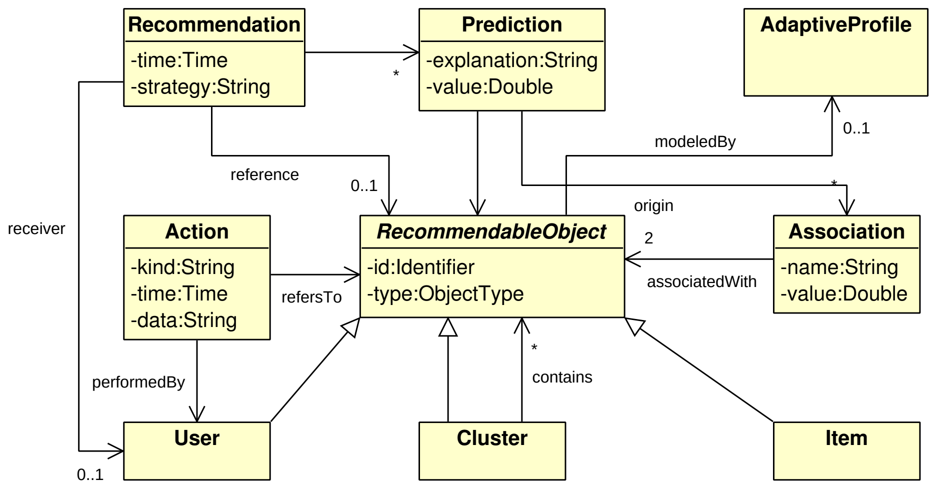


Figure 5.3.: Basic Data Model

As shown in Figure 5.3 all first class objects such as **User**, **Item** and **Cluster** are derived from an abstract root class **RecommendableObject**, uniquely identified by an identifier and a type (e.g., type *User*). Although the type is somehow redundant as part of an identifier, unique keys can be created more easily based on this information. Furthermore, the type information can be used as a filter attribute even on low level interfaces (e.g., calls to a database).

While **User** and **Item** objects are used to represent first class domain entities such as customers or consumer articles, **Clusters** are used to structure the problem domain. A **Cluster** can be seen as a named container for objects sharing commonalities in some respect. Between clusters associations can exist expressing some special relations, such as *is-sub-cluster-of*, *is-similar-to*, etc. modeled by the **Association** class. The number and kinds of associations is mainly guided by the problem domain. Music genres, often modeled as a hierarchy consisting of main- and sub-genres, are good samples for such clusters. Clusters itself as well as the association object-cluster can be created manually by the administrator or automatically by some cluster analyzing programs. As mentioned in Chapter 6 the cluster concept plays an important role in the context of profile refinement.

An **Association** represents a directed, weighted, and named relation between two objects and is mainly used to model business rules created by data mining modules or administrators. Association rule mining algorithms (Agrawal and Srikant, 1994), as used for shopping cart analyzes, are typical producers of such associations. The *Association* class can be seen as an analogy to the *RDF-triples*[4] as defined by the W3C[5] extended with a weight describing the intensity of the relation.

An **Action** object is used to describe the actions a user is able to perform on a given item and consists, beside references to the given user and item, of the action kind (e.g., buy, rate, search, etc.), some action data (e.g., search string) and the time the action was performed.

The **Recommendation** class is used to model the systems responses to recommendation requests. Basically, a **Recommendation** object contains a ranked list of **Prediction** objects (e.g., a list of consumer articles) representing the systems suggestions. This recommendation can be dedicated to a specific user in the context of personal recommendations and can be based on a specific object (e.g., “Users who bought product A also bought B, C, ...”). Furthermore, a recommendation refers to a specific strategy (e.g., *Hot Item Recommendation*) and contains time information as well.

A **Prediction** object describes the adequacy of an object in context of a specific recommendation. Beside the quality of this adequacy, described by the prediction value, also information being used by explanation models is provided. Because predefined business rules are one major source for assembling recommendations, **Association** objects form an appropriate basis for explanations.

[4]`http://www.w3.org/RDF` as of 07/2009
[5]World Wide Web Consortium, `http://www.w3.org` as 07/2009

In the case that more detailed information is available for a **RecommendableObject** an **AdaptiveProfile** object can be used, as presented in Figure 5.4 on page 75.

Basically an **AdaptiveProfile** consists of a definition- and an instantiation part. While the definition part contains a description of all possible attributes (the gross quantity) the instantiation is modeled by up to 3 **View** objects, each containing a set of concrete values. A **View** object represents one specific view of a profile (as defined in Chapter 6) and contains a set of **DimensionValue** objects forming the net quantity of the instantiated attributes of the given view. A **DimensionValue** object represents an instance of a certain **Dimension** representing a specific *Value*.

The **Dimension** class is used to model domain specific attributes (e.g., age, gender, etc.) and consists of a name (e.g., age), a specific **Type** (e.g., Integer) and an optional set of **Constraints**, describing restrictions/requirements to be granted (e.g., $0 \leq age \leq 130$). Note, **Dimension** objects are used to define the gross quantity of attributes used for modeling an *AdaptiveProfile*.

Type objects are used to model the basic characteristic of a **Dimension** analog to the type hierarchies of programming languages. The **WeightedType**, a subclass of **Type**, is of special interest because it is used to model weighted attributes such as *Preferences* or *Affiliations*, as described in Chapter 6.

The major advantage of the presented design is its flexibility concerning adaptations. Due to the generic data model removing or adding attributes can be performed easily by changing the profile definition.

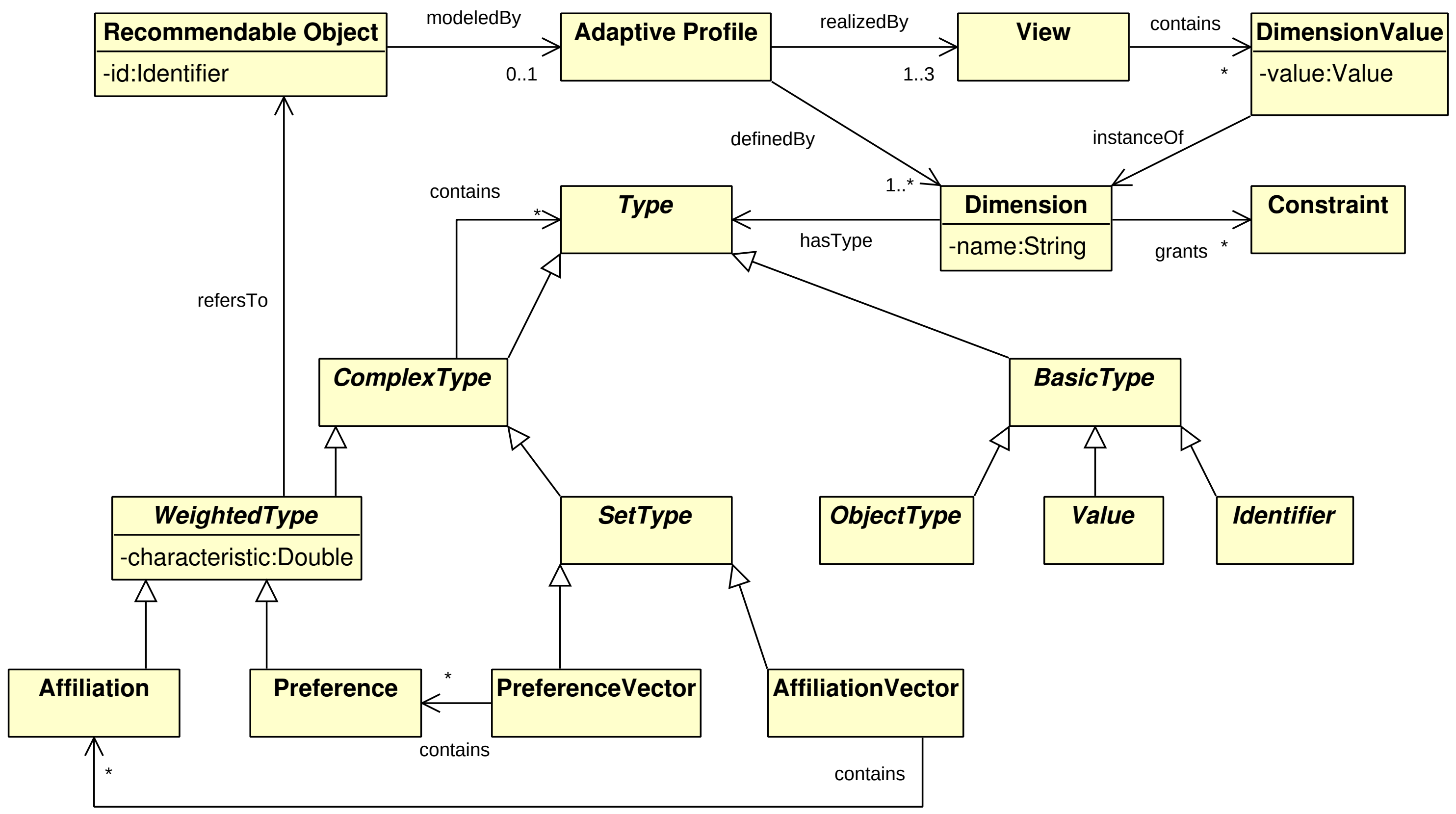


Figure 5.4.: Profile Data Model (main classes)

5.4. Summary

In this chapter a flexible and robust architecture was presented for best achieving the requirements defined above. Concerning the system architecture the major advantages can be summarized as follows:

1. The on-line assembling of pre-calculated business rules and contextual information (such as preferences and action history of the users) to personal recommendations reduces complexity and increases performance.

2. Storing business rules explicitly provide a complete overview to the administrator and ease of maintenance.

3. Generating the business rules off-line by self-contained computation units reduces complexity and increases scalability, stability, and performance of the on-line system. Furthermore, extending the system with additional data mining algorithms can be done easily and with minimum risk.

4. A continuously logging of personalization relevant user actions (e.g., rating of an item, visiting of a page, etc.) as well as system responses (e.g., provided recommendations) forms the basis for many kinds of analyses, especially those closely related to business aspects (e.g., *How many items, bought by users, were recommended in advance?*)

Furthermore, the profile model supports an evolutionary system development, because the complexity of the data model can be easily adapted. The system designer may start with a simple solution, using and representing the knowledge currently available, and is not forced to hypothesize about possible future directions and requirements. So the complexity of the system is always as high as necessary, without limiting expansion toward a more complex model.

6. Adaptive Profile Model

The more information available the better a personalization system can perform. In the contexts of recommender systems or user modeling servers this information is commonly provided in form of profiles containing the descriptions of the entities of interests such as users, items, or domain knowledge. A profile can be seen as a collection of information items describing a certain entity being available in many different forms:

- simple *name-value pairs* describing a certain attribute of an item such as age, gender, price, etc.
- a set of user actions or feedback information (e.g., ratings of users, tags of items, etc.) forming the basis for some learning algorithms
- more complex, compound attributes such as preference vectors adding probability to certain attributes (e.g., produced by some learning algorithms)
- rules describing some specific associations/relations among entities

Beside an appropriate set of attributes, further improvements of personalization services can be achieved by introducing the concept of context. In daily life users are interacting with information systems in a variety of different contexts resulting in different and perhaps contradictory context-profiles of a user. The usage of a search engine such as Google can be different in work or home contexts, the selection of restaurants or movies a person wants to book might be different concerning the day of week – e.g., at weekend with family or after work with business partners (Adomavicius et al., 2005).

Complex user modeling approaches such as the Unified User Context Model (UUCM) (Niederée et al., 2004) tackle this problem (see Section 3.2 on page 49), but they often rely on additional information hardly available in standard e-commerce applications (e.g., location information). In most cases user actions, a fundamental data source for personalization services, comprise of the following data:

- user identification, or at least session information in the case of anonymous users

- kind of action (e.g., buy, rating, etc.)
- item identification
- time of the action

Recommendation systems using time information for contextualization often use *reduction based approaches* (Adomavicius et al., 2005) where user actions (e.g., ratings) are clustered and analyzed along time constraints (e.g., all restaurant ratings on Sunday or Saturday vs. Monday till Friday).

The *Adaptive Personalization Approach* extends the contextualization to more psychological aspects of user feedback were the origin of an action – implicit or explicit feedback – is taken into account. The user's preferences can be defined explicitly on a questionnaire basis (e.g., as it is often performed during a registration process) and/or by analyzing his/her navigation behavior. In the former case a user explicitly expresses a self-assessment – the well defined needs – which has to be served by a recommender system in any cases. The later case corresponds to the ill defined needs which good personalization systems should be able to identify. For example: If a user expresses his/her interests for culture on a news portal but he/she is mostly reading articles about sports a good personalization system should recommend articles of both domains for satisfying the user's (perhaps idealistic) self-assessment and his/her real interests.

6.1. Modeling the User

As mentioned above, among the user's needs, a distinction can be made between well defined needs where the user is able to characterize an appropriate means of satisfaction, and ill defined needs where the user does not know how to satisfy or even how to define them. Furthermore, users are normally not isolated during the use of a personalized system also effecting the existing community. These effects can be very multifarious, ranging from deliberate interactions, such as the placement of ratings or recommendations, thus being a passive example for other users or algorithms, to giving/receiving feedback from/to other user.

Different views concerning aspects of a user in the context of a community are well addressed by the Johari window (Luft and Ingham, 1955; Thomas, 1992)[1], a psychological model for illustrating and improving self

[1]`http://www.businessballs.com/johariwindowmodel.htm` as of 11/2008

awareness and mutual understanding between individuals within groups – see Section 2.2 on page 12.

In context of personalization the information placed within the four regions of the Johari window can be seen as preferences or needs of a user and their visibility to the user him/herself and to the others. Furthermore, the role of the feedback contributor is shifting from the community to the personalization system itself. The *open/free area* contains all well defined preferences of a user within the personalization system (e.g., information given during a registration process). The *blind area* corresponds to ill defined preferences of a user which the system is able to identify. Identification can be based on the observation and analyzes of the users interaction with the systems, as it is done in content- or collaborative filtering approaches. The *unknown area* corresponds to preferences a user does not have/know at the moment and which cannot be derived from the user's interaction behavior. The *hidden area* corresponds to well defined preferences of a user which cannot be derived from the interactions by the system.

Therefore, the profile refinement process can be interpreted as the enlargement of the *open/free area* with means of feedback and disclosure as presented in Figure 6.1.

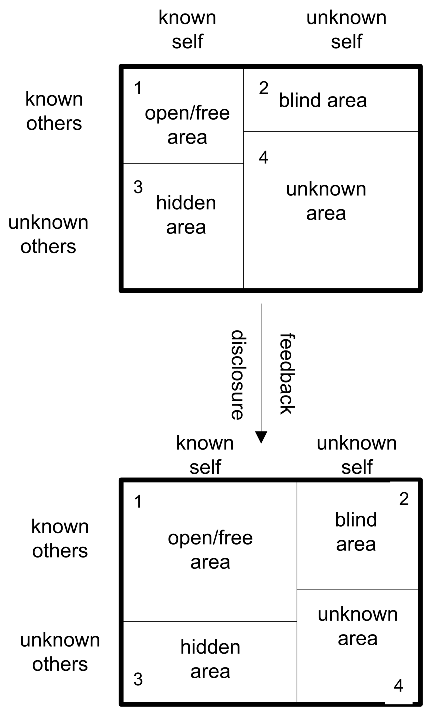


Figure 6.1.: Profile Refinement

Concerning these four regions, a personalization system should provide the following functionality:

1. *open/free area*: Generating recommendations based on the explicitly defined needs or preferences of a user. This can be achieved by using explicitly defined user preferences as the basis of recommendations.

2. *blind area*: Refinement of the user's profile concerning his/her preferences (and generating new recommendations) based on the user's interaction with the system. This can be achieved by applying recommendation approaches such as collaborative- and/or content-based filtering.

3. *hidden area*: Encourage the user to extend/refine his/her profile so that the system is able to improve the recommendation quality. This can be achieved by periodically asking the user in a non-intrusive way to complete/extend his/her profile.

4. *unknown area*: Broadening of the user's horizon (see Swearingen and Sinha (2002)) by supporting him/her in exploring his/her taste. This can be achieved by guiding the user to new territories of the item-space based on his/her current interests/preferences and certain domain knowledge. So, for example, within the *Ericsson's Media Suite - Music* the broadening of the user's musical horizon was guided using the inter-genre relations of STOMP (Rentfrow and Gosling, 2003) and his/her current interests as a starting point.

These considerations led to a multi-layer model where each user is modeled by a profile comprised of three different views as presented in Figure 6.2 on page 81. These views are:

- self-assessment
- system observations
- community-assessment/assessment of others

Each view, or sub profile, is implemented by a vector of attributes best describing the user model to represent. The *self-assessment* view is used to model the *self-portrait* of the user concerning information such as preferences, socio-demographic data such as age, gender, etc. mainly used to serve the *well-defined needs* but also psychological attributes, taken from models like the Five Factor Model (Tupes and Christal, 1992), the Pleasure-Arousal-Dominance (PAD) framework (Mehrabian, 1996) or Myers-Briggs Type Indicator (Myers and Myers, 1995), in order to get a

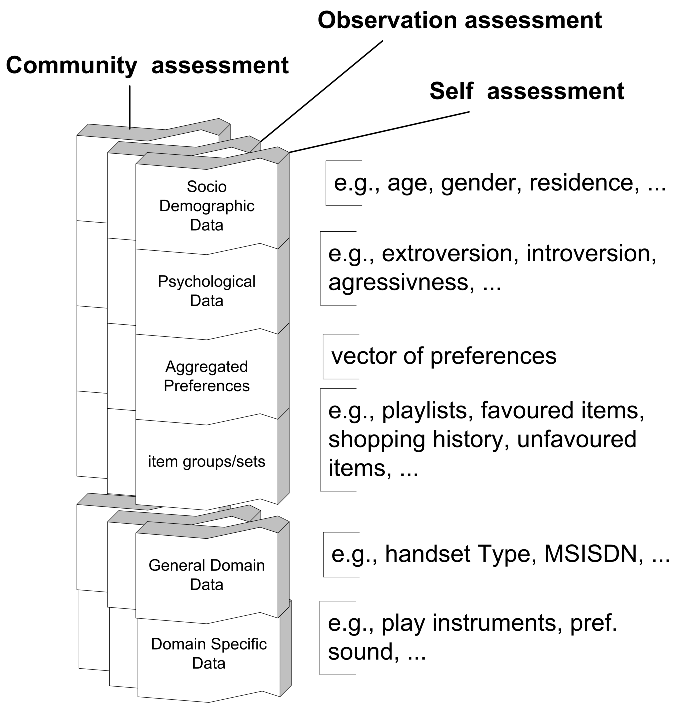

Observation assessment
Community assessment
Self assessment
Socio Demographic Data
Psychological Data
Aggregated Preferences
item groups/sets
General Domain Data
Domain Specific Data
e.g., age, gender, residence, ...
e.g., extroversion, introversion, agressivness, ...
vector of preferences
e.g., playlists, favoured items, shopping history, unfavoured items, ...
e.g., handset Type, MSISDN, ...
e.g., play instruments, pref. sound, ...

Figure 6.2.: Structure of User Profiles

broader impression/image of the user.

The *system assessment* is a view, where the behavior-based profile of a user is determined. The system observes the user while interacting in order to create a *dynamic* user profile. In contrast to the self-assessment, where users create representations of themselves (which is subjective, of course), the system observation profile represents what users really do in the system.

The *community-assessment* represents how a user is seen by others and can be used as a feedback or rating on the self-assessment view.

This complex model forms the basis on which a wide range of needs can be served. The information of the self-portrait can be used to satisfy the obvious needs – even (and especially important) when this description is somewhat idealized. The sub profile created and automatically refined through the system observation view is used to identify and satisfy behavior-based needs.

Where to place an attribute mainly depends on how this information can be elicitated:

- information such as socio-demographic data can only be defined by asking the user, therefore they are part of the self-assessment view
- explicit expressions of interest, such as ratings or purchases, are part of the self-assessment view
- implicit expressions of interest, such as visiting pages, are part of the system observation view
- feedback of other users concerning a given user are part of the community-assessment

The user profile structure as proposed for the *Ericsson's Media Suite - Music* supports the following groups of attributes:

1. Socio-demographic data, such as age, gender, and place of residence
2. Music genre preferences, currently referring to the genres used in the STOMP model (Rentfrow and Gosling, 2003)
3. Music connotation preferences based on a collection of moods and situations (e.g., car driving)
4. Compilations such as favored/unfavored artists, tracks, or playlists
5. Important aspects, such as preferred genres, importance of lyrics, preferred instruments
6. Historical user data based on bought, viewed, etc. items

In the case of recommendations the three views can be used separately or by combining them to a 'weighted-sum' profile by using predefined weights concerning the three views. The latter approach was implemented in Ericsson's Media Suite - Music.

6.2. Modeling the Items

A similar structure is also provided for the items to be recommended, where the affiliation of items to certain clusters (like genres, see below) is modeled with the help of three different views:

- the assessment of a domain expert
- the assessment of the (user) community
- affiliations calculated by a classifier systems as described later

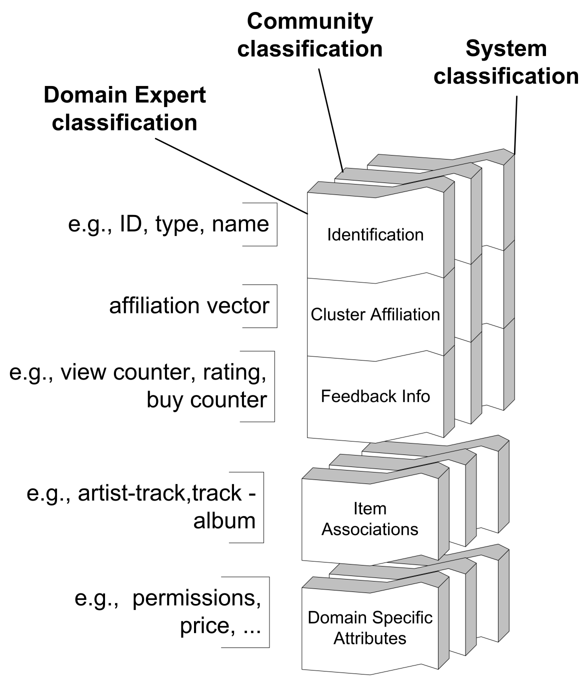

Community classification
System classification
Domain Expert classification
e.g., ID, type, name
Identification
affiliation vector
Cluster Affiliation
e.g., view counter, rating, buy counter
Feedback Info
e.g., artist-track,track - album
Item Associations
e.g., permissions, price, ...
Domain Specific Attributes

Figure 6.3.: Structure of Item Profiles

The structure is presented in Figure 6.3.

The domain expert's view often represents the opinion or assessment of the content owner, while the community view reflects how the content is seen by the consumers/users. By providing both views the content owner gets an important feedback and a better explanation model can be provided for the user concerning the recommended items. Especially in the context of music the affiliation of artists or tracks to some given genres is a very controversial topic, leading to arbitrary classifications and thus to hardly acceptable recommendations. The classifier view can be seen as an extension of the domain expert view, where 'third party' information is used to refine the item profile. This information can be provided simply by a catalog or even an appropriate classifier system.[2]

In the case of recommendations the three views can be used separately or a 'weighted-sum' profile can be created. The latter approach was implemented in Ericsson's Media Suite - Music.

[2]Within the *Ericsson's Media Suite - Music* project an audio classifier system, based on Music Information Retrieval methods (Pampalk et al., 2003; Aucouturier and Pachet, 2004), was used to define sound similarities.

6.3. Data Model

In the Adaptive Personalization approach, first class objects, i.e. users or items, are modeled along a multi-view profile consisting of three different views, as described above. Each view comprises of a set of *dimensions* describing specific attributes, such as the age of a person, the price of an item, etc., as shown in the conceptual data model in Figure 6.4.

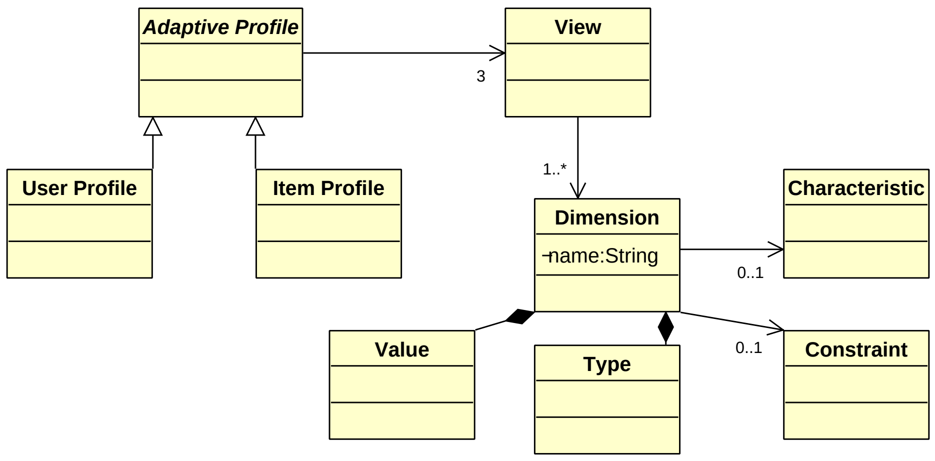


Figure 6.4.: Profile Data Model

Note, the model presented in Figure 6.4 represents a simplified, conceptual view showing only information relevant for the discussions of this section. This is especially true for the representation of the concept *dimension* where only the main attributes are depicted without any regards to an appropriate class hierarchy. The complete, technical data model is explained in Chapter 5.

A dimension itself is composed of

- a *Name* object (e.g., 'age')
- a *Value* object (e.g., '25')
- a *Data-Type* (e.g., Integer)
- the optional *Characteristic* of the value (e.g., 80% association to genre Pop)
- some optional *Constraints* concerning the value (e.g., min/max boundaries or functional constraints e.g., $a^2 + b^2 = c^2$)

The *Name* object of a dimension, an alphanumerical string, is used as the dimension's unique identifier within a view. The *Value* object of a dimension contains the current value, corresponding to the restrictions defined in the *Constraint*, and is interpreted according to the dimension's *Type*. A variety of different *data types*, ranging from Integer, String, etc. to more complex structures such as preference vectors, are provided within the Adaptive Personalization approach supporting the creation of appropriate domain models. Furthermore, these types form the basis for *distance functions* used to implement *k-nearest neighbor* algorithms as presented in Chapter 7. A more detailed technical description of the supported or proposed data types is provided in Chapter 5.

The optional *Characteristic* object, having a value between min (e.g., -1) and max (e.g., 1), is used for dimensions where the value can have different nuances concerning the *intensity* and is used to model skills, preferences or the prototypicality of items for specific clusters. In the context of the Adaptive Personalization approach, this information is used for profile refinement as described in Section 6.4 on page 86. Furthermore, two thresholds T_{neg} and T_{pos} are defined specifying if the intensity of a given dimension value is strong enough to be considered. Given the value p of a Characteristic object of a dimension d_i and the corresponding thresholds T_{neg} and T_{pos} we can specify:

- if $p \leq T_{neg}$ then the value of the dimension d_i has to be interpreted negatively
- if $p \geq T_{pos}$ then the value of the dimension d_i has to be interpreted positively
- if $T_{neg} < p < T_{pos}$ then the value of the dimension d_i can be ignored, because it's intensity is not strong enough

So, for example, the intensity of the preferences of a user for a specific genre g of literature or music can vary from a *negative* maximum (e.g., -1 with the meaning *hate it*) to a *positive* maximum value (e.g., 1 with the meaning *love it*). If the intensity is above T_{pos} then the system will try to find and recommend items out of the genre g. If the intensity is below T_{neg} then the system can use this information to construct a filter where all items out of g are removed from recommendations. In the context of recommendations, where a system is trying to find appropriate items, the upper threshold T_{pos} is more important than T_{neg}. Hence, for ease of discussion, T stands for T_{pos}.

Furthermore, a *Constraint* object can be defined to formulate specific semantical conditions concerning the range of dimension values (min, max values), logical expressions such as invariants, etc.

6.4. Profile: Initialization and Refinement

The ongoing refinement of profiles is a core feature of a personalization system mainly responsible for the performance and acceptance of the system. This continuous adaptation is not only important for user profiles, where a learning behavior of the system is expected by the user him/herself, but also for item profiles where the profile improvement can be used for better serving target groups.

But in the context of a real world system a successful adaptive behavior also has to meet the requirements *reasonability* and *planability* (see Chapter 1 and Chapter 5). The former requirement addresses the effort a user has to invest in training the system to learn his/her preferences and/or needs – addressing the *new user problem* (Adomavicius and Tuzhilin, 2005; Schein et al., 2002) – while the later focuses on the operators need exactly to define, adjust and communicate the preconditions which have to be met for being personalized (addressing the *cold start problem* (Adomavicius and Tuzhilin, 2005; Schein et al., 2002)).

The first version of Amazon's book recommender, called BookMatcher, required at least 20 to 30 ratings of a user before personalized recommendations could be generated. Not surprisingly this initial barrier was too high for most users and so this service was hardly used and was replaced by a better solution (Linden et al., 2003).

In this section a new, lightweight profile refinement approach is presented which is based on the following assumptions/preconditions:

1. each item[3] has a profile containing domain relevant dimensions (attributes), e.g., genre affiliation, etc.
2. these dimensions address some preferences/needs of users
3. all relevant actions of users on items (e.g., buy, rate, etc.) are logged within this system

The first two of these preconditions imply that some domain specific meta data is available which is relevant for the selection processes of users. In context of real world systems this assumption is not too restrictive, because some informative meta data must be available at least for presenting items to users (e.g., on web portals).

[3]the objects being recommended

6.4.1. User Profile: Initialization and Refinement

The initialization of a user profile, affecting the self-assessment view, can be performed through an optional, questionnaire based registration process asking for favored genres, artists, etc. As mentioned above, this self-assessment view is used as a baseline for generating recommendations serving the well defined needs. Both other views, the observation view and the community view, will start as empty profiles.

Standard machine learning approaches such as decision trees (Quinlan, 1986), clustering techniques (Witten and Frank, 2005; Pazzani and Billsus, 2007), etc. can be applied to the actions of the users thus forming the basis of the standard refinement process concerning the different profile views. So, for example, all items a user is viewing may form the basis for the refinement of the observation view, while explicit actions such as rating an item are dedicated to the self-assessment view.

Beside the applicability of these standard learning approaches a lightweight refinement algorithm, based on the assumptions above, was developed focusing on easily explainable and planable refinement strategies especially during the cold start phase for a given user. The basic idea of this profile refinement concept is to merge a defined set of dimensions of an item (those responsible for the satisfaction of user needs) the user is acting upon into the user's preference profile weighted by the importance of the action.

This merging process is based on an appropriate modeling of:

- the user's profile dimensions concerning preferences
- the item dimensions responsible for need satisfaction
- an appropriate merging function for changing the user's preference-dimensions according to the corresponding item dimensions

User preferences can be modeled by using dimensions with the capability of a weighted value representation of a given dimension/attribute as presented in Figure 6.4 on page 84. So the fact, that a given user likes a specific item (e.g., an artist, book, etc.) or concept i (e.g., genre, etc.) to a certain extend e can be modeled by using the identifier of i as the dimension's *Value* object and e to initialize the *Characteristic* object, having a value between a minimum and maximum value (e.g., -1 to 1) representing the different degrees of liking, ranging from *strong dislike* to *high preference*.

Also attribute values of items, such as the affiliation to a specific concept c (like genre, etc.) with a certain strength s, can be modeled similar to user preferences by using c as the value and s as the characteristic of the dimension.

The merging function $merge$ is responsible for an appropriate adaptation of the current user profile u_t (at time t) based on action a the user is performing on item i resulting in a refined profile u_{t+1} – see Definition 6.1.

$$u_{t+1} = merge_1(u_t, a, i) \tag{6.1}$$

The merging function $merge$ only effects preference dimensions in the user profile where only the corresponding *Characteristic* object is adopted but not the *Value* object itself. Furthermore, the merging function can create new preference dimensions if necessary.

So, for example, when a user is expressing high interest in a given item (e.g., buying a book) some relevant dimensions of this item (e.g., author, genre of the book) are merged into the user profiles where the *Characteristic* of the value object of the appropriate preference dimension is adopted. The set of relevant dimensions is specified by a filter f applied on the dimension set of an item in a given context – see Definition 6.2, where $merge_1$ is extended with this filter.

$$u_{t+1} = merge_2(u_t, a, i, f) \tag{6.2}$$

A simple merging function $merge_2$ is presented in Definition 6.3 where w_a represents the predefined importance (weight) of the action a while the expression $f * i$ defines the set of item dimensions to be considered in the merging process.

$$u_{t+1} = merge_2(u_t, a, i, f) = u_t + w_a(f * i) \tag{6.3}$$

The drawback of the function $merge_2$, presented in Definition 6.3, is its tendency to collect and to reinforce preferences in real world contexts, because users tend to give more positive feedback than negative – see Chapter 8. This is especially true in the context of the *observation view* refinement, where the behavior (e.g., navigation behavior) of the user is taken into account because most users try to find items they are interested in.

Therefore, a neglecting behavior is introduced – called *reduction by ignoring* – by applying a weakening function $decr(u)$ on preference dimensions of a user profile during the merging process. An appropriate weakening function $decr(u)$ degrades the value of the Characteristic object of a preference dimension toward a threshold T (e.g., 0) where the user's preference for the given item can be ignored – see Definition 6.4.

$$merge_3(u_t, a, i, f, decr) = decr(u_t) + w_a(f * i) \tag{6.4}$$

A simple implementation for $decr$ is the usage of a neglecting constant κ as presented in Definition 6.5

$$merge_3(u_t, a, i, f, \kappa) = \kappa u_t + w_a(f * i);\ 0 \leq \kappa < 1 \tag{6.5}$$

which will reduce the value of the Characteristic object during a merging process according to the course of the function presented in Figure 6.5[4].

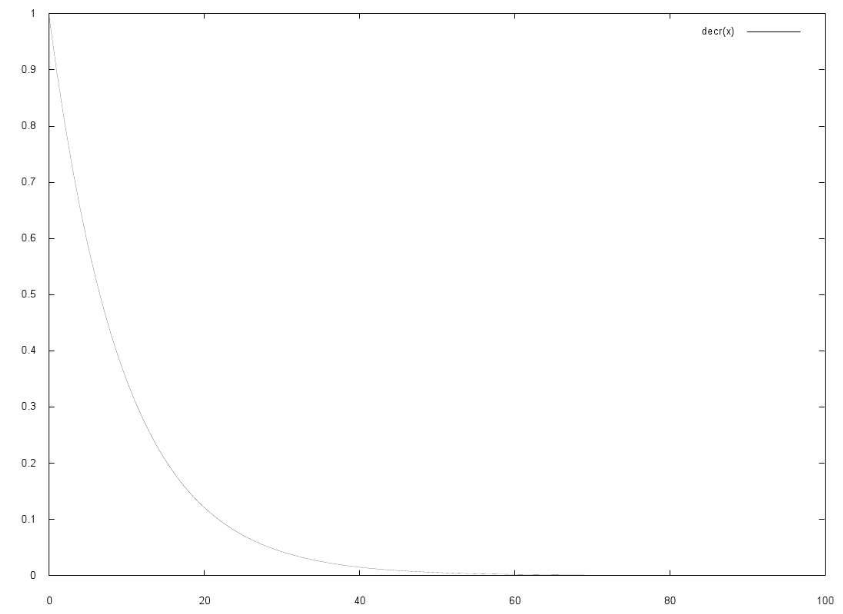


Figure 6.5.: Neglect Characteristic κu_t with $\kappa = 0.9$

Based on the definition of a merging function $merge$, as presented in Definition 6.5, we are now able to define the refinement/adaptation course for a given preference dimension. According to Definition 6.3 on page 88 the refinement of the *Characteristic* object p of the preference dimension d_i of a user profile u can be specified as follows

$$p_{t+1} = \kappa p_t + w_a q \tag{6.6}$$

[4]In many cases smoother functions (e.g., having a sigmoid characteristic S shape) like the Gompertz function, are more appropriate for implementing the weakening behavior of preferences.

where q is the value of the *Characteristic* object of the item dimension used by the merging function. By transforming this recursive equation we get

$$p_t = w_a q \sum_{i=0}^{t-1} \kappa^i = w_a q \left(\frac{1-\kappa^t}{1-\kappa} \right) \tag{6.7}$$

as the formula for calculating the value p_t if $p_0 = 0$ and $t \geq 1$. Based on the assumption that $-1 \leq w_a \leq 1$ and $-1 \leq q \leq 1$ we have to normalize the fraction $\frac{1-\kappa^t}{1-\kappa}$ by using the $\lim \sum_{i=0}^{\infty} \kappa^i = \frac{1}{1-\kappa}$ as shown in Definition 6.8 on page 90.

$$p_t = w_a q \left(\frac{1-\kappa^t}{1-\kappa} \right) (1-\kappa) = w_a q (1-\kappa^t) \tag{6.8}$$

The course characteristic of the refinement function as presented in Definition 6.8 is shown in Figure 6.6 on page 90.

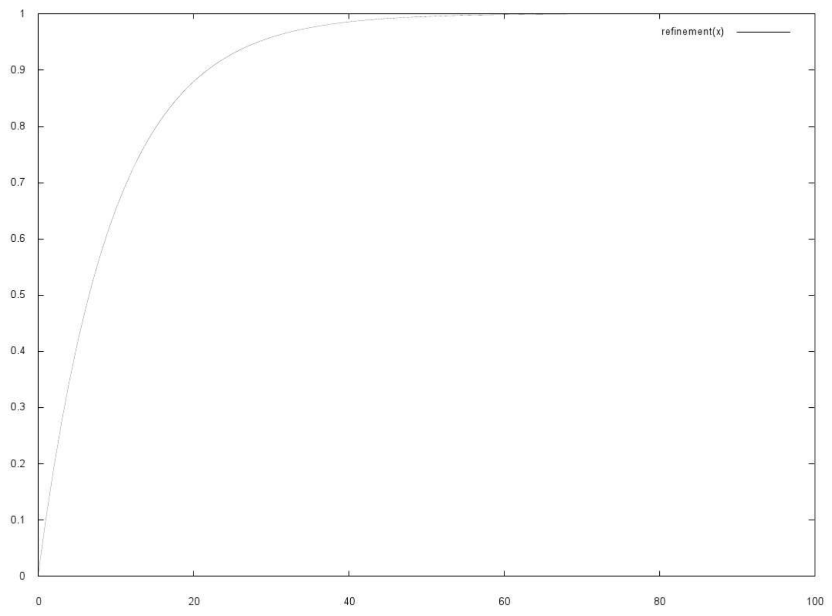


Figure 6.6.: Refinement Characteristic with $\kappa = 0.9, w_a = q = 1$

In the context of real world personalization systems it is important, especially for the operator of the system, to have a precise understanding when the provided learning behavior will become effective resulting in generating personal recommendations. Based on the equation defined in Definition 6.8 we are able to derive an appropriate threshold T for a given number of user actions necessary for incrementing the intensity of a specific preference dimension to be considered for personalization purposes. A preference is called effective if the value of p_t is above T after t merging processes, as shown in Definition 6.9

$$w_a q(1-\kappa^t) \geq T \tag{6.9}$$

Depending which parameters (e.g., w_a, κ) are known Definition 6.9 can be used to define the missing value. In most cases κ, the weakening function, will be predefined as well as q which defines the average intensity value of the corresponding dimension value of an item to be considered/merged. As a result, the weight of the action w_a and the threshold T have to be derived from Definition 6.9.

Transforming Definition 6.9 leads to a formula for the action value w_a – see Definition 6.10.

$$w_a \geq \frac{T}{q(1-\kappa^t)} \tag{6.10}$$

To define an appropriate value for T the following requirements must be considered

- because of $w_a \leq 1$ also $\frac{T}{q(1-\kappa^t)} \leq 1$
- use the minimal number of the strongest actions – necessary for making a preference effective – for defining T

leading to Definition 6.11

$$T \leq q(1-\kappa^t) \tag{6.11}$$

implying that the minimal number of necessary actions t is 1.

Example: Given a portal offering items (e.g., books, music, etc.) which are assigned to genres gradually expressing their prototypicality for the given genres. The preference of a user for a given genre g should be effective after *viewing* an item i of g with an item-genre affiliation intensity of at least 80% for five times ($t = 5$). Furthermore, *buying* an item is considered as the most expressive action and should be lead to an effective preference after buying at least two items of g ($t = 2$). Based on this information and the formula presented in Definition 6.9 on page 91 an appropriate action weight for the user actions *viewing* and *buying* must be derived. Furthermore, it is assumed that $\kappa = 0.9$.

First we want to define an appropriate value for T. Based on Definition 6.11 on page 91 and the fact, that buying is the strongest action we get (using $t = 2$):

$$T \leq 0.8(1 - 0.9^2) = T \leq 0.152 \tag{6.12}$$

Having defined this value for T we are now able to define the weights for the actions view w_{view} and buy w_{buy}. Applying the given parameters to Definition 6.10 on page 91 we get using $t = 5$)

$$w_{view} \geq \frac{0.152}{0.8(1 - 0.9^5)} = w_{view} \geq 0.464 \tag{6.13}$$

for the viewing action, and

$$w_{buy} \geq \frac{0.152}{0.8(1 - 0.9^2)} = w_{buy} \geq 1 \tag{6.14}$$

for the action buy (using $t = 2$). So, using these parameters, the operator of this sample portal is able to configure the personalization system precisely so that after 2 purchases or 5 viewing actions of a user on items of a genre g the user's preferences for g will be considered.

According to the refinement function as defined in Definition 6.6 on page 89 preference values can also be decreased, depending on the values of the parameters. The most important cases for such a preference dilutions are:

1. the contribution of the term $w_a q$ is negative: This situation occurs when a user is performing a negative action on item i, expressing his dislike for i e.g., by giving a negative rating like "don't like". In this case that the action weight is negative ($w_a < 0$) this will lead to a value decreasing during the merging process (assuming that $q > 0$). Also a positive action on an item with $q < 0$ could lead to $w_a q < 0$ but in domains, where the prototypicality of an item concerning a cluster is modeled, this is not a very common use case.

2. the contribution of the term $w_a q$ cannot compensate the effect of the weakening function $decr(u)$: This often occurs, when a user is performing a low expressive action (e.g., viewing an item where $|w_a|$ is nearby 0) on an item having e.g., a weak affiliation to a cluster.

3. the contribution of the term $w_a q$ is 0: This we call *reduction by ignoring* and is a special case of item 2. This occurs when a preference dimension d of a user is considered within a merging process, but the item i does not have a corresponding dimension for d. So, if a user

with preferences for *Classic Music* is buying a *Hard-Rock* song – having only affiliations to *Hard-Rock* – the users preference for *Classic Music* will be decremented.

6.4.2. Incorporating Domain Knowledge

As already mentioned in Chapter 4 (and being discussed in more detail in Chapter 5) structuring the problem domain is a core concept of the *Adaptive Personalization* approach. Inter related clusters, often used to represent domain knowledge (e.g., a genre tree or graph in the music domain), are supported as primary concepts and can therefore be used for user profile refinement.

Superclass relations, modeled as *is-a* associations among clusters, can be used to propagate preferences from concrete concepts to more abstract ones, by applying the merging strategies discussed so far along the superclass paths concerning the cluster affiliations of items.

Example: Let us assume an item i (e.g., a song) being associated with a cluster c_{sub} (e.g., genre *Classic Pop*) to a certain extent. Furthermore, cluster associations exist containing an *is-a* relation between c_{sub} and a superclass c_{super} (e.g., genre *Pop*). So, if a user is expressing his preference for item i (e.g., by buying it) the merging function of Definition 6.8 on page 90 is not only being applied to the cluster association with c_{sub} but also, adorned with some weakening factor, to c_{super}. The adapted formula can be found in Definition 6.15, using the path length as the weakening factor.

$$p_{super,t} = w_a q_{super}(1-\kappa^t)\frac{1}{pathLength(sub, super)+1} \tag{6.15}$$

6.4.3. Short-Term and Long-Term User Preferences

With the merging strategy discussed so far we are able to precisely configure the personalization system for individual recommendations, based on the explicit and implicit user feedback. But a serious drawback of this algorithm is its inflexibility concerning short time and long time preferences. If a system is configured for promptly serving new customers (e.g., by using a low value of κ and a low number of necessary actions) – thus addressing the new user problem – it tends to overstate the latest actions of the user and will mainly recommend items of the most current usage context. This strategy may be quite sufficient for walk-in customers but it will fail to satisfy regular clients. Otherwise, if the system is tuned for a tentative behavior concerning preference learning (e.g., by using a high value of necessary actions and a high value of κ) the system will suffer from the new user problem.

To overcome these limitations the merging strategy discussed above was extended with a session based concept for an appropriate modeling of short-term and long-term preference adaptations. The main idea is to split a user preference profile in a short-term and long-term model where the short-time preference profile is refined by the user feedback (e.g., actions) within a session and the long-term model's adaptation is based on a series of short-term preference profiles. According to Definition 6.6 on page 89 we can formulate the following equation:

$$p_{t+1} = \kappa_{long} p_t + w_{short} s(p_t) \tag{6.16}$$

where κ_{long} is the long-term weakening function and w_{short} is the weight of the session based adaptation $s(p_t)$ of the user's preference value. The session based refinement is implemented by $s(p_t)$ and can be formulated as follows:

$$s(p_t) = \kappa_{short} p_t + w_a q \tag{6.17}$$

where κ_{short} is the short-term weakening function and w_a is the action weight, as defined above.

At the beginning of a session, e.g., the login of a user, the long-term preferences are used to initialize the session based short-term preference profile. Within the current session the profile is adapted on the basis of the user's actions (see Definition 6.17) as discussed above. After the closing of a session (e.g., logout) the current short-term preference profile is merged with the long-term profile as described in Definition 6.16.

Based on this distinction between short-term and long-term preferences also the recommendation process can be optimized because both profiles can be used separately. While the short-term profile can be used to produce context relevant recommendations (context of use; e.g., *more like this* recommendations, see Swearingen and Sinha (2002)) the long-term profile can be used for generating more personal recommendations (e.g., *hot items* recommendations; *new items within preferred genres*, as proposed by Swearingen and Sinha (2002)).

6.4.4. Item Profile: Initialization and Refinement

Similar to the preference dimensions of user profiles also attribute values of items can be modeled by using dimensions with the capability of a weighted value representation. These dimensions are used to model affiliations/associations of items to specific concepts, such as clusters, and are used for the item profile refinement process. The fact, that an item i (e.g., book, song, etc.) is affiliated to a given cluster c (e.g., genre) to a certain extend e can be modeled by using the identifier of c as the dimension's *Value* object and e to initialize the *Characteristic* object, having a value between a minimum and maximum value (e.g., -1 to 1) representing the different degrees of the item's prototypicality concerning cluster c.

Typically the item profile will be initialized by a domain expert – affecting the domain expert view – either by manually classifying the items or by using third party information as delivered by content providers. Item descriptions, allocated by content providers (e.g., labels such as Sony, BMG, etc., in the music domain), are imported/uploaded into the portal for being presented to the user. In domains with a huge amount of items (e.g., music) this item description often has a poor quality.

Music genres, for example, are an often disgraced concept, but indispensable to a music portal. The most serious problem with genre is that they are not standardized and that they tend to be a source of dispute. When it comes to music styles the AllMusicGuide[5] offers 531, Amazon.com[6] 719 and MP3.COM[7] about 430 different genres (Uitdenbogerd and van Schnydel, 2002). Furthermore, music content providers often do not or cannot deliver appropriate genre information: So during the series of R&D projects accompanying the *Ericsson's Media Suite - Music* project some providers did not deliver genre information for about 20% of their content.

Appropriate strategies for portal operators to improve the item meta data quality are using classifier systems or by involving the community. Classifier systems, especially important in domains with large content sets, can be used by content administrators to generate additional meta data of items – affecting the *classifier view* of an item – either for compensating a missing administrator view as well as for providing alternative approaches to the item space. Applying a classifier on content data is a typically recurrent process performed by the content administrator during maintenance of the item's life cycle (e.g., uploading/removing of items).

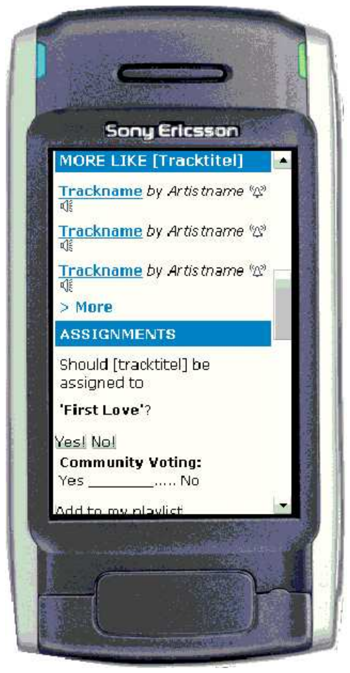
So, for example, in the context of the *Ericsson's Media Suite - Music* project an audio classifier system was used for finding similar sounding

[5]`http://www.allmusic.com` as of 11/2008
[6]`http://www.amazon.com` as of 11/2008
[7]`http://www.mp3.com` as of 11/2008

songs automatically (Aucouturier and Pachet, 2004; Pampalk et al., 2003; Aucouturier and Pachet, 2002), only based on signal analysis of the provided audio files. The generated meta data was used to create play lists – songs, that sound similar to a given one – as well as for providing mood specific, alternative genre-clusters (e.g., ’sad songs’).

Feeding portals regularly with interesting and engaging content is a very challenging and of course expensive task. The Adaptive Personalization approach tries to tackle this problem by involving the users in this process by encouraging them to post affiliations of items to predefined clusters[8]. The community feedback was used to maintain and refine the content of these clusters once defined and initialized by an administrator.

The community-assessment view is initialized as an empty profile and is refined by the feedback of the users concerning the classifications of items. In *Ericsson’s Media Suite - Music* the user can give feedback about the cluster affiliation of items such as tracks, artists, etc.,



MORE LIKE [Tracktitel]

Trackname by Artistname

Trackname by Artistname

Trackname by Artistname

> More

ASSIGNMENTS

Should [tracktitel] be assigned to

'First Love'?

Yes! No!

Community Voting:

Yes ________..... No

Add to my playlist

Figure 6.7.: Assigning a Track to a Cluster

The sample presented in Figure 6.7 on page 96, taken from *Ericsson’s Media Suite - Music*, demonstrates a use case of a mobile music portal

[8]The use of predefined clusters instead of tagging – as used in Web 2.0 approaches – was chosen in respect to the importance of the mobile channel and its limitation concerning user interactions.

where users can assign tracks to predefined music clusters according to moods and/or situations. In Figure 6.7 on page 96 the user is asked, if he/she would assign the given track to the music cluster *First Love*. Based on the different feedback (YES/NO) the track will be added to or removed from the specific cluster.

Based on these three different views an intensity value function $val(i, c)$ for defining the affiliation of an item i to a cluster c (or concept) can be defined as the combination of the administrator assignment, the community affiliation and the computation of a classifier system as described in Definition 6.18

$$val(i,c) = w_{adm}val(i,c)_{adm} + w_{com}val(i,c)_{com} + w_{sys}val(i,c)_{sys} \tag{6.18}$$

where $val(i, c)_{adm}$ is the affiliation intensity assigned by the administrator, $val(i, c)_{com}$ the overall assessment of the community and $val(i, c)_{sys}$ the assignment calculated by a classifier system. The three different weights w_{adm}, w_{com} and w_{sys} are used to define the importance of each view to the overall affiliation. According to the preference dimensions of user profiles discussed above, the following constraints are applied

$$\begin{array}{ll} -1 \leq val(i,c)_k \leq 1 & \\ 0 \leq w_k \leq 1 & k \in \{adm, com, sys\} \\ \sum w_k = 1 & \end{array} \tag{6.19}$$

to guarantee, that the value of $val(i, c)$ is normalized between the range of -1 and 1. Similar to the preference dimensions of user profiles, thresholds T_{neg} and T_{pos} are defined for specifying the value range of $val(i, c)$ where the item-cluster affiliation have to be considered by the system. According to the explanation above, we will focus only on T_{pos} denoted simply as T for ease of discussion. So, the affiliation of an item i to a given cluster c is effective, when $val(i, c)$ is above a given threshold T. More formally:

$$i \in c | val(i,c) \geq T \tag{6.20}$$

The most interesting part of $val(i, c)$ as defined in Definition 6.18 is $w_{com}val(i, c)_{com}$, describing the affiliation information contributed by the user community. This term is highly dynamic compared to the administrator's or classifier's view which are often only used for initializing an item's association to a given cluster by using specific values aff_{adm}, aff_{sys} (e.g., during the upload/import process of content items) – see Definition 6.21.

$$val(i,c) = w_{adm}aff_{adm} + w_{com}val(i,c)_{com} + w_{sys}aff_{sys} \tag{6.21}$$

In contrast, the community-assessment $val(i,c)_{com}$ is based on the number of submitted items – cluster associations performed by different users of the community. Concerning this community feedback the questions

1. how many user associations are necessary so that $val(i,c) \geq T$ for an item i having no other classification information ($val(i,c)_{adm} = 0$ and $val(i,c)_{sys} = 0$)?
2. how many user associations are necessary to overrule a given item-cluster initialization?

are highly relevant to the administrator or operator of a personalization system. While the first question is addressing the problem of items having no appropriate meta data the second one is focusing the problem of target group relevant meta data. Answering both questions appropriately will help the administrator to gather up-to-date meta data for his content.

In the Adaptive Personalization approach, the community based item-cluster association is modeled within dimensions where the number of positive and negative user associations are stored as the basis of the affiliation intensity calculation. Furthermore, each user can only submit one item-cluster affiliation for a given item i and a specified cluster c to avoid manipulation – as a result, it is reserved to registered users[9]only!

The intensity of this item-cluster association is defined by a sigmoid function as defined in Definition 6.22

$$aff(x) = \frac{2}{1+e^{-x}} - 1 \tag{6.22}$$

having a course characteristic as shown in Figure 6.8 on page 99
with the number of item-cluster affiliations submitted by the users as the unit of the x-axis (absolutely or relatively). Furthermore, the inverse function – used to define x given a specific value for $aff(x)$ – is defined as

$$aff^{-1}(y) = -ln(\frac{2}{y+1} - 1) \tag{6.23}$$

Equipped with these definitions we are now able to formulate the answers for the questions above. First, an appropriate threshold T must be defined so that n positive user affiliations – stressing the fact, that item i is element of cluster c ($i \in c$) – will activate that affiliation.

[9]A user is registered when he/she passed through an identification process, e.g., the typical 'registration' use case provided in many portals.

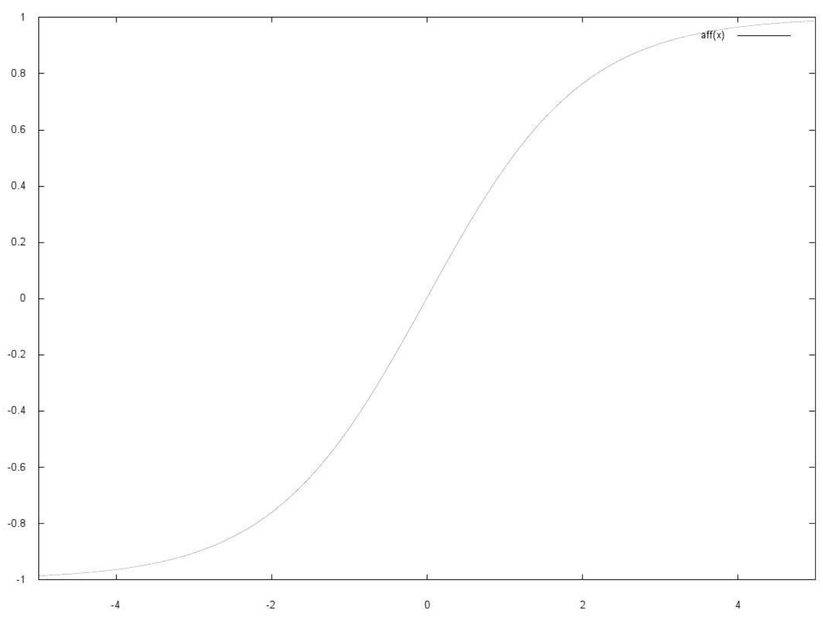


Figure 6.8.: Item-Cluster Affiliation Characteristic

So, for example, based on certain domain or user group specifics an administrator can decide that the appropriate course characteristic of $aff(x)$ for T is at $x = 2$ and that at least $nr = 15$ (positive) user affiliations are necessary to reach that point. Furthermore, the different weights are defined as follows: $w_{adm} = 0.6$, $w_{com} = 0.4$ and $w_{sys} = 0$ indicating that no classifier system is available. Based on these requirements and the equations above T is defined as $T = w_{com}aff(2) = 0.30$. The positive conversion factor f_{pos}, concerning the number of item-cluster affiliation submitted by the community for the x-axis unit, is defined as $f_{pos} = \frac{x}{nr} = \frac{2}{15} = 0.133$.

Furthermore, the administrator can define, that at least 25 contrary community affiliations are necessary to neutralize his item-cluster association. So the negative conversion factor f_{neg} can be defined as follows:

$$w_{adm}aff_{adm} - w_{com}val(i,c)_{com} \leq T \tag{6.24}$$

and can be derived to

$$val(i,c)_{com} \geq \frac{w_{adm}aff_{adm} - T}{w_{com}} \tag{6.25}$$

Applying the value of $val(i,c)_{com}$ to the inverse affiliation function defined in Definition 6.23 on page 98 will result in an x-value which can be converted to the necessary number of user feedback.

Given the maximum intensity $val(i,c)_{adm} = aff_{adm} = 1$ an administrator can assign to an item-cluster relation and the variables defined above, the value of $val(i,c)_{com}$ can be calculated as

$$val(i,c)_{com} \geq \frac{w_{adm} aff_{adm} - T}{w_{com}} = \frac{0.6 \cdot 1 - 0.30}{0.4} = 0.75 \tag{6.26}$$

By applying this value 0.75 to Definition 6.23 on page 98 we get

$$x = aff^{-1}(y) = -ln(\frac{2}{0.75+1} - 1) = -ln(0.14) = 1.95 \tag{6.27}$$

In a next step we can define f_{neg} as $f_{neg} = \frac{x}{nr} = \frac{1.95}{25} = 0.08$.

Summarizing we get the following results

- the value for the threshold T is set to 0.30
- the value for the positive conversion factor f_{pos} for mapping positive item-cluster affiliations (stressing $i \in c$) to the x-axis of the affiliation function as defined in Definition 6.22 on page 98 is set to 0.13
- the value for the negative conversion factor f_{neg} for mapping negative item-cluster affiliations (stressing $i \notin c$) to the x-axis of the affiliation function as defined in Definition 6.22 on page 98 is set to 0.08

6.5. Evaluation

Due to severe back-up and operational problems of the referred real world systems[10] resulting in loss of data, only partial aspects of our profile model could be evaluated. However, the results and conclusions presented in this section should be seen as evidences for the argued value-add. The full verification of the effectiveness of our profile model will be adduced in future work.

The strength of the Adaptive Profile Model is its multi-view approach, incorporating contextual information concerning the origin of a user action, combined with the ability to define the adaptive or learning behavior precisely. Based on the fact, that this adaptive behavior is limited to predefined clusters or concepts (for which an appropriate dimension has to be defined) this approach should be seen as an extension or supplement to standard profile approaches relying on machine learning techniques. Consequently a qualitative analysis approach was chosen for proofing its

[10]User feedback got lost because log tables were removed for saving disk space.

effectiveness, based on real world data instead of the comparison with standard techniques.

The analysis performed in this section is partially an anticipation of the examinations of Chapter 8 focusing on profile issues only. For proofing the effectiveness of the Adaptive Profile Model answers to the following questions were derived from real world data sets:

1. Can the argued value-add of the multi-view concept be observed in the context of *user* profiles and what is its effect?
2. Can the argued value-add of the multi-view concept be observed in the context of *item* profiles and what is its effect?

6.5.1. Description of Data and Applied Procedures

The data for this analysis was taken from two different instances of the *Ericsson's Meduia Suite - Music*, being hosted in Europe and in Malaysia:

- A copy of the databases from the European server was taken at the 05/31/2006, covering a period of user actions ranging from 9/2005 to 5/2006 – about 9 months.
- A copy of the databases from the Malaysian server was taken at the 05/16/2006, covering a period of user actions ranging from 10/2005 to mid 5/2006 – about 7.5 months.

Among other's, the databases contained the following information (for more details please see Chapter 8):

- User profiles: User descriptions consisting of attributes like unique identifier, age (optional), gender (optional), genre preferences, etc.
- Item profiles: Descriptions of items e.g., artist, tracks, products, etc., consisting of attributes such as identifier, type, description, genre affiliation, price, etc.
- User action: A history of user actions, stored in a specific database table
- System behavior: A history of system reactions, such as responses to recommendation requests, stored in a specific database table

These two sets of databases where installed locally on a MySql-Server. Beside some necessary *repair tables* no extra manipulation of the data was performed. Beside these two databases, no further information source was taken.

Furthermore, as explained in Chapter 8, the community view was not implemented for the user profile in this version of the *Ericsson's Meduia Suite - Music*, so the user related analyzes is restricted to the self-assessment and observation view.

6.5.2. Result

Based on examinations carried out in Chapter 8 the conclusion can be drawn, that the multi-view concept is successful/useful, because there are significant differences between the self-assessment and the observation view.

A significant difference between the genre preferences stored in the self-assessment view and the observation view could be found in the context of **user profiles**. So, on the basis of the preferences stored in the self-assessment view, the observation view accounts for 37% of additional/other preferences in the European installation and about 75% in Malyasia. Furthermore, 29% of the users in EU and 42% in MY get more/other recommendations due to the observation view than they would get only based on the self-assessment.

Concerning the cluster affiliation of **items**, significant differences between these two installations could be observed. While in Europe the user-based classification of items was hardly used, it was well accepted in Malaysia, where 43% of all artists and 16% of all tracks where classified by the community! By comparing the administrator's contribution to the item affilition with the associations carried out by the community, we found that the users were responsible for 96% of all artist- and 81% of all track classification in Malysia!

6.6. Contributions to the Research Question

The concepts of the *Adaptive Profile Model* presented in this chapter are directly addressing the second sub-question as defined in Section 1.2 on page 4.

> *How can contextual information be used to improve user models?*

Based on a psychological model of Joseph Luft and Harry Ingham - the Johari Window (Luft and Ingham, 1955) – a multi-view profile model was developed introducing the *context of origin* of user feedback. Concerning user profiles the three views – *self-assessment*, *system observation* and *community-assessment* – are used for optimizing the recommendation process for satisfying the well and ill-defined needs as well as for comprehensible explanation models. Furthermore, the incorporation of long- and short-term aspects of preferences (context of sessions) will help to build more stable user models, because context-driven variations of a user's behavior (e.g., searching for a gift for a friend) can be covered much better. Evaluations performed on data gathered from real world systems (see Chapter 8) showed strong evidences concerning the effectiveness of this multi-view approach, by identifying a significant set of non-overlapping preferences concerning the *self-assessment* and *system observation* view.

Additionally, the presented three fold item model supports the content administrator in maintaining valuable content meta data which will lead to better recommendations.

7. Recommendation Algorithms

The determination of similarity is a core task recommender algorithms try to solve efficiently. While collaborative filtering approaches establish user similarity relations based on ratings for generating suggestions, content-based techniques rely on the similarity of item attributes being preferred by users. In this chapter we will present two new algorithms for both of these research fields.

In the first section a new collaborative filtering algorithm is presented where the contextual importance of items and the relations of preference sets are combined in a new way for an improved similarity definition of users. The motivation for developing this approach was not to use additional data (e.g., *time* in Adomavicius et al. (2005)) to improve standard algorithms but the need to derive as much information as possible from the data provided. Existing approaches such as *Pearson Correlation* or *inverse user frequency* (Breese et al., 1998) do not meet this requirement adequately.

In the second section a new data structure, called D^2-Tree, is presented for solving the *k-nearest neighbor* (kNN) problem in a search space spanned by discretized attributes of profiles, a typical use case in the context of content-based approaches. The development of the D^2-Tree was mainly driven by the requirement to provide the weighting of search criteria attributes for each single request which is not supported by search structures such as K-D-B trees (Robinson, 1981).

7.1. The Pretty Good Recommendation Family

In general, collaborative filtering recommender algorithms try to predict ratings of items for a given user from a database of ratings of other users (see Chapter 3). Following the definition of Breese et al. (1998) the core procedure for predicting a rating r_{ui} of user u for an item i in memory-based approaches can be defined as

$$r_{ui} = \overline{r_u} + \kappa \sum_{v=1}^{n} w(u,v)(r_{vi} - \overline{r_v}); \quad u \neq v \tag{7.1}$$

where $\overline{r_u}$ is the average rating of user u, κ a normalizing factor, n the number of users stored in the database and – most important – the weight $w(u, v)$, expressing the kind of relation between two users u and v (e.g., *similarity, correlation*). Several different strategies such as *Pearson Correlation* see Definition 3.2 on page 23, *inverse user frequency* see Definition 3.4 on page 24, etc. exist for defining this weight.

Our main criticism concerning these strategies is, that their computation is too simple and that important aspects concerning the complex relations among users are ignored. Most of these approaches are only using the set of common ratings of two users u and v as a basis (e.g., Pearson Correlation, vector similarity) sometimes extended with further information as done in *default voting* or *inverse user frequency*. Having a closer look at the *Pearson Correlation*, the most common relation measure (see Definition 7.2), will demonstrate these shortcomings.

$$w(u, v) = \frac{\sum_{i=1}^{n}(r_{u,i} - \overline{r_u})(r_{v,i} - \overline{r_v})}{\sqrt{\sum_{i=1}^{n}(r_{u,i} - \overline{r_u})^2(r_{v,i} - \overline{r_v})^2}} \tag{7.2}$$

As a consequence of Definition 7.2 all users having the same relative ratings ($(r_{v,i} - \overline{r_v})$) will get the same relation weight $w(u, v)$ – which is not a proper result.

But what are the characteristics of a good user-user relation? Which aspects, beside the obvious rating values, should be considered? A simple thought experiment will help us to answer this question basically.

Given a database containing favorite items I_j of users ignoring, for simplification, the different levels of liking. Furthermore, we assume the existence of 4 users, each having a set of preferred items U_k ($k = 1..4$) – identified as ellipses – as shown in Figure 7.1 on page 107. As as result, a ranked list of most similar users for U_4 should be determined.

Not surprisingly, U_3 is most similar to U_4, because both users share the same set of preferred items ($U_4 = U_3$). In other words, U_3 can be seen as an exact clone of U_4 implying perfect similarity. Consequently, the second best match would be a clone taken from the past, having one preferred item less. In our example, U_2 can be seen as this clone from the past, because U_2 is a real subset of U_4 ($U_2 \subset U_4$). The weakest similarity relation exists between U_4 and U_1 – although having the same intersection with U_1 ($U_4 \cap U_1 \neq \{\}$) as U_2 – because each of them has preferences, missing by the other. So, by only considering the kind of intersections of the preference sets, we could identify the following similarity order:

$$sim(U_u = U_v) > sim(U_u \subset U_v) > sim(U_u \cap U_v \neq \{\}) \tag{7.3}$$

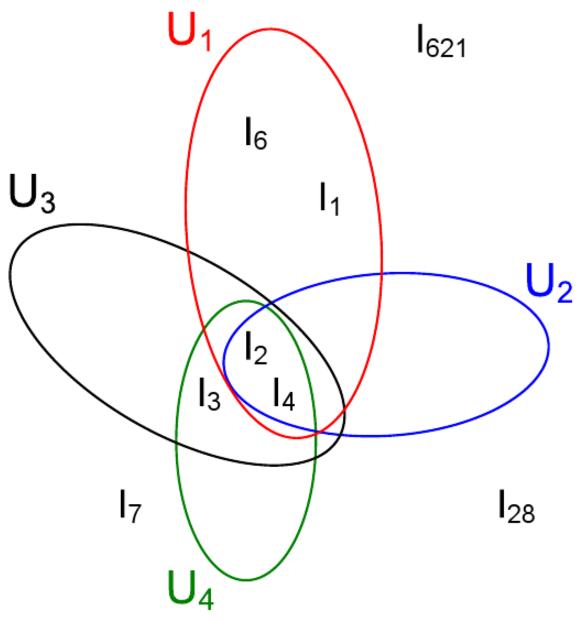


Figure 7.1.: Favorite Itemsets of Users

The inclusion of this kind of information in the process for defining the best matching users is one major improvement of our algorithm – but still further conclusions can be drawn from our sample.

Concerning the characteristics of an item rating, used for the definition of like minded users, the following factors can be identified: (i) the strength of the rating, (ii) its popularity concerning all users and (iii) its importance for a specific user. By skipping the obvious first aspect, it is evident that items being everybodies-darling, are less helpful than rare ones. So, for example, item I_2 being preferred by all 4 users is contributing nothing to the definition of a ranking or sort order of users. The *inverse user frequency*[1](Breese et al., 1998) technique, an analogy to the *inverse document frequency* as used in information retrieval (Heyer et al., 2006), is somehow addressing aspect (ii) by applying weights, based on the rating frequency, to the users ratings.

While this approach is addressing the discriminatory power of an item rating concerning all users its local importance, stressed by aspect (iii), is still missing. The underlying assumption is, that the importance of an item within a preference set decreases with the size of this set, a concept borrowed from the field of *information retrieval* where it is known as the *term frequency*.

[1]$\log(\frac{n}{n_j})$ where n is the number of users and n_j is the number of users having rated item j

Now armed with the consideration above, we are able to identify the parameters important to consider when defining a similarity measure $w(u, v)$ concerning two users u and v:

1. The similarity and strength of item ratings concerning elements out of E where $E = U_i \cap U_j$
2. The similarity of U_i and U_j: The smaller the number of items out of $U_i \cup U_j$ not being part of the intersection $E = U_i \cap U_j$ – the better. Best if $U_i = U_j$
3. The size of the intersection set $|E| = |U_i \cap U_j|$. The larger this intersection – the better
4. The discriminatory power and the local importance of elements of E: The higher the discriminatory power and the importance of elements out of E the better.

While some of these factors are quite obvious and are already used by certain collaborative filtering techniques (Herlocker et al., 2000; Breese et al., 1998) the real innovation is introduced by how we are dealing with item 2 and 4!

After having defined the set of relevant users v_i new item suggestions for a given user u can be generated on the basis of the user ratings and the similarity correlation between u and v_i. In the following sections we provide the formal basis for the definition of an appropriate user correlation and how appropriate items are calculated upon this set of users. Furthermore, our approach is evaluated by comparing its performance with that of standard algorithms. Thus, the basic idea that a smaller set of more relevant users will produce better results than a larger set with less quality, will be proofed.

7.1.1. Finding Appropriate Users

In this section the definitions are introduced for defining an appropriate similarity correlation. The preferences – positive or negative – of users u_k, concerning items i_j are expressed by rating tuples of the following form:

$$R_{ui} = \langle u_k, i_j, r_{ui}, time \rangle \, ; r_{ui} ... user\ rating; \ r_{min} \leq r_{ui} \leq r_{max} \tag{7.4}$$

Where r_{min}, r_{max} are the minimum/maximum ratings a user can assign to an item. Furthermore, the set of preferred items P of a user u is defined by

$$P_u = \{i | i \in R_{ui}\} \tag{7.5}$$

and the set of evaluators of an item i (e.g. users preferring a track or artist) is defined by

$$F_i = \{u | i \in P_u\} \tag{7.6}$$

Furthermore, we define the relative popularity pop of an item i as

$$pop(i) = \frac{|F_i|}{|U|} \tag{7.7}$$

where $|U|$ is the number of all users. The discriminatory power/factor $disc$ of an item i is defined on a basis of the inverse of the popularity pop – also known as *inverse user frequency* – of an item i.

$$disc(i) = -\lg_2(pop(i)) \tag{7.8}$$

Applying this function will lead to (i) a minimum value "zero" when the popularity is maximal – e.g. when item i is liked by all users – and (ii) to a maximum when only few users prefer item i (infinity when no user likes i). Additional to the normalized standard rating

$$rate(u,i) = \frac{r_{ui}}{|r_{max} - r_{min}|} \tag{7.9}$$

we introduce the definitions for *relative rating* and *adjusted rating* for evaluation purposes concerning different algorithm variants. Relative ratings are used by standard algorithms such as *Pearson Correlation*. The normalized, *relative rating* of an item i by a user u is defined:

$$rate_{rel}(u,i) = \frac{r_{ui} - \overline{r_u}}{|r_{max} - r_{min}|};\ \overline{r_u}...average\ rating\ of\ user\ u \tag{7.10}$$

One drawback of the relative rating is, that it tends to become zero, especially in the context of few or equal ratings. To avoid or soften this effacement we introduce the *adjusted rating* by adding the normalized rating to the relative rating:

$$rate_{adj}(u,i) = rate_{rel}(u,i) + rate(u,i) \tag{7.11}$$

By using Definition 7.11 the original rating classification (e.g. good, neutral, bad) is, adjusted with the deviation from the average rating, reintroduced in the calculation. The impact of Definition 7.11 is shown in a a more complex use case below.

Sample: Assuming we have a 5 score rating scale, ranging from 1 (very bad), 2 (bad), 3 (neutral), 4 (good), to 5 (very good) and two users A and B with average ratings $avg_A = 1$ and $avg_B = 4$, furthermore both users rated an item i with $rate(A, i) = 2$ and $rate(B, i) = 5$. For both users A, B the relative rating for item i has the same value: $rate_{rel}(A, i) = 1/4$ and $rate_{rel}(B, i) = 1/4$! By using the adjusted rating we get: $rate_{adj}(A, i) = 1/4 + 2 = 2.25$ and $rate_{adj}(B, i) = 1/4 + 5 = 5$, cutting off boundary overflows.

In further discussions we use $rate$ as a placeholder for one of the three possible rating variants.

Now having defined some different versions of item ratings we are able to define the rating difference of two users u and u concerning a set of common rated items by calculating the *weighted mean*

$$rate_{Diff}(u, v) = \frac{\sum_{i=1}^{n} (rate_{ui} - rate_{vi})\, disc(i)}{\sum_{i=1}^{n} disc(i)} \tag{7.12}$$

This rating difference, describing the average deviation concerning common ratings, will be used to calculate predictions for new items for a given user. Next we introduce the local importance imp of an item i for a user u as the ratio of the rating and the size of his/her preference set. The importance $imp(u, i)$ is defined as:

$$imp(u, i) = \frac{r_{ui}}{|P_u|} \tag{7.13}$$

By replacing the item rating r_{ui} with the relative or adjusted rating we get the appropriate (relative/adjusted) importance value. With the definitions above we are now ready to define the quality q of the similarity (correlation) between two users u, and v with

$$q(u, v) = \sum_{i=1}^{n} (imp(u, i) imp(v, i) disc(i));\ i \in P_u \cap P_v,\ n = |P_u \cap P_v| \tag{7.14}$$

With Definition 7.14 we are addressing factors 1, 3 and 4 we defined above – but 2 is still missing!

As mentioned above, the considerations behind factor 2 are that the measure of the overlapping of two preference sets $Pref_A$, $Pref_B$ has an effect on the similarity of two users A and B. So, for example the fact, that $Pref_A$ is a subset of $Pref_B$ implies a stronger similarity between A and B than an intersection $A \cap B$. Furthermore, this implies that the similarity relation between A and B is a directed relation.

But once more, we can borrow some concepts from other research fields. A quality measure concerning *overlapping* can be found in the domain of association rule mining (ARM) (Agrawal and Srikant, 1994; Park et al., 1995). Based on a number of sets (baskets/transactions), each containing some items (goods), ARM tries to find significant item relations of the form $X \rightarrow Y$. ARM is a standard algorithm for shopping cart analysis. The most important parameters in ARM are *support* and *confidence*. The *support* describes how often an item (or set of items) can be found in different sets and is defined as:

$$sup(X) = \frac{number\ sets\ containing\ X}{number\ all\ sets};\ X ... a\ set\ of\ items \tag{7.15}$$

The *confidence* is used to describe the quality of a relation $X \rightarrow Y$ and is defined as

$$conf(X,Y) = \frac{sup(X \cup Y)}{sup(Y)} \tag{7.16}$$

By using the sets F_i containing the evaluators/fans of an item as our sets for the ARM algorithm where X and Y are sets of users (fans) of size 1, we can use $conf(X,Y)$ as an overlapping measure of the preference sets.

So the similarity correlation between two users u and v – $sim(u,v)$ – can now be defined as

$$sim(u,v) = conf(u,v)q(u,v) \tag{7.17}$$

7.1.2. Predicting Items

In contrast to other algorithms (Breese et al., 1998; Lemire and Maclachlan, 2005) the item prediction is based on two similarity measures: The rating difference $rate_{Diff}$ (see Definition 7.12 on page 110) on the one hand and the similarity correlation Definition 7.17 on the other. The prediction value $p(u,v,i)$ of an item i for user u – not having rated this item in advance – is computed on the rating of user v in the following way:

$$p(u,v,i) = rate_{vi} + rate_{Diff}(u,v) \tag{7.18}$$

By applying this pairwise prediction value on a list of similar users the overall prediction value $pred$ of the item i for the user u can be defined. Given a user u we can create a sorted list L_u – corresponding to the similarity weight/correlation value – of the most similar users by applying Definition 7.17 on page 111 to a set of users. The most similar users (e.g. the first n elements of list L_u) of u are used to compute the prediction value for a given item i, based on the weighted rating differences

$$pred(u,i) = \frac{\sum_{v=1}^{k} (p(u,v,i) sim(u,v))}{\sum_{v=1}^{k} sim(u,v)} \tag{7.19}$$

7.1.3. Evaluation

The algorithm was tested on the GroupLens[2] 100k Movielens data set using an averaged five fold test based on the available, prepared datasets. This choice was made to produce comparable results which can be used/referred by other researchers in the community.

Data set

The used data set for the evaluation, is one of the most used test data sets for CF-algorithms in the recommender research community provided and managed by GroupLens. The characteristics of this data set are:

- it contains 100.000 ratings from 943 users on 1682 movies
- the ratings range from 1(worst) to 5(best)
- each user has rated at least 20 movies
- 5 prepared (disjoint) data sets are provided for running 5-fold tests with a training base consisting of 80% and a test set having 20% of the overall ratings

Additional some simple demographic user information such as age, gender, occupation, postal code, etc. is provided.

Test procedure

For producing comparable results we used the five prepared data sets *u1.base*, ..., *u5.base* for training purpose and the corresponding test data *u1.test*, ..., *u5.test* for evaluation in the course of an 5-fold cross validation test. As a test metric we used the *Mean Absolute Error* (MAE) – the standard metric for evaluating collaborative filtering algorithms – to determine the prediction quality of the algorithms.

The MAE is defined as the average difference between the predicted ratings and the real user ratings, as defined within the test sets. Formally, MAE can be defined as:

[2] `http://www.grouplens.org` as of 11/2008

$$MAE = \frac{\sum_{i=1}^{N} |p_i - r_i|}{N} \tag{7.20}$$

where p_i is the predicted value for item i and r_i is the user's rating.

For evaluating the improvements achieved by our approach we performed a 5-fold cross validation test upon the $n\%$ most similar users (concerning the complete list of similar users), ranging from the best 2% up to 100% (all users). Furthermore, these tests were performed with 3 variants of the *Pretty Good Correlation* (PGC) based on standard rating, relative rating and adjusted rating. Additional an implementation of the Pearson algorithm, as presented in (Resnick et al., 1994), was taken as the reference implementation.

Test results

As shown in Figure 7.2 and Figure 7.3 on page 114[3] each of the three variants of the PGC family outperform Pearson by far, especially in the context of using only few users.

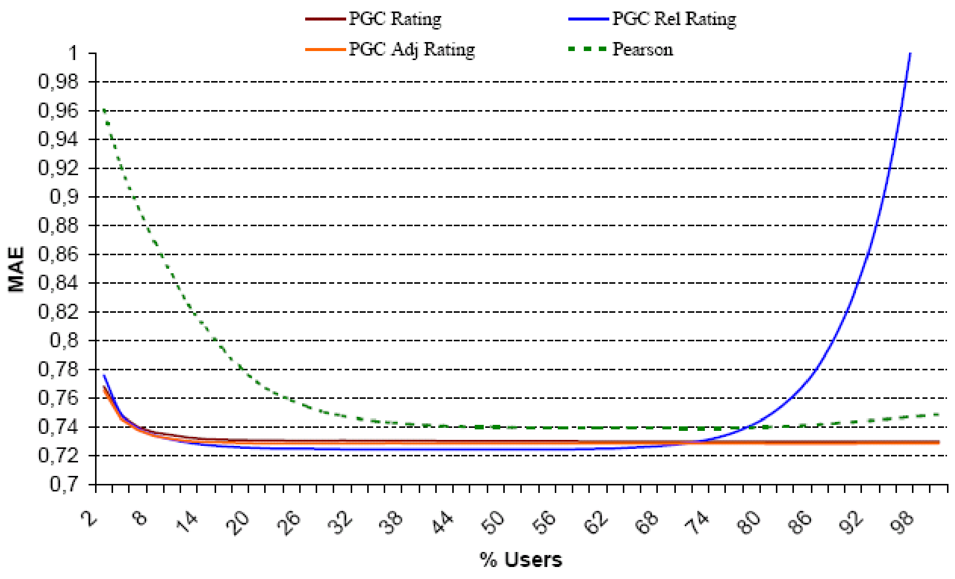


Figure 7.2.: Overview: MAE of PGC Algorithms and Pearson

The PGC algorithm using the relative rating – PGC Rel Rating – (see Definition 7.10 on page 109) showed the best performance with a minimum MAE of 0.7245 using 50% of the available users (a table containing the

[3]Both figures show the same data – the diagram denoted as *Zoom* has a more fine grained MAE scale

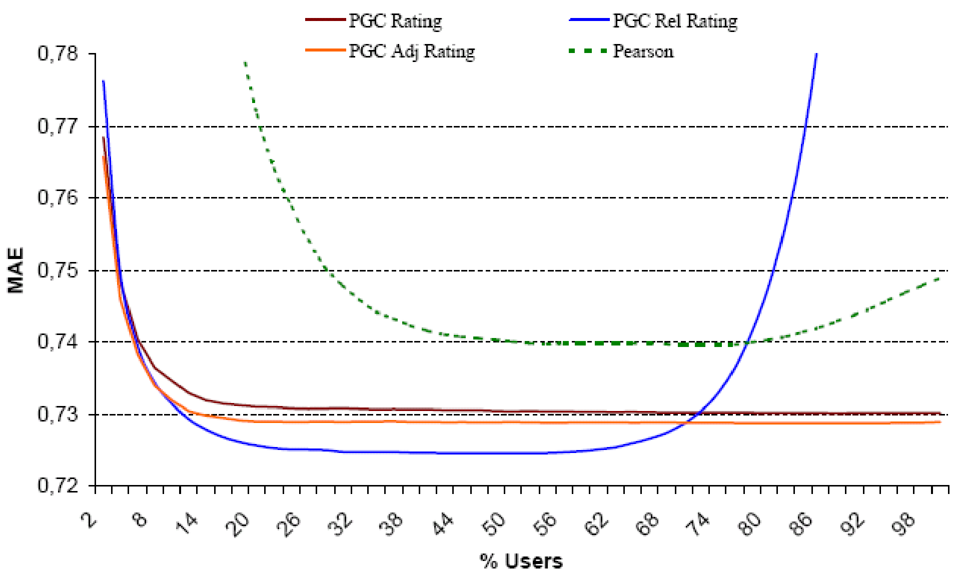


Figure 7.3.: Zoom: MAE of PGC Algorithms and Pearson

values can be found in Appendix A on page 179). Although having the minimal MAE values of all tested algorithms, the performance of *PGC Rel Rating* dramatically changes to worse, when a larger number of users – the more dissimilar onces – are taken into account! An explanation of this behaviour is, that this algorithm is very sensible to real similarity and can not deal very well with double false (opposite ratings of unsimilar users) contributions to the similarity correlation.

But the most interesting fact of the presented algorithms is their good behavior in the context of using only a limited number of similar users. All three variants of the PGC family have a MAE about 0.77 when using only the best 2% of the similar users which is about 25% better than the corresponding MAE of Pearson (having 0.96).

Furthermore, also other algorithms such as *Slope One* (Lemire and Maclachlan, 2005) are outperformed by our approach. In their paper Lemire and Maclachlan (2005) published MAE values about 0.752[4] for all three variants – Bi-Polar Slope One, Weighted Slope One, Slope One – of their algorithm, based on the Movielens 100k data set.

[4]Using the normalized value 0.188 – see (Lemire and Maclachlan, 2005), p. 474, 475

7.2. Profile-Based Approach: The D^2-Tree

One major concept of the *Adaptive Personalization* approach is the user driven creation and refinement of profile attributes. This aggregated information can be used for meta data based recommendations by finding profiles with similar attribute values. In this section a new data structure the *Discretized Dimension Tree* (D^2-Tree) is presented for solving the *k-nearest neighbors problem* in a high dimensional search space stretched by dimensions of profiles. The D^2-Tree is, similar as the K-D-B tree (Robinson, 1981), a combination of B-tree (Comer, 1979) and kD-tree (Procopiuc et al., 2002) concepts adopted for data mining problems where profiles with a high number of dimensions with limited number of discretized values form the search space. Unlike other trees for solving the k-nearest neighbors problem the structure of the D^2-Tree is not intended to reflect the distances between the profiles directly but rather provides the infrastructure for appropriate algorithms solving this task dynamically.

7.2.1. Requirements

Finding a set of similar items to a given one is a core task of a recommender system. While collaborative filtering approaches define user similarity based on item ratings other approaches, e.g., content-based filtering, rely on the similarity of common attribute values of the item descriptions. Application areas for the latter approach, being addressed by the D^2-Tree, are:

- Body shops, where the best matches between the skill profile of a user and job specifications have to be defined
- Gift finder, where item descriptions are matched against the preferences of a person
- Dating platforms, where personality descriptions must be compared to find the most harmonic results

Applications like these are often based on a profile system for modeling the items of interest and an appropriate matching functionality for defining the accordance of profiles. Typically these profiles consist of a variety of matching relevant attributes often having discretized value ranges (e.g., age-group, dress size, etc.). The search for best matching profiles to a given one can be described as the definition of the *k-nearest neighbors* in a search space spanned by the profiles attributes. Furthermore, the impact of an attribute concerning the result set must often be handled on an individual level at each request and in real-time too.

So an appropriate algorithm for solving the k-nearest neighbor problem in this context has to meet the following requirements:

1. The search space is spanned by an extensive number dimensions, corresponding to the attributes of the profiles
2. The value range of a dimension consists of discretized values
3. The definition of the k-nearest neighbors must be performed in real time
4. The impact of dimensions concerning the result must be supported on a request level (e.g. ignoring of a dimension, weighting of dimensions)

It is the last requirement why well-known data structures such as kD-trees (Procopiuc et al., 2002) or K-D-B trees (Robinson, 1981) cannot be used because these approaches build a rigid structure reflecting the similarity of profiles. Furthermore, the real time behavior is very important especially in the context of real world applications. In the next sections a new data structure, the D^2-Tree, is presented meeting the requirements defined above.

7.2.2. Definitions

For the consecutively considerations the following definitions will be used.

Profile: A profile is a description of an item consisting of a number of attributes and a unique identifier. As a sample, a user of a system can be modeled with the help of a profile containing attributes such as age, gender, name, date of birth, etc.

Dimension: A dimension implements a search criteria within the search space and consists of a unique identifier (e.g. dimension name), a type and a set of discretized values, describing all possible characteristics of this dimension. So, for example, the *dress size* of a person can be seen as a sample of a dimension, having the type *ordinal* and a possible value set of *small, medium, and large*.

Dimension type: The type of a dimension defines its discretized value range as well as the set of operations which can be performed on the value objects. Samples of possible types are integer, nominal, cardinal, set, etc.

Dimension value: Specific characteristics of a dimension are implemented by dimension value objects, consisting of one of the provided dimension values.

Difference function: For defining the difference of two dimension values u and v of dimension i a function $diff_i(u, v)$ is provided defining the normalized difference of two values. For ease of discussion, this function can be defined as

$$diff_i(u, u) = \frac{u - v}{|max_i - min_i|} \tag{7.21}$$

where max_i and min_i are placeholders for the maximum and minimum values of dimension i. Additional to these definitions a distance measure for defining the similarity of two profiles is introduced in Definition 7.22

$$dist(P, Q) = \sqrt{\sum_{i=1}^{n} w_i \cdot diff_i(P, Q)^2}; \quad w_i \geq 0; 0 \leq diff_i \leq 1 \tag{7.22}$$

describing the weighted Euclidean distance of two profiles P and Q where w_i is the weight of the i-th dimension and $diff_i(P, Q)$ is the difference function of dimension i.

Defining an appropriate difference function for each dimension is a core task in constructing the D^2-Tree, because with $diff_i$ specific domain knowledge is incorporated in the process of solving the k-nearest neighbors problem. Typically these functions will be carefully handcrafted with respect to the problem domain.

7.2.3. Structure

In contrast to common tree-based approaches for solving the k-nearest neighbors problem (e.g. kD-trees) the structure of the D^2-Tree does not directly reflect the similarity of two profiles but provides indexing mechanism and control information for solving this problem efficiently. A D^2-Tree is defined as follows:

1. The basic structure of a D^2-Trees is implemented by a B-Tree, where each layer of inner tree nodes corresponds to one dimension of the underlying search space

2. While the leaf nodes of the tree contain the profiles, the inner nodes only contain routing information to other subtrees. So, a D^2-Tree reflecting a n-dimensional search space consist of n+1 layers.

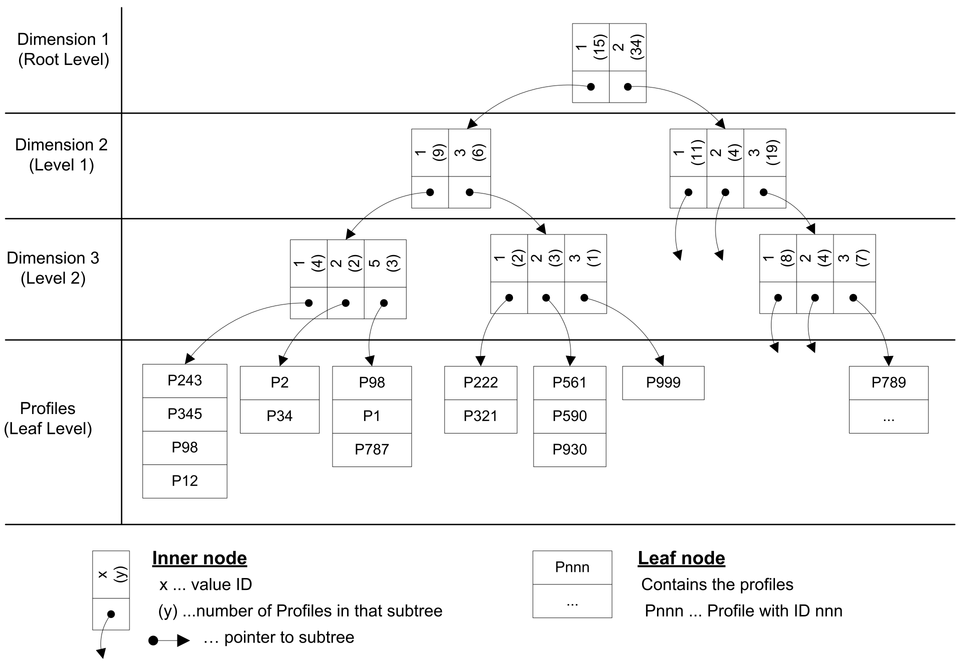


Figure 7.4.: D^2 Tree with 3 Dimensions

3. The more important a specific dimension is the nearer it is placed to the root node, which is modeled by the most important dimension. If no order of importance can be defined in advance, those dimensions with the highest discriminatory power concerning the distribution of profiles should be used to form the higher tree layers[5].

4. Inner nodes contain triples of the form $\langle valueID, (nrProfiles), \rightarrow\rangle$ where $valueID$ is an identifier for a value object of the corresponding dimension, $(nrProfiles)$ describes the number of profiles stored in leaf nodes available in that subtree the reference $\rightarrow$ is pointing to.

In Figure 7.4 a D^2-Tree with three dimensions is shown, being an example for a search space spanned by user attributes such as *age*, *height* and *weight*.

Although the D^2-Tree uses some concepts of B-trees and kD-trees there are several important differences to these structures:

1. Due to the fact that the value range consists of well-known, predefined values no splitting or merging of inner nodes is necessary based

[5]The root node is defined as the highest layer.

on insertion or deletion operations. So potentially cascading tree organizations can be avoided in contrast to B-trees derivatives such as B^+- or B^*-trees (Samet, 2005). The consequence of this advantage is not only an improved performance but also a better applicability in the context of persistent indexes for databases (e.g. reduction of locks).

2. The maximal depth and width of the D^2-Tree does not depend on the number of profiles being managed by this structure (in contrast to kD-trees!) but only on the number of dimensions and their value ranges.

3. A fast navigation through the tree structure is supported based on the hash-table like structure of the inner nodes. Based on the fact, that each reference to a subtree is identified by a value ID, these entries can be accessed very efficiently either by using hash structures or by supporting a direct indexing (array) in time critical applications.

4. By only storing the value IDs instead of the value objects within the inner nodes reduces the amount of allocated memory, especially in the context of memory intensive value objects (e.g. strings).

5. Due to the additional information concerning the number of profiles available per subtree – the (nrProfiles) attribute of the inner node triple $\langle valueID, (nrProfiles), \rightarrow \rangle$ – efficient operations for solving the kNN problem can be implemented (e.g. tree pruning). This is especially important in the context of recommender systems where typically a small but adequate number of recommended items should be presented to the user.

Providing appropriate difference functions $diff_i$ for each dimension is a key factor for the performance and applicability of D^2-Trees. Although each instance of a D^2-Tree can have its own set of difference functions, these functions are typically carefully modeled together with the corresponding dimensions thus forming the standard behavior of a tree dimension for a given domain. By providing explicit definitions of these $diff_i$ functions also complex domain knowledge can easily be incorporated, e.g. by using pre-calculated look-up tables.

An example for the definition of a dimension *gender* is presented in XML style in Listing 7.1.

```
<dimension name='gender' >
  <type  name='nominal'> </type>
  <value-range name='values'>
    <value ID='1'>male</value>
    <value ID='2'>female</value>
  </value-range>
  <diff-function name='genderDiff'>
    <rule value='0'>male-male | female-female</rule>
    <rule value='1'>male-female|female-male</rule>
  </diff-function>
</dimension>
```

Listing 7.1: Definition of a Dimension

This dimension with name *gender* has a *nominal* type with the value range *(male, female)* having the value Ids *1* and *2*. Here the difference function is declared as a simple rule, expressing that the difference function *genderDiff* will return 0 when both operands have the same value or 1 in the case that they are different. Accordingly a D^2-Tree can be defined by providing the following information (see Listing 7.2):

1. a mapping between profile attributes and search space dimensions
2. associations of the dimensions to layers of the tree
3. appropriate difference functions, based on the profiles attributes

```
<tree name="simpleUserTree" >
  <level nr='1'
    dimension='gender'
    attribute='Profile.getGender'>
  </level>
  <level nr='2'
    dimension='ageGroup'
    attribute='Profile.getAgeGroup'>
  </level>
  <level nr='3'
    dimension='dressSize'
        attribute='Profile.getSize'
        diff-function='diffDressSize'>
  </level>
  <root level='1' />

  <diff-function name='diffDressSize'>
   ...
  </diff-function>
</tree>
```

Listing 7.2: Definition of a D^2-Tree

Each level of the tree – identified with a number, starting with 1 at the root level – is defined by a dimension (e.g. *gender, ageGroup*), the corresponding attribute of a profile (e.g. Profile.getGender) and an (optional) difference function (here *diffDressSize*) to be used instead of the standard implementation, provided by the dimension itself.

7.2.4. Tree Operations

Having these basic principles of the D^2-Tree in mind, we will now have a closer look to the operations supported by this data structure. Basically, the following operations are provided by the D^2-Tree:

- insert: A profile is inserted into the tree. If the tree does not exist it will be created.
- remove: Removes a profile from the tree.
- get: Returns a list of profiles, exactly matching a set of given dimension values (search criteria).
- getNearestNeighbors: Returns a list of profiles best matching a set of given dimension values (search criteria).

For the following explanations we assume the existence of a profile system providing a set of profiles p_i together with the corresponding definition of the D^2-Tree (dimensions, difference functions, etc.). Furthermore, we assume that a mapping *dimension – tree layer* exists and that a root node is available. A *Java* like notation is used to formulate the algorithms.

insert

With method *insert(TreeNode node, Profile p, int level)* – see Listing 7.3 on page 122 – a profile p is inserted into the tree defined by $node$. This recursive function works as follows:

1. if *node* is a leaf node insert profile *p* in *node* and return
2. if *node* is an inner node check if a subtree for the appropriate profile attribute for the current dimension exists (find a triple $\langle valueID, (nrProfiles), \rightarrow\rangle$ where $valueID = p.attribute$)
3. if no subtree is found create a new node *n*, corresponding to the current level, and insert it into the current node
4. take *n* as the current node and proceed with step 1

In Listing 7.3 on page 122 a code snippet containing the definition of the method *insert* (see line 1 to 20) together with the calling sequence (line 23) is shown.

```
1  insert(TreeNode node, Profile p, int levelCnt)
2  {
3    wasInserted=false;
4    if(  node.isLeafNode() )
5      wasInserted=node.insertProfileIfNotExists(p);
6    else
7    {
8      valueId = getValueIdOf(levelCnt, p);
9      newNode = node.getSubTree( valueId );
10     levelCnt=levelCnt+1;
11     if( newNode==null)
12     {
13       newNode=createNodeForLevel(levelCnt);
14       node.addSubTree(valueId, newNode);
15     }
16     wasInserted = insert( newNode, p, levelCnt );
17     if( wasInserted )
18       newNode.incLeafEntryCounter();
19   }
20 }
21 ...
22 Profile p = ...;
23 insert(root, p, 0);
24 ...
```

Listing 7.3: Insert a Profile

The boolean variable *wasInserted* is used to increment the *nrProfiles* counter of each inner node triple along the insertion path over the tree dimensions. The counter is increased (line 18) only if profile p was inserted in a leaf node (line 5). Function *getValueIdOf(levelCnt, p)* returns the valueID of the value of the profile's attribute, corresponding to the dimension at level *levelCnt*. Furthermore, function *createNodeForLevel(levelCnt)* creates a new node for the tree level defined by *levelCnt*. Both functions are based on the relations $Dimension - ProfileAttribute - TreeLevel$ as described above.

remove

Method *remove(TreeNode node, Profile p)* removes the profile p from the D^2-Tree defined by *node*. This recursive function works as follows:

1. if *node* is a leaf node remove profile *p* and return
2. if *node* is an inner node check if a subtree for the appropriate profile attribute of the current dimension exists (find a triple $\langle valueID, (nrProfiles), \rightarrow\rangle$ where $valueID = p.attribute$)
3. return, if no subtree exists (profile p cannot be element of this tree)
4. take the subtree's root node as the current node and proceed with step 1

```
remove(TreeNode node, Profile p)
{
  wasRemoved = false;
  if( node.isLeafNode() )
    wasRemoved=node.removeProfileIfExists(p);
  else
  {
    valueId = getValueIdOf(levelCnt, p);
    nextNode = node.getSubTree( valueId );
    if( nextNode != null)
    {
      wasRemoved = remove(nextNode, p);
      if(wasRemoved)
      {
        if( nextNode.isEmpty() )
          node.removeSubTree( valueId );
        else
          nextNode.decLeafEntryCounter()
      }
    }
  }
}
...
Profile p = ...;
remove(root, p);
...
```

Listing 7.4: Remove a Profile

If no subtree (defined by *nextNode*) is found this method returns without changing the tree. If the profile could be removed successfully, indicated by the boolean variable *wasRemoved*, the counter for the available profiles of this subtree is decremented or the whole subtree is removed.

get

Method *get(TreeNode node, Profile p)* returns a set of profiles p_i, having the same attribute values as p. This recursive function works similar to *remove*.

1. if *node* is a leaf node, assign all profiles to the result set
2. if *node* is an inner node check if a subtree for the appropriate profile attribute of the current dimension exists (find a triple $\langle valueID, (nrProfiles), \rightarrow\rangle$ where $valueID = p.attribute$)
3. return, if no subtree exists (no profiles such as p can be elements of this tree)
4. take the subtree's root node as the current node and proceed with step 1

```
Set get(TreeNode node, Profile p)
{
  Set resultSet = null;
  if( node.isLeafNode() )
    resultSet=node.getAllEntries();
  else
  {
    valueId = getValueIdOf(levelCnt, p);
    nextNode = node.getSubTree( valueId );
    if( nextNode != null)
      resultSet= get(nextNode, p);
  }
  return resultSet;
}

...
Profile searchProfile = ...;
Set result = get(root, searchProfile);
...
```

Listing 7.5: Get Profiles

As mentioned above, this method returns all profiles p_i, having the same characteristics (concerning the attributes covered by the tree dimensions) as the search profile p.

getNearestNeighbors

The method *getNearestNeighbors(TreeNode root, Profil p, int minNrProfiles* – see Listing 7.6 on page 127 – implements the *k-nearest neighbor* search for the D^2-Tree structure defined by *root* and tries to return at least *minNrProfiles* profiles p_i being most similar to the given search profile p. The similarity of two profiles is based on the sum of the difference functions of the tree dimensions, according to Definition 7.22 on page 117.

Given a D^2-Tree with root node $root$, a search profile p and the constraint that at least *minNrProfiles* should be returned, the non-recursive algorithm works as follows:

1. S, the set of the current tree nodes, is initialized with the root node of the D^2-Tree
2. for each triple $\langle valueID, (nrProfiles), \rightarrow \rangle_j$ of all nodes in S (inner tree nodes) the difference d_{ij} of the *valueID* to the corresponding search profile attribute *p* of the current tree level i is defined by applying the appropriate difference function $diff_i(valueID, p.attributeValue)$.
3. define the minimal set of triples T having the minimal sum of all level differences ($\sum_{j=0}^{i} d_{ij}^2$) covering at least *minNrProfiles* of leaf entries ($\sum nrProfiles \geq minNrProfiles$).

4. if triples exist being not elements of T but having the same difference to the search profile p than these triples are also affiliated to T. This strategy should avoid, that promising subtrees are pruned too early in the calculation process. This situation can occur when a subtree, being pruned at level i due to a marginal greater difference value would provide far better results at level $i + k$ than elements of T.

5. all tree nodes being referenced by the triples of T are used to form the new set S ($S = T$)

6. if S consists of leaf nodes, the algorithms stops returning the profiles of the lef nodes in S as a result

7. if S consists of inner nodes proceed with step 2

In Figure 7.5 on page 126 this process is shown in the context of an example where (at least) 5 most similar profiles should be determined, given a search profile p with the values $< 1, 2, 1 >$ and a D^2-Tree having 3 levels of inner nodes. For ease of discussion we assume, that the *valueIDs* stored in the triples correspond to the referenced values and that the difference functions $diff_i$ of the three levels ($i = 1, 2, 3$) are only computing the absolute value of the attributes differences: ($diff_i = |valueID - p.attributeValue|$).

1. starting with the root node, the set T of the best matching subtrees is defined covering at least 5 leaf node entries (profiles). In our sample presented in Figure 7.5 on page 126 the left triple with $valueID = 1$ is taken, because of its minimal distance $d_{ij} = |1 - 1| = 0$ to the corresponding value of the search profile and the sufficient number of obtainable profiles (15).This subtree adorned with the current distance 0 to the search profile p is stored.

2. the same operation is now repeated for the node found in step 1 (left node of level 1 in Figure 7.5 on page 126) resulting in two subtrees having the same overall distance 1 (calculated as $\sum_{j=0}^{1} d_{ij}^2$) to the search profile p. Although each of the two subtrees cover a sufficient number of profiles (9 entries for the left and 6 entries for the right triple) both are taken in respect for finding better results.

3. by proceeding the search process, two nodes are now analyzed at level 2 (the first two nodes of dimension 3, see Figure 7.5 on page 126) resulting in two triples having the same difference value, pointing to leaf nodes.

4. now having reached the leaf level of the D^2-Tree the algorithm stops returning the leaf node entries as the result

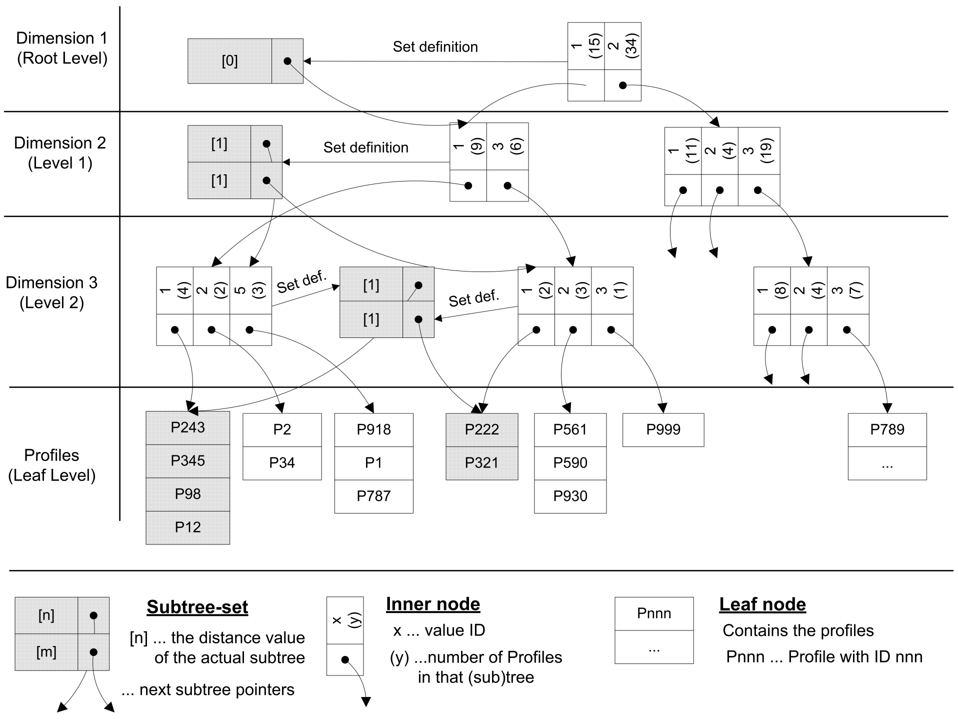


Figure 7.5.: Searching 5 Nearest Neighbors of $p = < 1, 2, 1 >$

The advantage of the strategy regarding the pruning of equivalent subtrees on a higher tree level (see step 2 at tree level 1) is shown in step 3. If only one of the two subtrees were taken at level 1, the result set would have contained less optimal profiles. So, for example, by selecting the left tree (triple: $\langle 1, (9), \rightarrow \rangle$) the result set would only contain 4 best matches (profiles P243, P345, P98, P12) and in the case of choosing the right node entry (triple: $\langle 3, (6), \rightarrow \rangle$) even only 2 best results (profiles P222, P321) were found. The most obvious drawback of this strategy is that enlarging the set of subtrees being analyzed will also rise the costs of computing results. A *Java* like code snippet, showing the algorithm in more detail, is presented in Listing 7.6 on page 127.

```
TreeSet getNearestNeighbors(TreeNode root, Profil p, int minNrProfiles)
{
  int      sumLeafEntries=0;
  TreeSet  actSubTrees = null;
  TreeSet  nextSubTrees = null;
  SetEntry entry = null;
  TreeNode node = null;

  nextSubTrees = createSubTreeSetForLevel( 0 );
  root.findBestSubTrees(p, 0.0, nextSubTrees);

  for(int levelCnt=0; levelCnt<nrTreeDimensions(); levelCnt++)
  {
    actSubTrees=nextSubTrees;
    sumLeafEntries=0;

    nextSubTrees = createSubTreeSetForLevel( levelCnt+1 );
    while( actSubTrees.hasNextEntry() )
    {
      entry = actSubTrees.nextEntry();
      node = entry.getTreeNode();
      sumLeafEntries = sumLeafEntries + node.getLeafEntryCounter();
      node.findBestSubTrees(p, entry.getDifference(), nextSubTrees);

      if( nextSubTrees.hasEnoughEntries(sumLeafEntries, minNrProfiles ) )
        break;
    }
  }

  return actSubTrees;
}

...
Profile searchProfile = ...;
TreeSet nearestNeighbours = getNearestNeighbors(root, searchProfile, 5);
...
```

Listing 7.6: Search Nearest Neighbors

The two methods *findBestSubTrees* and *hasEnoughEntries* are the core elements of this code sample. While the former function is responsible for the definition of the best matching subtrees (based on the algorithms described above) the latter implements the constraint of returning a minimal number of results. Both methods are proper starting-points for optimizations, because of their major impact on the performance behavior of the algorithm.

An important requirement defined above, the applicability of weights at each request, was not considered in the discussion so far. Based on the considerations presented above, the algorithm can easily be extended for meeting these requirements, by introducing a *weight vector* $\langle w_1, ..., w_i, ..., w_n \rangle$ containing individual weights for each dimension i being used for weighting the differences ($w_i d_{ij}^2$) within the method *findBestSubTrees*. An adapted code sample can be found in Listing 7.7 on page 128.

```
TreeSet getNearestNeighbors(TreeNode root, Profil p,
                            Vector weights, int minNrProfiles)
{
  int      sumLeafEntries=0;
  TreeSet  actSubTrees = null;
  TreeSet  nextSubTrees = null;
  SetEntry entry = null;
  TreeNode node = null;

  nextSubTrees = createSubTreeSetForLevel( 0 );
  root.findBestSubTrees(p, 0.0, weights, nextSubTrees);

  for(int levelCnt=0; levelCnt<nrTreeDimensions(); levelCnt++)
  {
    actSubTrees=nextSubTrees;
    sumLeafEntries=0;

    nextSubTrees = createSubTreeSetForLevel( levelCnt+1 );
    while( actSubTrees.hasNextEntry() )
    {
      entry = actSubTrees.nextEntry();
      node = entry.getTreeNode();
      sumLeafEntries = sumLeafEntries + node.getLeafEntryCounter();
      node.findBestSubTrees(p, entry.getDifference(), weights, nextSubTrees);

      if( nextSubTrees.hasEnoughEntries(sumLeafEntries, minNrProfiles ) )
        break;
    }
  }

  return actSubTrees;
}

...
Profile searchProfile = ...;
Vector  personalWeights = ...;
TreeSet nearestNeighbours = getNearestNeighbors(root, searchProfile,
                                                personalWeights, 5);
...
```

Listing 7.7: Weighted Search Nearest Neighbors

7.2.5. Ignoring Dimensions

The temporary ignoring of dimensions in a search space in the context of a specific request is a very important use case and was therefore defined as a requirement for the D^2-Tree in Section 7.2.1 on page 115. In the D^2-Tree this can be achieved by simply skipping the tree level being ignored and proceeding with the operation described above with the subtrees of the ignored nodes.

Method *get(TreeNode node, Profile p, Set result*) must be adapted being capable for handling set of subtrees. The necessary extensions are shown in Listing 7.8.

```
 1 get(TreeNode node, Profile p, Filter filter, Set result)
 2 {
 3   if( node.isLeafNode() )
 4     resultSet.add( node.getAllEntries() );
 5   else
 6   {
 7     TreeNode node nextNode = null;
 8
 9     if( filter.ignoreDimension( node.getDimensionId() ) )
10     {
11       TreeSet all = node.getAllSubTrees();
12       for(Iterator it=all.iterator(); it.hasNext(); )
13       {
14         nextNode = it.next();
15         get(nextNode, p, result);
16       }
17     }
18     else
19     {
20       valueId = p.getValueIdOf( node.getDimensionId() );
21       nextNode = node.getSubTree( valueId );
22       get(nextNode, p, result);
23     }
24   }
25 }
26
27 ...
28 Filter   personalFilter=...;
29 Profile  searchProfile = ...;
30 Set      result = new Set();
31 get(root, searchProfile, personalFilter, result);
32 ...
```

Listing 7.8: Get: Ignoring Dimensions

The filter object (see signature) contains the information concerning the dimensions being ignored, which can be requested by calling the method *filter.ignoreDimension(dimId)* for a given dimension. The necessary adaptations for dealing with sets of subtrees can be found from line 7 to line 17.

Incorporating filter aspects in the *getNearestNeighbors* method can be done with minimum effort, by simply adapting the exit condition of the loop in line 26 (see Listing 7.9).

```
1  TreeSet getNearestNeighbours(TreeNode root, Profil p,
2                              Filter filter, int minNrProfiles)
3  {
4    int        sumLeafEntries=0;
5    TreeSet    actSubTrees = null;
6    TreeSet    nextSubTrees = null;
7    SetEntry   entry = null;
8    TreeNode   node = null;
9
10   nextSubTrees = createSubTreeSetForLevel( 0 );
11   root.findBestSubTrees(p, 0.0, nextSubTrees);
12
13   for(int levelCnt=0; levelCnt<nrTreeDimensions(); levelCnt++)
14   {
15     actSubTrees=nextSubTrees;
16     sumLeafEntries=0;
17     nextSubTrees = createSubTreeSetForLevel( levelCnt+1 );
18     while( actSubTrees.hasNextEntry() )
19     {
20       entry = actSubTrees.nextEntry();
21       node = entry.getTreeNode();
22                         doIgnore=filter.ignoreDimension(node.getDimensionId());
23       sumLeafEntries = sumLeafEntries + node.getLeafEntryCounter();
24       node.findBestSubTrees(p, entry.getDifference(), nextSubTrees);
25
26       if( !doIgnore && nextSubTrees.hasEnoughEntries(sumLeafEntries, minNrProfiles))
27         break;
28     }
29   }
30
31   return actSubTrees;
32 }
33
34 ...
35 Profile searchProfile = ...;
36 TreeSet nearestNeighbours = getNearestNeighbours(root, searchProfile, 5);
37 ...
```

Listing 7.9: GetNearestNeighbors: Ignoring Dimensions

7.2.6. Cost Analysis

In this section a cost analysis of the standard D^2-Tree methods is presented on the basis of counting the number of required operations. The following characteristics (see Table 7.1) of the D^2-Tree are important for our analysis:

n	...	the number of inner node tree levels (tree height is $n + 1$)
m	...	the minimal number of profiles to be returned (if available)
$v_\oslash$	...	average number of node entries
k	...	average number of subtrees to be considered ($k \le v_\oslash$)

Table 7.1.: D^2-Tree Characteristics

The costs of the D^2-Tree methods are defined by $cost$ functions based on the operations listed in Table 7.2:

H	...	hash- or index function for finding a triple within a node
C	...	compare function
I	...	insert operation of a triple within a node
R	...	remove operation of a triple within a node
N	...	creation of a new node
D	...	deletion of a node
S	...	subtraction operation (applying difference function)
A	...	addition operation

Table 7.2.: Basic Operations

Costs of Method: insert

For each of the n inner node levels of the D^2-Tree method *insert* checks if an appropriate subtree (modeled as a triple in a node) exists. This checking is done by identifying an appropriate triple (hash access) and comparing it with corresponding attribute value. Assuming a D^2-Tree with n dimensions and that no new node have to be created, the costs of this insert operation are defined by n hash-operations and 1 insert operation of the profile in the leaf node – see Definition 7.23

$$cost(insert_{Std}) = nH + I \tag{7.23}$$

If at tree level l no triple is found new nodes must be created too. The adapted cost function can be found in Definition 7.24.

$$cost(insert_{New}) = cost(insert_{Std}) + (n + 1 - l)(I + N) \tag{7.24}$$

Costs of Method: remove

Method *remove* behaves similar to *insert* also reflected by the cost function – see Definition 7.25

$$cost(remove_{Std}) = nH + R \tag{7.25}$$

Removing the last entry of a node will also lead to the deletion of this node from the tree, possibly causing restructuring operations within the parent nodes. If, starting at level l, a path exists in the D^2-Tree having only 1 entry at each level, the removing of the leaf entry will also cause the deletion of this subtree/path. The adapted cost function can be found in Definition 7.26.

$$cost(remove_{Del}) = cost(remove_{Std}) + (n + 1 - l)(R + D) \tag{7.26}$$

Costs of Method: get

Concerning the traversing of the inner nodes method *get* induces equivalent costs as method *remove* or *insert*. If a specific profile, identified by an ID, should be returned an additional hash operation in the leaf node must be performed. The cost function is defined as:

$$cost(get_{Std}) = (n + 1)H \tag{7.27}$$

Costs of Method: getNearestNeighbors

The costs of the method *getNearestNeighbors* are defined by the operations needed to define the appropriate subtrees. Starting with the root node, appropriate subtrees are determined by (i) calculating the difference between the node triples and the corresponding attribute of the given search profile (costs: $v_{\oslash}S$) and (ii) finding the best k subtrees (costs: $C\sum_{i=1}^{k}(v_{\oslash} - i)$) which is (iii) controlled by incrementing a profile counter and comparing it with the number of of results to be returned (costs: $(k-1)A + kC)$) per subtree. The appropriate cost function for the root node (level 1) is shown in Definition 7.28.

$$\begin{aligned} cost(getNN_1) &= v_{\oslash}S + C\sum_{i=0}^{k-1}(v_{\oslash} - i) + ((k-1)A + kC) = \\ &= v_{\oslash}S + kC(v_{\oslash} - \frac{k-1}{2}) + ((k-1)A + kC) \end{aligned} \tag{7.28}$$

Having found the best k subtrees the same operations are now applied to all triples $v_\oslash$ of the root nodes of these trees ($kv_\oslash$). So, on level 2, the cost function is defined as:

$$cost(getNN_2) = kv_\oslash S + C\sum_{i=0}^{k-1}(kv_\oslash - i) + ((k-1)A + kC) =$$
$$= kv_\oslash S + kC(kv_\oslash - \frac{k-1}{2}) + ((k-1)A + kC) \quad (7.29)$$

So, for a D^2-Tree with n dimensions (levels), we can formulate an overall cost function:

$$cost(getNN_{Std}) = \sum_{j=0}^{n-1}(k^j v_\oslash S + C\sum_{i=0}^{k-1}(k^j v_\oslash - i) + ((k-1)A + kC)) \quad (7.30)$$

Obviously, the most important factors concerning costs are the number of triples stored in nodes and the number of subtrees required for defining the minimal number of profiles. Furthermore, if k and $v_\oslash$ have high values and if $k \approx v_\oslash$ the sorting term $\sum_{i=0}^{k-1}(v_\oslash - i)$ will have an order of $O(v_\oslash^2)$. In that cases sorting structures such as red-black-trees (Samet, 2005) can be used for further optimizations.

Discussion

Assuming, for ease of discussion, that the costs of all operations are equivalent to C, the cost functions of the tree methods can be simplified as shown in Table 7.3.

Method	**Cost function**	**Order**
$insert_{Std}$	$C(n+1)$	$O(n)$
$remove_{Std}$	$C(n+1)$	$O(n)$
get_{Std}	$C(n+1)$	$O(n)$
$getNN_{Std}$	$C\sum_{j=0}^{n-1}(k^j v_\oslash + \sum_{i=0}^{k-1}(k^j v_\oslash - i) + 2k-1)$	O$(nv_\oslash)\, to\, O(v_\oslash^{n+1})$

Table 7.3.: Cost Functions

As shown in Table 7.3 the standard versions of the methods *insert, remove* and *get* have a linear behavior only depending on the depth n of the D^2-Tree.

In contrast, method *getNN* has a more complex cost function depending on the depth n, the number of triples per node (defined as $v_\oslash$) and the number k of subtrees being considered. For defining the order of costs we further simplify the cost function as shown Table 7.3 on page 133. After resolving the function above, we get:

$$\begin{aligned} cost(getNN_{Std}) &= C\sum_{j=0}^{n-1}(k^j v_\oslash + \sum_{i=0}^{k-1}(k^j v_\oslash - i) + 2k - 1)) = \\ &= C\sum_{j=0}^{n-1}(k^j v_\oslash + k^{j+1} v_\oslash - \frac{k(k-1)}{2} + 2k - 1) \end{aligned} \tag{7.31}$$

Considering the constraints, that $n >> 1$ and $v_\oslash \geq k$ the most relevant term of the expression $(k^j v_\oslash + k^{j+1} v_\oslash - \frac{k(k-1)}{2} + 2k - 1)$ is $k^{j+1} v_\oslash$ which will be used user for approximating the overall costs. Thus, ignoring all other terms, we get

$$cost(getNN_{Std}) = C\sum_{j=0}^{n-1}(k^{j+1} v_\oslash) = C(v_\oslash \frac{1-k^{n+1}}{1-k} - 1) \quad ; k \neq 1 \tag{7.32}$$

for $k \neq 1$ and

$$cost(getNN_{Std}) = C\sum_{j=0}^{n-1}(k^{j+1} v_\oslash) = C v_\oslash n \quad ; k = 1 \tag{7.33}$$

in the case that $k = 1$. While Definition 7.33 can directly be used to define the lower threshold of our cost approximation, an upper threshold can be found by assuming that $k = v_\oslash$. Thus, the formula presented in Definition 7.32 can further be simplified:

$$\begin{aligned} cost(getNN_{Std}) = C(v_\oslash \frac{1-v_\oslash^{n+1}}{1-v_\oslash} - 1) = C(\frac{v_\oslash - v_\oslash^{n+2}}{1-v_\oslash} - 1) \approx \\ \approx C\frac{v_\oslash^{n+2}}{v_\oslash} = C v_\oslash^{n+1} \end{aligned} \tag{7.34}$$

In contrast to Definition 7.33, where the costs have a pretty good linear characteristics concerning n and $v_\oslash$, the upper bound costs of Definition 7.34 rise to the power of n!

Having such a huge range concerning the theoretical costs of this method, it is important to understand the role of the involved parameters and how their characteristics will be (more likely) in real time applications.

As shown in Definition 7.30 on page 133 the most critical parameters concerning the performance of *getNearestNeighbors* are the number of nodes being considered ($v_\oslash$) and the minimal number of requested profiles m which is used to define the necessary k subtrees of the next level. In order to produce high quality recommendations good personalization systems try to produce a limited, manageable number of results instead of long lists of items. In practice m is very small, typically within the order of 10-th, in contrast to the number of items which often ranges from some thousands (small shops) to even millions (e.g. on-line music stores). Thus, also the number of subtrees k, necessary to produce the requested result, will be very small in practice, especially concerning the higher levels (near root) of the D^2-Tree. Furthermore, also the value ranges of discretized dimensions are typically covering moderate numbers of values.

Example: Given a D^2-Tree with 10 dimensions ($n = 10$) each having a value range of 5. Moreover we assume that the tree is completely filled, so that one profile exists for each possible value combination ($nr = 5^{10}$), resulting in $v_\oslash = 5$. Based on that structure the $m = 20$ nearest neighbors of a given search profile s should be defined. Starting at the root node, each of the 5 available subtrees is containing $v_\oslash^{n-1} = 5^9$ profiles and so only one subtree must be considered ($k = 1$). The process continues, using only one subtree, until the penultimate level ($level = 9$) is reached where 4 subtrees ($k = 4$) are necessary for generating 20 results. So, on the last inner node level of the D^2-Tree ($level = 10$) 4 nodes, having together 20 triples, must be considered for calculation.

Using these figures together with a cost function $cost(v_\oslash, k)$[6] for given values $v_\oslash$ and k – see Definition 7.35

$$cost(v_\oslash, k) = C(v_\oslash + k(v_\oslash - \frac{k-1}{2}) + 2k - 1) \tag{7.35}$$

we can calculate the real effort

$$\begin{aligned} cost(getNN_{Sample}) &= C(8 \cdot cost(5,1) + cost(5,4) + cost(4 \cdot 5, 20)) = \\ &= C(8 \cdot 11 + 26 + 269) = 383C \end{aligned} \tag{7.36}$$

where the first term ($8 \cdot cost(5,1)$) describes the costs at the first 8 levels, the second term the costs at the a single node on the $9th$ level and the last term the costs for calculating 4 nodes of the last inner node level. So, as a result, only ≈ 400 operations have to be performed for finding 20 profiles

[6]Derived from Definition 7.28 on page 132 assuming that all operations have the same costs as C.

out of 5^{10} which is far more close to $nv_{\oslash}$ (50) than to $v_{\oslash}^{n+1}$ (5^{11}).

But the reduction to only one subtree at a given level bares some risks, especially when (i) the number of profiles covered by the subtree is nearly equal to the requested number and (ii) all levels of the D^2-Tree have nearly the same weights. In that case a subtree being ignored at level i, can produce the best match at level $i + 1$. A proper strategy to handle such situations is the extension of the minimal set of necessary subtrees (T) by a small number of next best alternatives.

7.3. Contributions to the Research Question

In this chapter we have shown, how adding contextual information can improve recommendation algorithms without introducing new data (e.g., time) which standard approaches do not consider.
The *pretty good recommendation* approach, presented in the first section, incorporated information concerning the importance of an item in local (per user) and global (for all users) contexts as well as in the context of the pairwise relations of preference sets. The evaluation showed, that the presented similarity correlation outperforms standard algorithms, especially in a context with little information. Furthermore, it could be proofed, that the incorporation of local and global item importance (concepts similar to the $TFxIDF$ measure used in *text retrieval*) together with the overlap degree of the preference sets – computed using the confidence of association rule mining – can improve the similarity relation of users.

The D^2-Tree, presented in this chapter, is a new data structure for solving the *k-nearest neighbors* problem in a search space, spanned by a high number of discretized dimensions. Although well-known concepts were incorporated in this structure, significant differences exist. In contrast to classic search trees the structure of the D^2-Tree does not reflect the adjacency of profiles but supports algorithms for defining this closeness dynamically. The most important characteristics of the D^2-Tree are:

- A D^2-Tree has a constant height and a well-known maximum width, allowing better performance/memory consumption estimations which is especially important for time critical applications.
- Compared with other trees (e.g. B^*-, B^+-trees) much less changes of the tree structure will occur, operations such as merging or splitting of nodes do not exist. Therefore, costly locks of tree nodes can be reduced to a minimum, an important issue in the context of parallel systems.

- Due to its relatively constant structure the D^2-Tree is particularly suitable for persistent applications, especially important in the context of huge data sets.
- The separation of the tree structure and the closeness of the profiles, obtained by the incorporation of the difference functions, supports a weight-based approach for solving the kNN-problem.
- By using explicitly defined difference functions, domain knowledge can easily be imported and adapted without effecting the search structure.
- The performance of the tree mainly depends on the number of dimensions and their respective value ranges, and not on the number of profiles.

The D^2-Tree, presented in the second section, is a complement for the *Adaptive Profile Model* providing an efficient solution for the k-nearest neighbor problem. The most important advantages of the D^2-Tree are the support of weighted requests and the ability to incorporate domain knowledge even dynamically, thus supporting the context of use concerning a specific request.

8. Evaluation and Examination

The analysis presented in this chapter should provide the basis for the verification of major concepts of the Adaptive Personalization approach as well as a deeper insight into a real world personalization system and the constraints it has to deal with.

The data on which this examination is based on was provided by two instances of the *Ericsson's Media Suite - Music (EMM)*, a content download platform incorporating an implementation of the personalization approach presented in this thesis. Because of the specifics of the music domain, the limitations of this instant implementation (mostly triggered by business constraints) and the shortcomings concerning system operations (e.g. system problems, down-times, etc.) this examination can only be seen as a qualitative analysis of the Adaptive Personalization concepts.

This chapter is organized as follows. In the first Section 8.1 a short introduction to the EMM, explaining important aspects of its implementation, is provided. Next, in Section 8.2 on page 146 the objectives and the basic conditions of this examination are discussed providing the basis for our analysis, presented in Section 8.3 on page 148. The results and implications are discussed in Section 8.4 on page 165.

8.1. Ericsson's Media Suite - Music

The *Ericsson's Media Suite - Music (EMM)* is a content download platform focusing on the delivery of music related content on the mobile channel (a web interface was supported too) and went on-line in Europe, Asia, and the USA. The personalization aspects of the *Ericsson's Media Suite - Music* were implemented on the basis of API calls – a proprietary XML over HTTP protocol – to a self-contained recommender system, called *Recommender Engine (RE)*, being an implementation of the Adaptive Personalization approach. Both applications, the EMM as well as the RE, were implemented in *Java* and stored their data in a SQL database - see Figure 8.1 on page 140.

Due to this separation of these two applications a data synchronization process was implemented keeping the data in the RE database up-to-date. Basically, changes concerning first class objects such as user and item

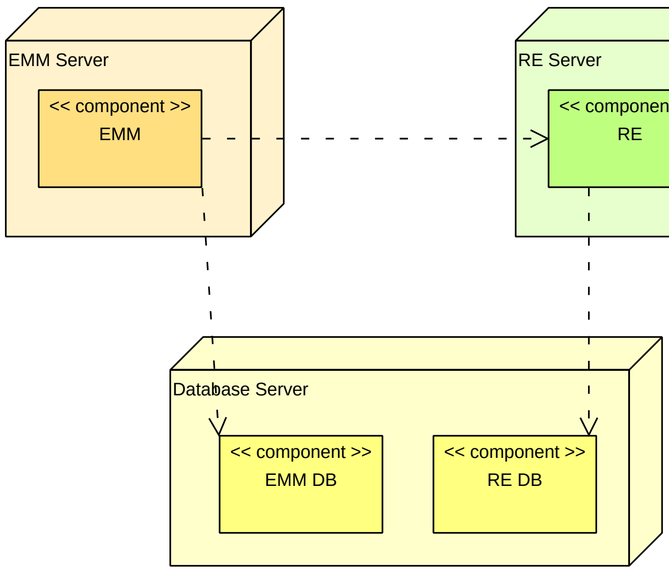


Figure 8.1.: Deplyoment of Components

records in the EMM triggered an update processes in the RE immediately. For importing huge sets of content items a special upload service, called *bulk upload*, was provided. In the case of unavailability of the RE (e.g. RE was not started in advance, RE stopped, etc.) no update or synchronization process was performed or scheduled – resulting in divergent datasets in EMM and RE. Furthermore, both systems were implemented meeting multi-tenant requirements.

8.1.1. Data Model

In Figure 8.2 on page 141 an overview of the conceptual data model of the first class objects of the EMM, being of interest for the personalization, is given showing the inter-class associations.

The class **Artist** was used for modeling song writers/performers and consisted of attributes such as name, identifier, a list of associated clusters, etc. This artist information was used to present a given artist of a track and/or album to the community. Furthermore, no distinction was made between a group such as *The Rolling Stones* and actors e.g., *Madonna*. The **Track** class was used to represent single songs or tracks on a conceptual basis by using attributes such as name of the track, release date, list of associated clusters, etc. Different audio instances of a given track (e.g. an MP3 audio file) were modeled as **Assets** associated with that track – see Figure 8.2 on page 141.

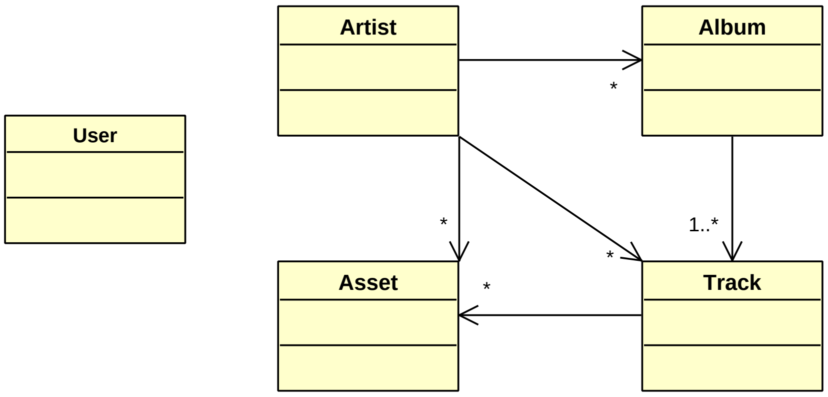


Figure 8.2.: EMM Item Object Model

Although no compilations of tracks could be downloaded by users, the **Album** concept was introduced mainly based on marketing considerations, containing attributes such as name of the album, release date, list of associated clusters, etc. In contrast to artists and tracks, which were presented as first class objects on dedicated areas of the portal – called *Artist Page* and *Track Page* respectively – the album information was only used in context with artists and tracks to provide e.g. a collection of albums of an artist or as a hint where a given track can be found.

All download able products provided by the EMM were modeled as **Assets** of different types such as full audio files, wall papers, ring tones, games, etc. which could be associated with a track (e.g. full audio) or artist (e.g. wall paper). An asset consisted of an asset type, name of the asset, sales information such as first/last date for valid downloads, list of associated clusters, etc.

Furthermore, all these classes included operational data such as tenant identification, permissions, date of upload into the system, data for representation (e.g. pictures like covers) etc. too.

All relevant information concerning the user was collected in **User** objects, consisting of attributes such as socio demographic information such as age and gender (both optional), personal preferences concerning music genres, favored/unfavored artists and tracks, just bought assets, compilations of songs[1], the MSISDN[2], the type of the current handset, etc.

[1]A user based compilation of tracks was called *playlist*, although it could not be played.

[2]MSISDN is the (most common used) abbreviation of *Mobile Subscriber Integrated Services Digital Network Number*

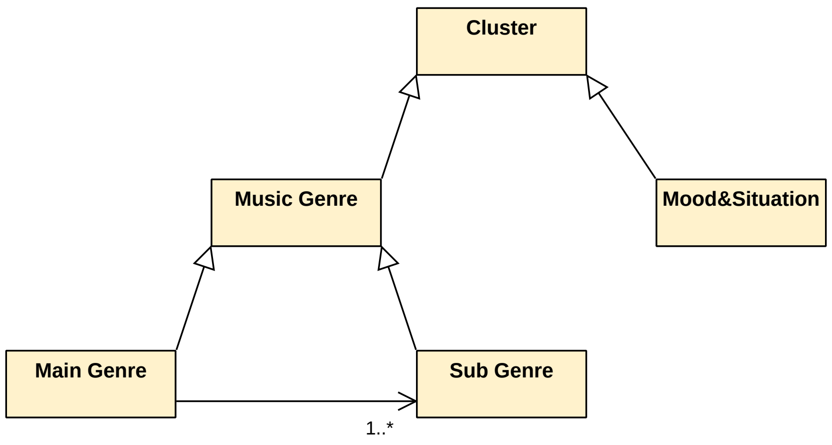


Figure 8.3.: EMM Cluster

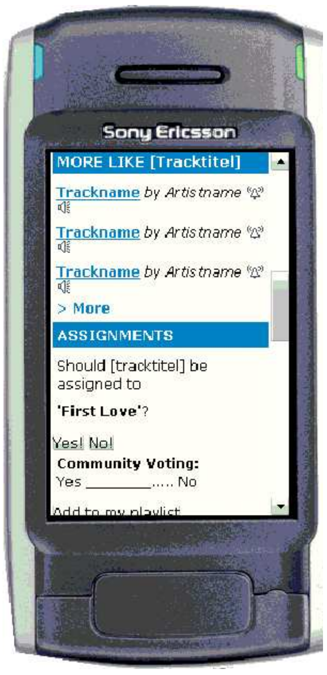
The item space was structured according to Figure 8.3. The traditional *Music Genres* were modeled as a two layer structure consisting of main genres (e.g. *rock*) and sub genres (e.g. *hard rock*) which could be managed by the portal administrators allowing them to add and delete specific genres. The *Mood&Situation* clusters, which can be seen as alternative music genres, were introduced to provide a need based classification of music related content. Samples of such clusters were

- Dance Party
- First Love
- Action&Sports
- Under Pressure
- In the Blues

which could be administrated by portal operators too. In contrast to the music genres, where the classification process was solely performed by administrators, the capability of assigning an item to a *Mood&Situation* cluster was provided to the community too. The community based classification of artists or tracks was called *tuning*. In Figure 8.4 on page 143 this tuning use case is shown for the mobile channel, where the user is asked if a given track, should be assigned to *First Love* cluster.

In contrast to the Web interface, where all possible options were presented to the user as a multiple choice list, in the the WAP[3] interface only

[3]Wireless Application Protocol


MORE LIKE [Tracktitel]

Trackname by Artistname

Trackname by Artistname

Trackname by Artistname

> More

ASSIGNMENTS

Should [tracktitel] be assigned to

'First Love'?

Yes! No!

Community Voting:

Yes ________..... No

Add to my playlist

Figure 8.4.: Mobile Tuning of a Track

one option, being defined by some selection procedure[4], was provided to the user.

8.1.2. Personalization Aspects

In the context of the recommender engine a user was defined as *registered* if he/she passed through an optional registration process provided by the EMM, where age, gender and genre preferences were requested and stored in the self-assessment view of the user's profile. Beside this optional registration process the following information was used for implementing profile refinement, as described in Chapter 6.

- *purchase of assets*: A purchase action occurred, when a user downloaded a product (e.g. a wall paper). The purchase action got the highest action weight ($w_{purchase} = 0.38$).

- *rating of artists and tracks*: A rating action occurred, when a user posted an explicit rating describing his liking or disliking of an item (artist or track) ($w_{liking} = 0.21$, $w_{disliking} = -0.21$).

[4]In the current version a random process was taken

- *viewing of artists, tracks and assets*: A viewing action occurred, when a user visited the presentation area of an item ($w_{view} = 0.09$).
- *pre-viewing of assets*: A pre-viewing action occurred, when a user pre-viewed an asset (e.g. listening of an audio file) ($w_{pre-view} = 0.09$).
- *add track to playlist*: Adding a track to a playlist, a personal compilation of tracks, was seen as an important feedback comparable to a positive rating ($w_{add-playlist} = 0.21$).

Based on this information and the meta data of the content, the following recommendation use cases were implemented:

- Hot Recommendation
- Songs, similar users like
- Assets, other users also bought
- Other artists you might like
- Other tracks you might like

The *Hot Recommendation*, available for registered users only, provided songs based on the user's genre preferences and the up-to-dateness of the tracks. Songs, presented recently, were lower prioritized and disliked tracks as well as tracks of unfavored artists were ignored. Furthermore, tracks with no available products/assets for the user's handset were ignored. The Hot Recommendation itself was composed up to 5 (sub) rules each contributing some items to the overall resulting list. The configuration of these rules concerning the result list was done on the basis of a context free grammar – see Figure B.6 on page 186 - on an administration level. These rules were:

1. Take most recent tracks of favored artists out of the self-assessment profile
2. Take most recent tracks of favored artists out of the observation profile
3. Get high rated tracks of similar users
4. Most recent tracks of favored genres out of the self-assessment profile
5. Most recent tracks of favored genres out of the observation profile

As the standard setting a combination of rule 4 and rule 5 was chosen. A sample, how this recommendation was presented to the user, is shown in Figure B.1 on page 181 in Appendix B.

The *Songs similar people like* recommendation was available for registered users only. High rated tracks – implicitly (e.g. view the product page) and explicitly – of similar users were used to calculate this recommendation. The similarity relation between users was defined by: Age, gender, genre preferences and country. Disliked tracks or tracks with no products for the user's handset were ignored. A sample, how this recommendation was presented to the user, is shown in Figure B.2 on page 182 in Appendix B.

The *Assets, other users also bought* recommendation referred to the shopping history of users and was not limited to registered users. Based on a given product a list containing items, bought by users having also bought the given one, was generated. Furthermore, disliked products or products not available for that user's handset were ignored. A sample, how this recommendation was presented to the user, is shown in Figure B.4 on page 184.

The *Other artists you might like* recommendation was based on artist relations and was available for registered and unregistered users. This recommendation was composed up to two (sub) rules and could be configured on an administration level. These rules were:

1. Take predefined artist-to-artist relations
2. Take high rated artists of similar users

Rule 1 referred to the case of the availability of 3^{rd} party meta data concerning artist relations (e.g. data from AllMusicGuide[5]). Rule 2 generated a list of artists based on ratings of similar users, where user similarity was based on: Gender, age, and genre preferences. Furthermore, filters concerning unfavored artists, permissions and handling of explicit information were applied. The standard behavior was implemented by Rule 2!

The *Other tracks you might like* recommendation was based on different track relations – given a specific track – and was provided for registered and unregistered users. This recommendation was composed up to four (sub) rules and could be configured on an administration level. These rules were:

[5]`http://www.allmusic.com` as of 08/2008

1. Take tracks that *sound similar* (was not provided for the version under examination)
2. Take new tracks of the same artist
3. Take tracks based on shopping cart, starting with the given one
4. Take new tracks out of same genre

Furthermore, filters concerning unfavored tracks, tracks of unfavored artists, permissions, and handling of explicit information were applied. The standard behavior was implemented by rule 2 and rule 4. A sample, how this recommendation was presented to the user, is shown in Figure B.3 on page 183 in Appendix B.

8.2. Objectives and General Conditions

To check/verify the acceptance and potential of the personalization approach, answers to the following questions, closely related to the research question, have to be found:

1. How efficient is the multi-view profile approach – what is the impact/benefit?
2. Which features/concepts are accepted/rejected/ignored by the users? (e.g. tunings of items, rating of items, playlists, etc.)
3. How far do the current recommendation use cases meet the users requirements?
4. Which conclusions can be derived from the analysis?

Furthermore, these analysis should be applied to different instances of the EMM.

8.2.1. Definitions

Throughout this chapter, the following shortcuts are used:

- EU: European installation, refers to the origin of the data
- DB: Database
- EMM: The Ericsson's Media Suite - Music content download platform, refers to the data stored in the EMM database only

- MY: Malaysian installation, refers to the origin of the data
- RE: Recommender Engine, refers to the data stored in the recommender database only
- #: ’Number of’

8.2.2. Description of Data and Applied Procedures

Two data sources were taken for this analysis:

- A copy of the databases (EMM-database, RE-database) from the European server – taken at the 05/31/2006. The covered period to be analyzed ranges from 9/2005 to 5/2006 – about 9 month. Multiple tenants were hosted on the EU installation.
- A copy of the databases (EMM-database, RE-database) from the Malaysian server – taken at the 05/16/2006. The period to be analyzed ranges from 10/2005 to mid 5/2006 – about 7.5 month. The MY installation was only used by one tenant.

These two databases where installed locally on a MySql[6]-database to perform the analysis. Beside some necessary repair tables no extra manipulation (e.g. check if associations/entries are consistent) of the data was performed. Beside these databases, no further information source was taken. Although several operators were hosted on the EU installation the Europe data was taken as a whole and not further divided into operator specific data sets. Therefore, all conclusions/statements concerning operator specific adaptation/concepts in the EU context must be seen in that light. For most analysis the data stored in recommender database was taken.

8.2.3. Limitations and Short-Comings

Due to some handling errors during the copying process of the MY databases log-entries of EMM got lost (e.g., behavioral data of users concerning EMM features) and therefore no log specific analysis concerning EMM were performed. Furthermore, the MY data is somewhat incomplete because the recommender engine was sometimes stopped at peak-time due to insufficient hardware infrastructure, resulting in a loss of some user feedback. So, only a very small data set, concerning the call frequency of recommendation strategies, was available for the MY installation[7]

[6]`http://www.mysql.com` as of 11/2008
[7]log entries of 06/16/2006, ranging from 1:00pm to 11:00pm

8.3. Examination of On-line Systems

8.3.1. Users and Items

Concerning EU, we found 873.357 registered users in the RE but 1.017.495 in the corresponding EMM – a difference of 16% (base: recommender database)!. In MY we found 755.744 registered users in the RE, but 1.327.123 in the corresponding EMM DB - a difference of 75%!

Furthermore, 59% of all registered users in EU (basis RE) and 62% in MY were *active users*. A user is classified as active, if he/she performed at least one action e.g., view, buy, rate, etc. For further discussions, only the active users were taken into account.

Concerning content, the differences between the synchronized databases EMM and RE were not that dramatical in EU as well as in MY. In EU this difference was about 1% while in MY it ranged between 1% (tracks) and 4%(artists). Detailed data can be found in Table 8.1.

# of	**EU**	**MY**
Registered User RE	873.357	755.744
Registered User EMM	1.017.495	1.327.123
active Users	517.264	474.513
(%RE ; %EMM)	(59% ; 50%)	(62% ; 36%)
Artists RE	9.193	3.846
Artists EMM	9.295	4.005
Tracks RE	152.588	36.348
Tracks EMM	152.897	36.844
Assets RE	171.147	45.817
Assets EMM	172.282	48.093

Table 8.1.: Basic Data

In contrast to the content situation – where the synchronization between RE and EMM is not a critical problem – we have enormous differences concerning users in EU (16%) and MY (75%). The MY situation could be explained by the fact, that the RE was sometimes stopped during peak time – but this explanation cannot be applied to the EU installation. Most likely, these differences are symptoms of an inadequate or missing synchronization prozess concerning user registration which has to be considered in further versions.

8.3.2. Meta Data Quality: Item-to-Genre Affiliation

The recommender engine under examination uses genre information of items such as artists, tracks, etc. for basic recommendations and learning behavior (e.g. observed preferences). As mentioned above, the association of items to specific music genres is done by the EMM content administrator mainly based on the meta data provided by content providers (e.g. Sony BMG).

In the following tables and figures, the distribution of tracks and artists concerning music genres is shown for the EU and MY content.

# of	**EU**	**MY**
available genres (subgenres)	67	40
tracks, associated with at least one genre (% all tracks)	142.746 (93%)	36.021 (99%)
tracks without genre association (%all tracks)	9.842 (7%)	327 (1%)
all track-genre associations	267.269	61.031
avg. associations per track	1,8	1,7
artists, associated with at least one genre (% all artists)	9.193 (93%)	3.650 (94%)
artists without genre association (% all artists)	641 (7%)	196 (6%)
all artist-genre associations	16.364	6400
avg. associations per artist	1,8	1,7

Table 8.2.: Genre Affiliation

In both installations (EU, MY) we have a very high number of associations, meaning that most tracks and artists ($> 93\%$) are associated with at least one cluster!

But examining the distribution of items along the given genres shows a dissatisfying situation. Concerning tracks, the genre distribution is shown in Figure 8.5 on page 150 for EU and in Figure 8.6 on page 150 for MY. For reasons of clarity the figures do not contain any legend information or numbers but they provide an impression concerning the distribution of tracks along basic genres (a more detailed information can be found in Appendix C on page 187).

The distribution of tracks concerning genres is a disaster for EU and by far not optimal in MY. Concerning EU, 65% of all tracks were associated with *Classic Pop*, 7% with *Classic Rock*, 5% with *Jazz* and 3% with *Other Pop* resulting that four genres are covering about 80% of all track content!

In contrast, the content situation is slightly better in MY, because 80% of the track content is covered by eight different genres. Furthermore, no single genre covers more than 20% of the tracks! In MY the eight most prominent genres and their coverage are: *Other* with 20%, *Pop Danceclub*

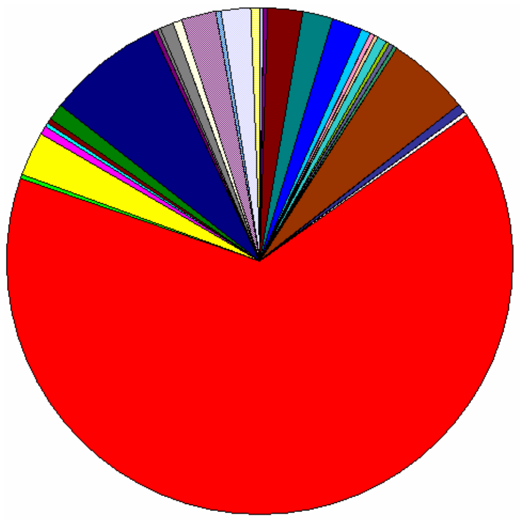


Figure 8.5.: Track – Genre Distribution in EU

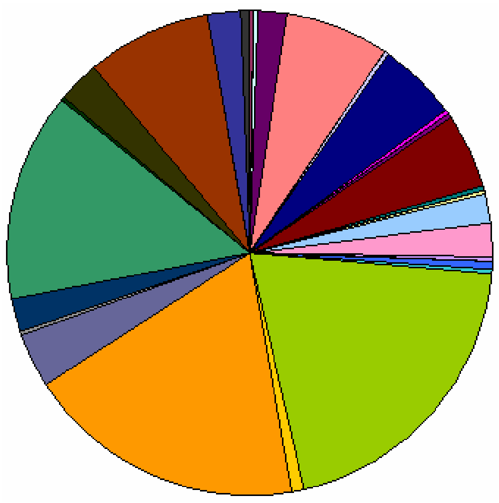


Figure 8.6.: Track – Genre Distribution in MY

with 18%, *Other Pop* with 14%, *Classic Rock* with 8%, *Chinese Pop* with 7%, *Classical* with 6%, *Jazz* having 5%, and *Hip Hop* 4%.

Considering the high number of associated tracks ($> 93\%$), the low number of average associations (≈ 1.8) and the fact, that some content providers do/can not deliver genre information for portions of their content (up to 20%!) it seems that genres such as *Classic Pop* or *Other* are used as default associations. As a result, the genre information can only be used very carefully!

8.3.3. User Profiles

For each active user three profile views were stored, consisting of the self-assessment view (SELF), the observation view (OBSRV) and a combination of these two views (COMB). The community view, as proposed in the Adaptive Personalization concept, was not implemented due to the lack of a user-user based feedback functionality on the portal.

The self-assessment view was constructed during the optional registration process for obtaining access to the *Hot Recommendations* and *Songs similar people like* functionality placed on a specific area of the portal (called *My Mobile Music area* see Figure B.1 on page 181 and Figure B.2 on page 182). Within this registration process the user declared his/her age, gender, and music genre preferences. As a consequence, the *Hot Recommendation* and *Songs similar people like* recommendations were only provided to users, having a SELF profile! The observation view was constructed by observing the behavior of the user concerning actions like rating or buying an item, visiting of item pages (e.g. artist page), etc. as described in Chapter 6.

The combined view was constructed as a combination of the SELF and OBSRV view, calculated as defined in Definition 8.1 on page 151

$$COMB = weight_{self}SELF + weight_{obsrv}OBSRV \tag{8.1}$$

with a higher prioritization of the SELF view ($weight_{self} = 0.6$ and $weight_{obsrv} = 0.4$). The COMB profile was used for calculating recommendations and was introduced due to performance considerations only. Furthermore, the COMB profile was only created/stored when it was *effective* – when at least one preference value was above a given threshold.

In EU only 1.7% (9.172) of the active users had a SELF profile, 82% (424.804) had a OBSRV profile and 12% (63.074) had a COMB profile. In contrast, in MY 15% of all active users had a SELF profile, 85% had an

OBSRV profile and 32% had a COMB profile.

By analyzing the music genre preferences of the users by counting the number of genre associations we found:

- The average number of associations of a SELF profile is eight in EU and MY
- The average number of associations of a OBSRV profile is five in EU and eight in MY
- The average number of associations of the OBSRV profile, being not covered by the SELF profile, is three in EU and six in MY
- The average number of associations of a COMB profile is nine in EU and eight in MY
- At least 29% of the users having a SELF profile have a larger COMB profile in EU and 42% in MY.

Surprisingly there is an enormous difference between the EU and MY users, concerning the willingness/ability to perform a registration process 1.7% to 15% – about a factor of nine! Based on the current settings of the RE (being equal for MY and EU) 12% of the active users in EU but 32% in MY are benefiting from personalization in some way (e.g. by applying filters).

Based on the data presented above the conclusion can be drawn, that the multi-view concept is successful/useful, because there are significant differences between the SELF and the COMB profile. So, on the basis of the preferences of the SELF profile, the OBSERV profile accounts for 37% of additional/other preferences in EU and about 75% in MY. Furthermore, 29% of the users in EU and 42% in MY get more/other recommendations due to the OBSRV profile than they would get only based on the SELF profile.

8.3.4. User Actions

In this section user activities, such as purchases, viewing artists or track pages, etc. are analyzed.

Purchases

From the basic purchase statistics provided in Table 8.3, the following conclusions can be drawn:

1. Not surprisingly, registered users are the better customers. In EU 0.6% of the active customers were responsible for 6% of all purchases! In MY 6% of the active users were responsible for 29% of all purchases

2. The portion of real customers were quite similar in EU (27%) and in MY (31%)

3. Only a limited number – 9% in EU and 31% in MY – of all assets were bought

# of	**EU**	**MY**	**Comment**
all purchases	302.658	401.724	
distinct purchases (% of all purchases)	288.526 (95%)	330.196 (82%)	Ignoring multi purchases of the same item of a user!
purchasing users (% of active users)	141.502 (27%)	147.868 (31%)	
distinct purchased assets (% of all assets)	15.537 (9%)	10.364 (31%)	
purchases of registered users (% of all purchases)	18.579 (6%)	116.213 (29%)	User, having a SELF profile
registered users who purchased (% of active users)	2.854 (0,6%)	29.758 (6%)	

Table 8.3.: Basic Purchase Statistic

One important business ratio of download portals is the *purchase frequency* which defines how many users are performing how many purchases. This *purchase frequency* is shown in Figure 8.7 on page 154 and Figure 8.8 on page 154[8] – more detailed information can be found in Appendix C on page 187.

Although theses purchase statistics – especially for MY – are not far away from well-known content download platforms such as JAMBA[9] it should be stressed, that the majority of the users are buying only once or twice (85% in EU and 78% in MY) – a typical behaviour of occasional customers!

[8]The 0% markers were caused by rounding
[9]`http://www.jamba.de` as of 12/2008

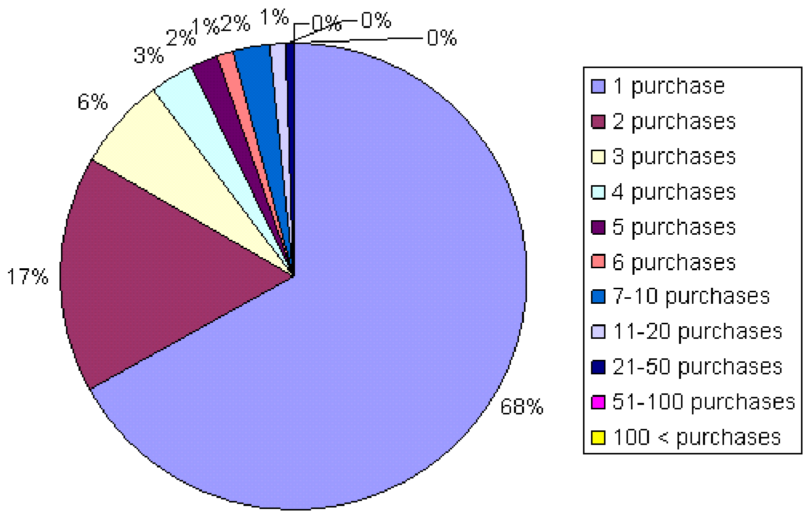


Figure 8.7.: Purchase Frequency in EU

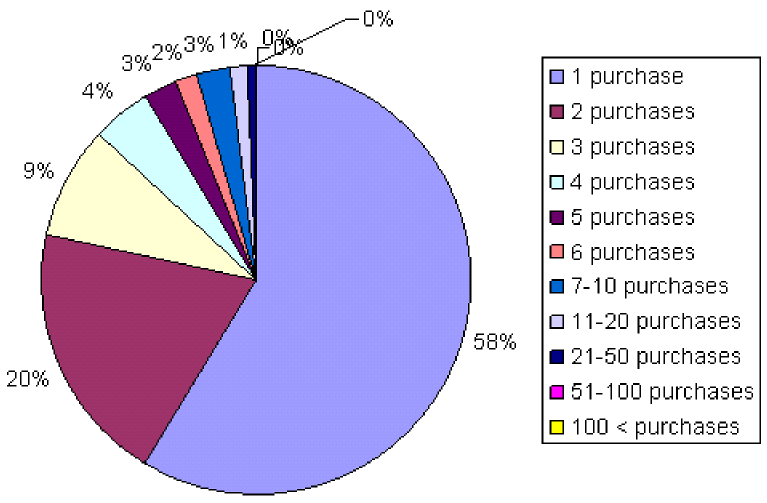


Figure 8.8.: Purchase Frequency in MY

Item Tuning

An *item tuning* occurred when a user performs a classification of an item – provided for artists (Table 8.4 on page 155) and tracks (Table 8.5 on page 156)– to one or more *Mood & Situation* Clusters.

# of	**EU**	**MY**	**Comment**
tuned artists (% of all artists)	945 (10,3%)	1.660 (43,2%)	# of artists, which were visible as tuned elements on the portal.
artists tuned by the administrator (% of all artists)	84 (1%)	63 (2%)	Each administrator tuning had an effect
artists tuned by the community (% of all artists)	895 (9,7%)	1.642 (42,7%)	Not every tuning had an effect due to thresholds
Web artist tunings (% all tunings)	260 (8%)	n.a.	Tuned via the WEB interface
WAP artist tunings (%all tunings)	3.181 (92%)	n.a.	Tuned via the WAP interface

Table 8.4.: Tuning of Artists

Comparing the two installations EU and MY is not reasonable, because not all operators in EU offered item tuning to their subscribers. Concerning the MY installation the item tuning feature seems to be widely accepted by the users, because 43% of all artists and 15% of all tracks were tuned and the lion-share of this classification was made by the community – see Figure B.5 on page 185! Another interesting aspect is that this service seems to be accepted also on mobile devives!

Definition of Liked/Disliked Artists

The user was able to add an artist to his/her list of *preferred* or *disliked* artists. These artists were stored in a specific list manageable by the users. Table 8.6 shows the acceptance of this feature by the community for EU and MY.

In contrast to EU where only 3% of the active users defined a favourite artist in MY this feature was well accepted by the community, because 14% of all active users declared at least one artist as favourite.

# of	**EU**	**MY**	**Comment**
tuned tracks (% of all tracks)	1.977 (1,3%)	5.710 (16%)	# of tracks, which are visible as 'tuned' elements on the portal
tracks, tuned by the admin (% of all tracks)	856 (0,6%)	1.036 (3%)	Each admin tuning has an effect
tracks tuned by the community (% of all tracks)	1.239 (0,8%)	5.315 (15%)	Not every 'tuning' has an effect due to thresholds
Web track tunings (% all tunings)	243 (5%)	n.a	No data available for MY
WAP track tunings (%all tunings)	4.788 (95%)	n.a	No data available for MY

Table 8.5.: Tuning of Tracks

# of	**EU**	**MY**
users having favored artist list (in SELF) (% of act users)	15.593 (3%)	65.647 (14%)
distinct favored artists (in SELF) (% of all artists)	1.608 (17%)	2.027 (53%)
average size of favored artist list	1,3	1,9
users having unfavored artist lists (% of active users)	1.119 (0,2%)	6.082 (1%)
distinct unfavored artists (% of all artists)	428 (4%)	1.160 (30%)
average size of unfavored artist lists	1,1	1,3

Table 8.6.: Liked/Disliked Artist

Definition of Liked/Disliked Tracks

By rating a track *good (“I like this track”)* or *bad (“I don’t like this track”*) – see Figure B.5 on page 185 this track was added to an appropriate like/dislike list in the user’s profile implicitly. These lists were used as filter criteria only, e.g. by suppressing the appearance of a disliked track in personalized recommendation lists.

# of	**EU**	**MY**
users having favored track list (in SELF) (% of act users)	14.684 (3%)	67.982 (14%)
distinct favored tracks (in SELF) (% of all tracks)	2.406 (2%)	6.101 (17%)
average size of favored track list	1,2	1,9
users having unfavored track list (% of active users)	2.398 (0,5%)	8.432 (2%)
distinct unfavored tracks (% of all tracks)	1.795 (1%)	3.432 (9%)
average size of unfavored track list	2	1,5

Table 8.7.: Liked/Disliked Tracks

These results are very similar to those found in the artist context. While in EU only 3% of the active users defined a favourite track in MY this feature was well accepted by the community because 14% of all active users rated at least one track as *good*! This significant difference between EU and MY could probably be explained by the different positioning of the two installations. While in MY the EMM was launched as a music platform also offering music relevant content, in EU (e.g., in Poland) the EMM was more positioned as a content download platform, competing with platforms as JAMBA[10]. Furthermore, users posted more positive than negative track ratings, especially in MY.

Playlists

A playlist is a named collection of tracks, created by a registered user. The name playlist is somewhat misleading because such a collection could not be replayed! It’s use was limited to present personalized music collections to the community.

In contrast to the EU installation, where this feature was not very popular, it was accepted by the MY community very well. Once more, this difference could be explained by the different usage scenarios. In MY, the platform

[10]`http://www.jamba.de` as of 11/2008

# of	**EU**	**MY**
playlists	7.585	213.934
users having a playlist (% of all active users)	6.307 (1%)	57.988 (12%)
all playlist entries (tracks)	10.151	259.372
of distinct entries (% of all tracks)	1.648 (1%)	6.116 (13%)
tracks of an average playlist	1,3	4,5

Table 8.8.: Playlists: Track Compilations

was launched with a focus on music which engaged users to experiment and consume the provided feature, while in the context of styling a hand-held device, play-lists are less important.

Visits of Artist Pages

The visit of a user on the artist page – the area of the portal where artist information was presented – was called *viewing* action of an artist. All multiple views of a user, concerning an artist, were eliminated in Table 8.9.

# of	**EU**	**MY**
viewed artists	277.537	1.615.663
distinct artists being viewed (% all artists)	4.537 (49%)	3.329 (87%)
users viewing an artist (% active users)	177.029 (34%)	313.197 (66%)
average viewed artists	1,6	5,2

Table 8.9.: Viewed Artist Statistic

Continuing the trend, that MY users are more active, there is a significant difference between MY and EU concerning the number of viewed artist. MY users are visiting an artist page about three times more frequently than EU users! Furthermore, concerning MY, most available artists (87%) have been viewed!

Figure 8.9 on page 159 and Figure 8.10 on page 159[11] show the viewing frequency of users concerning artists with the sobering result that the majority of the users – 90% in EU and 70% in MY – have only visited two artist pages! More details can be found in Appendix C on page 187.

[11]The 0% markers were caused because of rounding

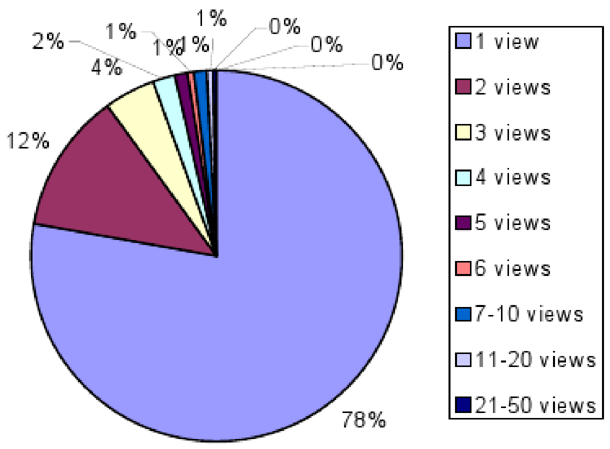


Figure 8.9.: Artist Viewing Frequency in EU

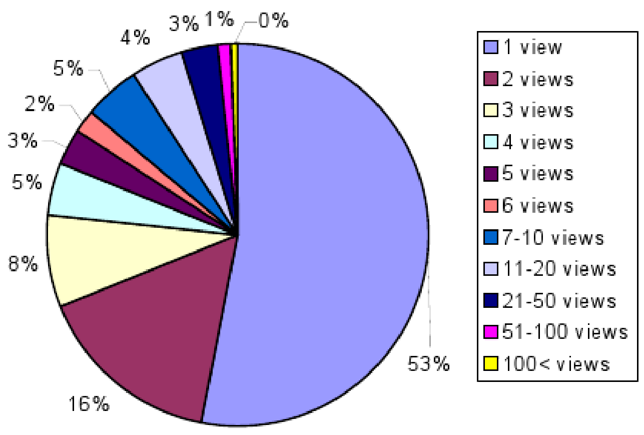


Figure 8.10.: Artist Viewing Frequency in MY

Once again, the significant difference between MY and EU could properly be explained by the multi-tenant setup in EU and the two different usage scenarios.

Visits of Track Pages

The visit of a user on the track page – the area of the portal where track information was presented – was called *viewing* action of a track. All multiple views of a user, concerning a track, were eliminated in Table 8.10 on page 160.

# of	**EU**	**MY**
viewed tracks	823.423	2.814.584
distinct tracks being viewed (% all tracks)	22.808 (15%)	20.905 (58%)
users viewing an track (% active users)	391.318 (76%)	377.230 (79%)
average viewed tracks	2,1	7,5

Table 8.10.: Viewed Track Statistic

Also in the context of viewed tracks MY users are visiting different track-pages about three times more often than EU users! The track viewing frequency of users is shown in Figure 8.11[12] and Figure 8.12. Detailed numbers can be found in Appendix C on page 187.

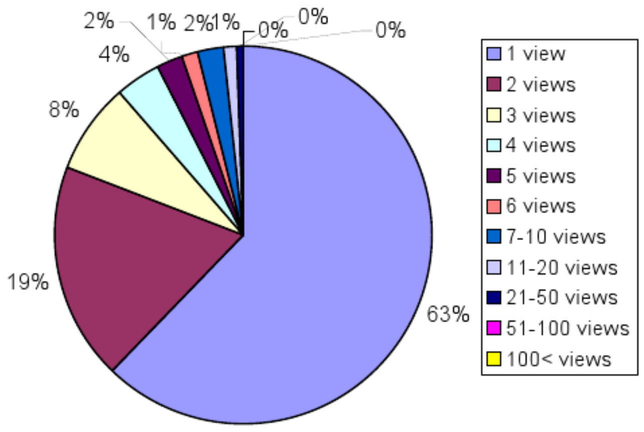


Figure 8.11.: Track Viewing Frequency in EU

Concerning tracks, there is a significant difference between EU and MY, showing that 80% of EU users only visited two track pages, compared to

[12]The 0% markers were caused by rounding

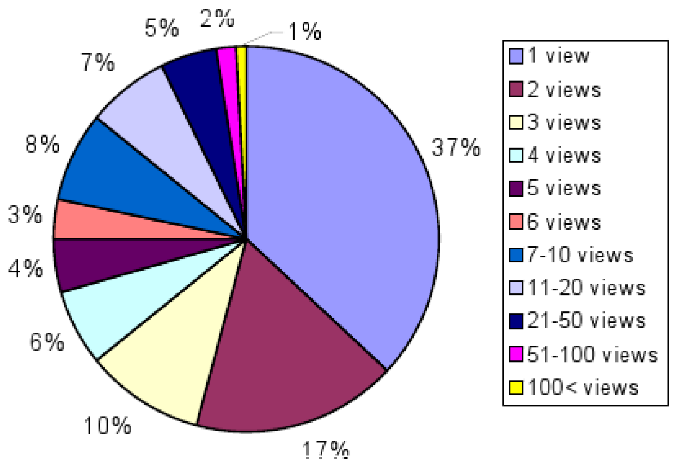


Figure 8.12.: Track Viewing Frequency in MY

more than six page-visits of 80% of MY users!

Visits of Asset Pages

The visit of a user on the asset page – the area of the portal where information of downloadable products was presented – was called *viewing* action of an asset. All multiple views of a user concerning assets were eliminated.

# of	**EU**	**MY**
viewed assets	797.729	2.868.750
distinct assets being viewed (% all assets)	23.278 (14%)	20.166 (44%)
users viewing an asset (% active users)	329.938 (64%)	382.748 (80%)
average viewed assets	2,4	7,5

Table 8.11.: Viewed Assets Statistic

Although the data shown in Table 8.11 is somewhat similar to the track viewing behavior above, some differences exist:

- While in MY the same number of users are viewing tracks and assets (79% vs. 80%) in EU more users are viewing tracks (76%) than assets (64%)
- While in EU the portion of distinct viewed tracks and assets is quite similar (15% to 14%) in MY only a smaller portion of the assets (44%) is viewed, compared to tracks (58%)

A more detailed statistic of viewed assets – concerning the *viewing frequency* – is shown in Figure 8.13 on page 162[13] and Figure 8.14 on page 163. Detailed numbers can be found in Appendix C on page 187.

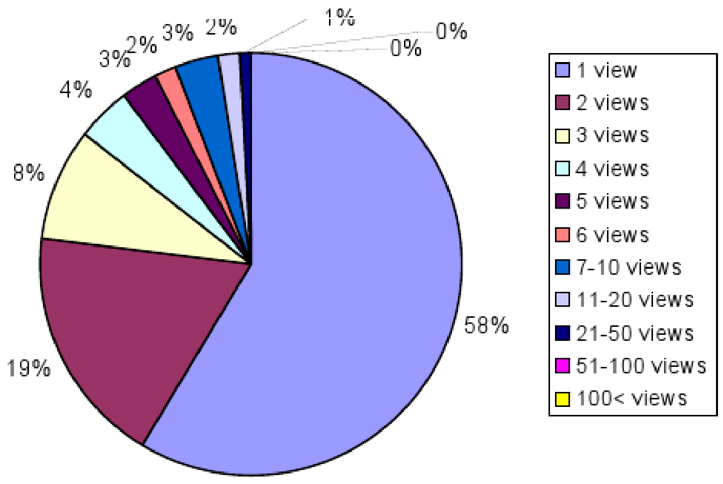


Figure 8.13.: Asset Viewing Frequency in EU

Previewed Assets

A *previewing* action occurred when a user activated the provided preview feature on an asset page (e.g. replaying an audio asset). No preview actions could be identified in the MY database!

# of	EU	MY
viewed assets	3.949	0
distinct assets being viewed (% all assets)	691 (0.4%)	0
users viewing an asset (% active users)	2.176 (0,4%)	0
average viewed assets	1,8	0

Table 8.12.: Pre-viewed Asset Statistic

The data shown in Table 8.12 on page 162 implies the following conclusion:

- Previewing is a hardly used feature in EU: Although the DB entries are well distributed over the analyzed period, less than 0.5% of all

[13]The 0% markers were caused by rounding

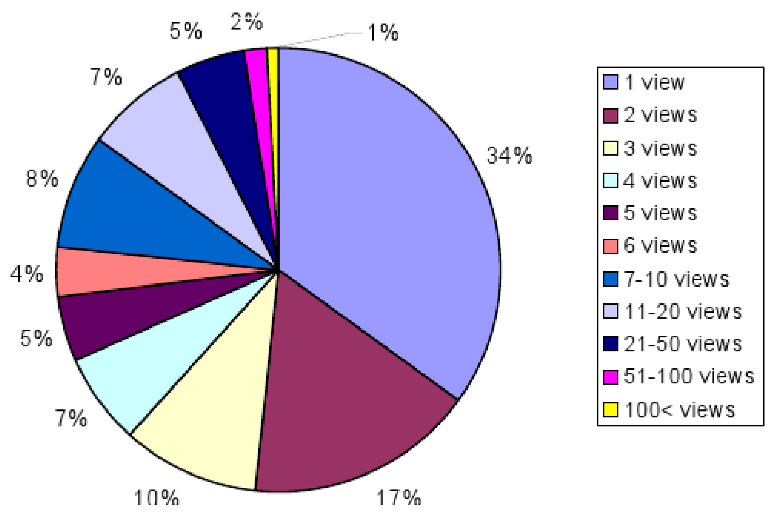


Figure 8.14.: Asset Viewing Frequency in MY

users are using this feature – maybe this is also a consequence of the multi-tenant setup in EU.

- Previewing is not active in MY, or no user never used it, or – most likely – the call to the according RE API is missing.

Artist Rating

A *rating* action of an artist occurred, when a user posted an explicit rating *I like/don't like this artist* on the artist page.

Obviously users tend to give more positive ratings than negative once. Another interesting fact – concerning MY – is, that a large number of artists are rated and that only 1% of the users have rated 32% of all artists negatively while 14% of all users have rated 53% of all artists positive!

Track Rating

A *rating* action of a track occurred, when a user posted an explicit rating *I like/don't like this track* on the track page – see Figure B.5 on page 185.

Obviously positive ratings are preferred to negative once by the community.

Distribution of Recommendation

In this section the calling frequency of the different recommendation strategies (see Section 8.1.2 on page 143) are analyzed.

# of	**EU**	**MY**
like ratings of an artist	18.802	128.165
distinct liked artists (% all artists)	1.622 (18%)	2050 (53%)
users liking an artist (% active users)	15.748 (3%)	67.462 (14%)
average number of *liked* artist per user	1,2	1,9
disliked ratings of an artist	1.345	8.779
distinct disliked artists (% all artists)	454 (5%)	1.219 (32%)
users bad rating an artist (% active users)	1.199 (0,2%)	6.863 (1%)
average number of disliked artist per user	1,1	1,3

Table 8.13.: Good/Bad Rating of Artists

# of	**EU**	**MY**
like ratings of a track	17.392	130.906
distinct liked tracks (% all tracks)	2.414 (1,5%)	6.122 (17%)
users liking a track (% active users)	14.789 (3%)	68.659 (14%)
average liked tracks per user	1,2	1,9
dislike ratings of a track	4.838	13.453
distinct disliked tracks (% all tracks)	1.801 (1%)	3.486 (9%)
of users disliking a track (% active users)	2.461 (0,4%)	8.779 (2%)
average disliked tracks per user	1,9	1,5

Table 8.14.: Good/Bad Rating of Tracks

Recommendation	**# of calls**
Other tracks you might like	13.411
Assets, other users also bought	9.953
Hot Recommendation	9.121
Other artists you might like	1.003
Songs, similar users like	85

Table 8.15.: Number of Recommendation Requests in MY

Although the recommendation strategies – *Songs, similar users like* and *Hot Recommendation* – are both accessable within the *My Mobile Music* area the former recommendation is hardly used compared with the latter one.

8.4. Contributions to the Research Question

Although the provided data base is inadequate for detailed analysis and conclusions it can be used as an indicator in which direction further investigations should be aligned. From our point of view, the most important results of this analysis are, that the results of MY can be seen as an indicator that the basic concepts of the Adaptive Personalization are valuable approaches, accepted by the community. In more detail:

- Users are willing to create personal profiles – 14% of all active users created a SELF profile. Although this value could not be compared to other platforms[14] it was considered as very high by the responsible product management and sales departments of the EMM, who are used to conversion rates between 3% and 5% concerning optional features.
- The multi-view approach seems to be a value, because a significant difference between the genre preferences collected in the SELF and OBSRV profile view could be identified. Due to that difference more/additional recommendations can be presented to the user together with an appropriate explanation.
- Item tuning is an accepted feature, unburdening the administrator from maintaining the content.

Especially for MY the engagement of the community in the content generation and/or quality refinement process can be observed, items were tuned and feedback were given. In MY, 43% of all artists and 16% of all tracks were classified, mainly by the community! Therefore, the conclusion can be drawn that the community-based classification of items, here implemented as *Mood&Situation* clusters, is a well accepted feature. The partially significant differences between EU and MY must be analyzed in more detail. Some differences, mainly concerning content and divergent set of features, could be explained by the multi-tenant situation in EU.

The typical *active user* purchases two assets, views 1.5(EU)/5(MY) artists and 2(EU)/7(MY) tracks. This implies that the users are no

[14]At the time writing this analysis, statistics of other mobile music platforms were hardly available.

heavy/frequent users (especially for EU) and therefore the development and adaptation of a profile over time is less important. Concerning personalization this means, that there are only few chances to convince users of the quality of recommendations. This can be seen as an affirmation of the profile refinement approach, presented in Chapter 6, and its focusing on a fast creation of a useful preference model.

Furthermore, due to the very diverse characteristics of these two installations, more efforts should be directed to an ongoing tuning/monitoring/-analyzing process concerning recommendation quality, especially during the *warm-up/cold-start* phase of the system.

9. Future Work

In this chapter some directions for further research concerning core concepts of the *Adaptive Personalization* approach are discussed.

9.1. Adaptive Profile Model

As mentioned in Section 6.5 on page 100, the strength of the *Adaptive Profile Model* is its multi-view approach, incorporating contextual information concerning the origin of a user action, combined with the ability to define the adaptive or learning behavior precisely. Based on the fact, that this adaptive behavior is limited to predefined clusters or concepts, for which an appropriate dimension has to be defined, this approach should be seen as an extension or supplement to standard profile modelling approaches relying on machine learning techniques. Furthermore, our multi-view model should be combined with other modeling approaches incorporating contextual aspects. For example, the *Unified User Context Model* (UUCM), developed by Niederée et al. (2004), could be a good candidate for such a combination, because of it's multi-dimensional model, the usage of cognitive patterns and the capturing of the fact, that users will interact with a system in different working contexts.

9.2. Recommendation Algorithms

One drawback of the collaborative-filtering algorithm discussed in Section 7.1 on page 105 is its potentially bad runtime behavior when calculating the contribution of mainstream items to the similarity correlation. The paradox of this situation is, that the less an item is contributing to the similarity relation the more resources are needed to perform this calculation, because of the $O(n^2)$ of the pairwise similarity calculation of all evaluators of an item. This is especially problematic in contexts, where a large number of users (e.g., millions) are rating few items, thus creating typically "top lists" such as the *Top 40* songs in the music domain. Possible strategies to solve this problem are:

1. ignoring every-body's-darlings (items having a $disc(I)$ value below a given threshold) during similarity computation
2. only taking the n most important items of each user, where n should be as small as possible (e.g., the 7 most important items of each user)

Although both approaches are focusing on important items, the latter tends to stress user similarities defined by rare and high rated items, which might lead to fewer but very valuable suggestions.

Another important topic is the consideration of time series in generating recommendations. Especially in domains where the order of item consumption will have an impact on the user ratings (e.g., in the domain of books or articles) the consideration of a proper (time) order can be very important. A person, being a rookie in a new domain will be overburdened when confronted with an in-depth key paper instead of an easy readable introduction. So the question arises, if and how this information can be used to find the best timely order in which items should be recommended to the user. Furthermore, these series can be seen as preference trajectories which can be used not only to predict items but also to recommend a complete preference evolution path to a user. This might especially be valuable if the items can be clustered, because suggestions for the mid- and long-term evolution of user preferences could be generated additionally.

10. Conclusion

Information overload is a serious problem with economic dimensions affecting a rapidly growing number of people in a variety of circumstances. Amongst others, personalization can be seen as a major concept to face this problem. The creation of a user model and the derivation of predictions from this model is the core task of a personalization system, requiring appropriate information and adequate techniques. Thus, the careful and broad usage of all the available information, implicitly or explicitly, is of fundamental importance, especially in domains with no or poor meta data (e.g., mobile commerce).

In this thesis we presented a new approach, called *Adaptive Personalization*, integrating contextual information for improving user and item models, as well as recommendation techniques. The core concepts of the *Adaptive Personalization* approach were developed and presented along the following three research questions as defined in Section 1.2 on page 4:

1. Which already known techniques/approaches exist and why they are not sufficient for the given purpose?
2. How can contextual information be used to improve user and item models?
3. How can contextual information be used to improve rating based recommendation strategies?

The first research question was tackled in Chapter 3, where a variety of different strategies for implementing personalization systems were presented. Most of these approaches were successfully applied to certain problems, thus demonstrating the power of personalization techniques. However, our main criticism is, that these approaches do not use all the information available, even in situations with minimal information. This is perhaps not a big issue in classical e-commerce applications based on standard Web interfaces, but a problem in domains with limited access or typical walk-in customers (e.g., m-commerce). Contextual solutions as presented by Adomavicius and Tuzhilin (2005) or Niederée et al. (2004) mainly discuss the detection or modeling of different *contexts of use* (e.g., at home, at work, with friends, etc.) but do not consider the *context of origin* of user feedback (e.g., self-assessment of a user, observed user behavior, etc.). Furthermore, collaborative filtering techniques rely on user ratings and the popularity of items (e.g., *inverse user frequency* improvement – see Definition 3.4 on page 24) but they do not use all aspects of the set characteristics of the user ratings (e.g., the overlapping of two rating sets, size of the ratings or preference sets, etc.)

The second research question was addressed in Chapter 6. Inspired by a psychological model created by Joseph Luft and Harry Ingham – the Johari Window (Luft and Ingham, 1955) – a multi-view profile model was developed introducing the *context of origin* of user feedback. Concerning user profiles the three views – *self-assessment*, *system observation* and *community-assessment* – are used for optimizing the recommendation process for satisfying the well and ill-defined needs as well as for supporting comprehensible explanation models. Furthermore, the incorporation of long- and short-term aspects of preferences (context of sessions) helps to build more stable user models, because context driven variations of a user's behavior (e.g., searching for a gift for a friend) can be covered much better. Evaluations performed on data gathered from real world systems (see Chapter 8) approved the effectiveness of this multi-view approach, by identifying a significant set of non-overlapping preferences concerning the *self-assessment* and *system observation* view. Additionally, the presented three fold item model supports the responsible person in maintaining valuable content meta data which will lead to better recommendations.

In Chapter 7 the third research questions was tackled by showing how existing contextual information, concerning the usage of ratings and rating sets, can be used to improve recommendation algorithms. The *pretty good recommendation* approach, presented in the first section, incorporated information concerning the importance of an item in local (per user) and global (for all users) contexts as well as in the context of the pairwise relations of preference sets. We showed, based on a standard data set for recommender systems, that our algorithm outperforms standard techniques, especially in the context of less information.
The D^2-Tree, presented in the second section of Chapter 7, is a custom-made complement for the *Adaptive Profile Model* providing an efficient solution for the k-nearest neighbor problem. The most important advantages of the D^2-Tree are the support of weighted requests and the ability to incorporate domain knowledge even dynamically, thus supporting the context of use concerning a specific request.

Furthermore, a bird's-eye-view of the *Adaptive Personalization* was provided in Chapter 4 by introducing the core aspects as well as other important topics such as an appropriate procedure model or the descriptions of applied recommendation strategies. A flexible architecture, concerning the profile module as well as the whole system, was presented in Chapter 5, focusing on ease of use, scalability and extendability. An extensive, qualitative evaluation of the *Adaptive Personalization* approach was presented in Chapter 8, based on real world data sets provided by an implementation of our approach, indicating the usefulness of the core concepts. Further research issues mainly concerning the multi-view profile model as well as the collaborative-filtering algorithm were presented in Chapter 9.

In this thesis, we proposed a new approach for user and item modeling, a new collaborative-filtering algorithm, a new index structure for solving the k-nearest neighbor problem together with a proposal for an appropriate and flexible architecture. Furthermore, we evaluated the performance of our concepts using different approaches and proved their effectiveness. Due to the new and broad utilization of the implicitly and explicitly available information, the concepts of the *Adaptive Personalization* can easily be combined with other personalization approaches or techniques.

Summing up, we believe, that the concepts of the *Adaptive Personalization* approach are useful and valuable contributions to the research concerning personalization and recommender systems providing support in handling the problem of information overload efficiently.

List of Figures

List of Tables

Listings

A. MAE Data Tables

Table A.1.: MAE data table

%Users	PGC Rating	PGC Rel Rating	PGC Adj Rating	Pearson
2	0.768453124	0.776312856	0.765740322	0.960838695
4	0.748419652	0.749547622	0.745901904	0.92169267
6	0.740386243	0.739413385	0.738277257	0.89332819
8	0.736463957	0.734397346	0.733976112	0.868696106
10	0.734686671	0.7315081	0.731963045	0.845937727
12	0.732947434	0.729293795	0.73040897	0.824837978
14	0.731946926	0.727883357	0.729813715	0.808983961
16	0.731488496	0.726824386	0.729513249	0.793988516
18	0.731237625	0.726107946	0.729147018	0.782305473
20	0.731030287	0.725623762	0.728989759	0.771587487
22	0.730998513	0.725280823	0.728978642	0.764091464
24	0.730821598	0.725093178	0.728927673	0.758792846
26	0.730751665	0.725073108	0.728953013	0.754171959
28	0.730800837	0.724989272	0.729007513	0.750370062
30	0.730846999	0.724762348	0.728922366	0.747931549
32	0.730746657	0.724716116	0.72900674	0.745944253
34	0.730652624	0.724721776	0.728994243	0.74421694
36	0.730694653	0.724736562	0.729023135	0.743286401
38	0.730666978	0.724644946	0.728931262	0.74232802
40	0.730680034	0.724625176	0.728958476	0.741617617
42	0.730575993	0.724580368	0.728888744	0.741052719
44	0.730585931	0.724589944	0.728923374	0.740768129
46	0.730566263	0.724589604	0.728880146	0.740542201
48	0.730448715	0.72458436	0.728888522	0.740354449
50	0.730402054	0.724557237	0.728942136	0.740062108
52	0.730479356	0.724587384	0.72887551	0.739832934
54	0.730391985	0.724594454	0.728842994	0.739727955
56	0.730403212	0.72470702	0.728851166	0.739832021
58	0.730373196	0.724847667	0.728890994	0.739858362
60	0.730333754	0.725087218	0.72889822	0.739894684
62	0.730323636	0.725384561	0.728886691	0.739882285
64	0.730335981	0.725961711	0.728857411	0.739846415
66	0.730285485	0.726601698	0.728875769	0.739805264
68	0.730258179	0.727387865	0.728868297	0.739732715

continued on next page

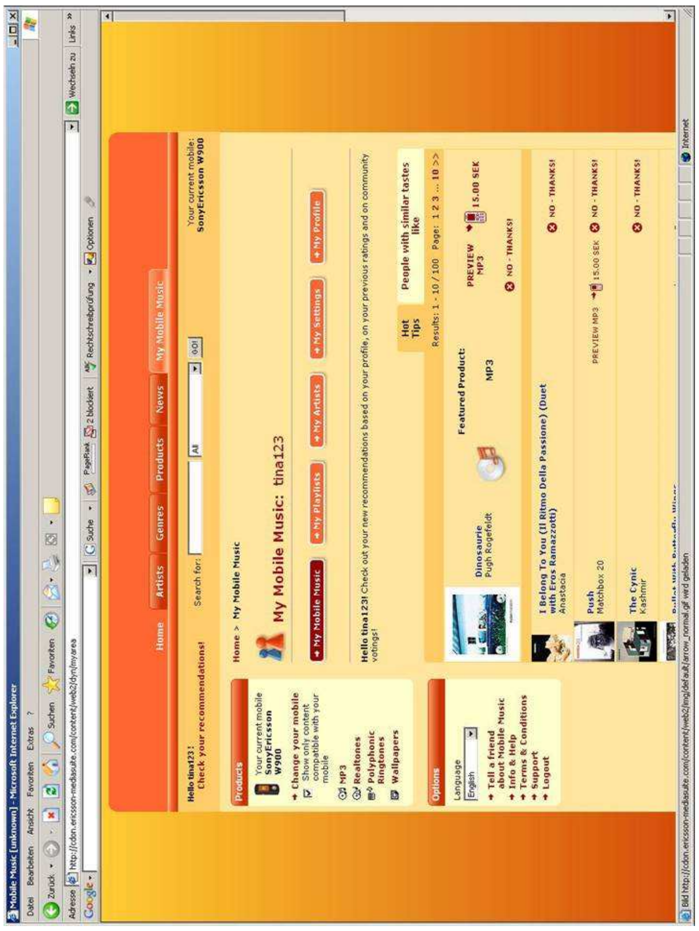
%Users	PGC Rating	PGC Rel Rating	PGC Adj Rating	Pearson
70	0.730266202	0.728666422	0.728848072	0.739643285
72	0.730216159	0.730428495	0.728821375	0.739639567
74	0.730216718	0.732921495	0.728822875	0.739643587
76	0.730247551	0.736548515	0.728799377	0.739656914
78	0.730230387	0.741523843	0.728793185	0.739986711
80	0.730197141	0.747960674	0.728779297	0.740351356
82	0.730192994	0.756750809	0.728756911	0.740796408
84	0.73019	0.768255181	0.728754123	0.741439534
86	0.730173822	0.783944296	0.728759381	0.742078508
88	0.730162221	0.804836585	0.728761148	0.74293368
90	0.73017603	0.831918247	0.728779555	0.743813925
92	0.730182647	0.866903825	0.728810221	0.744726412
94	0.730175152	0.913343282	0.728800373	0.745848378
96	0.730201296	0.975858625	0.728833291	0.746896308
98	0.730195391	1.062518388	0.728841513	0.747885578
100	0.73018516	1.222174231	0.72892911	0.74894205

B. Samples from Online Systems

In this chapter some screen shots of the (white labeled) *Ericsson's Media Suite - Music (EMM)* are presented to the convenience of the reader.

B.1. Preference-Based Recommendations

The *Hot Tips* recommendation, as presented in Figure B.1, is based on the users preferences and was provided to registered users only.


Mobile Music [unknown] - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse http://cdon.ericsson-mediasuite.com/content/web2/dyn/myarea
Wechseln zu Links
Suche PageRank 2 blockiert Rechtschreibprüfung Optionen
Home
Artists
Genres
Products
News
My Mobile Music
Search for: All GO!
Products
Options
Language English
My Mobile Music
My Playlists
My Artists
My Settings
My Profile
Hot Tips
People with similar tastes like
PREVIEW MP3
NO - THANKS!
Bild http://cdon.ericsson-mediasuite.com/content/web2/img/default/arrow_normal.gif wird geladen
Internet

Hello tina123!
Check your recommendations!

Your current mobile: SonyEricsson W900

Your current mobile SonyEricsson W900
Change your mobile
Show only content compatible with your mobile
MP3
Realtones
Polyphonic Ringtones
Wallpapers

Tell a friend about Mobile Music
Info & Help
Terms & Conditions
Support
Logout

Home > My Mobile Music

My Mobile Music: tina123

Hello tina123! Check out your new recommendations based on your profile, on your previous ratings and on community votings!

Results: 1 - 10 / 100 Page: 1 2 3 ... 10 >>

Dinosaurie
Pugh Rogefeldt
Featured Product: MP3
PREVIEW MP3 15.00 SEK
NO - THANKS!

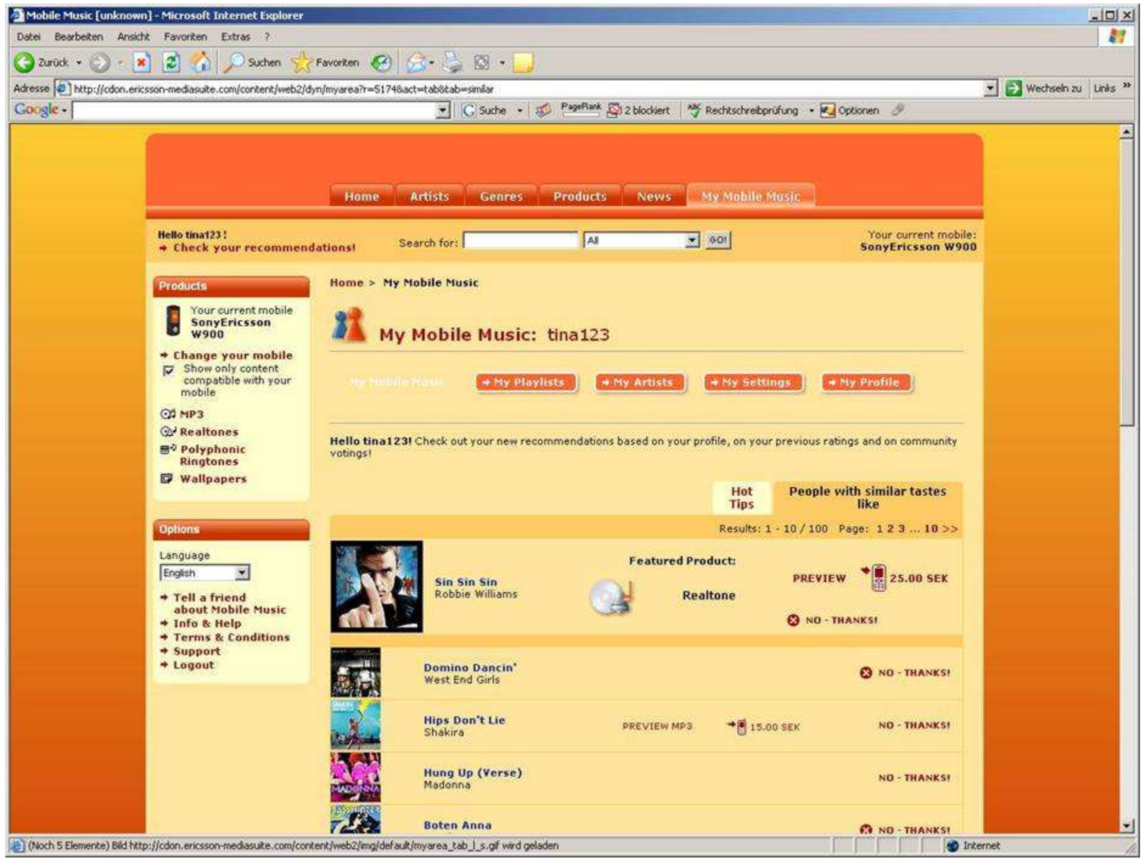
I Belong To You (Il Ritmo Della Passione) (Duet with Eros Ramazzotti)
Anastacia
NO - THANKS!

Push
Matchbox 20
PREVIEW MP3 15.00 SEK
NO - THANKS!

The Cynic
Kashmir
NO - THANKS!

Figure B.1.: Hot Tips Recommendation List

The *People with similar tastes like* recommendation, as presented in Figure B.2, is based on the users preferences and socio demographic data and was provided to registered users only.



Mobile Music [unknown] - Microsoft Internet Explorer

Adresse http://cdon.ericsson-mediasuite.com/content/web2/dyn/myarea?r=5174&act=tab&tab=similar

Home | Artists | Genres | Products | News | My Mobile Music

Hello tina123 !
→ Check your recommendations!

Search for: All GO!

Your current mobile:
SonyEricsson W900

Products

Your current mobile
SonyEricsson W900

→ Change your mobile
Show only content compatible with your mobile

MP3
Realtones
Polyphonic Ringtones
Wallpapers

Options

Language English

→ Tell a friend about Mobile Music
→ Info & Help
→ Terms & Conditions
→ Support
→ Logout

Home > My Mobile Music

My Mobile Music: tina123

My Mobile Music | → My Playlists | → My Artists | → My Settings | → My Profile

Hello tina123! Check out your new recommendations based on your profile, on your previous ratings and on community votings!

Hot Tips | People with similar tastes like

Results: 1 - 10 / 100 Page: 1 2 3 ... 10 >>

Sin Sin Sin
Robbie Williams

Featured Product:
Realtone

PREVIEW 25.00 SEK

NO - THANKS!

Domino Dancin'
West End Girls
NO - THANKS!

Hips Don't Lie
Shakira
PREVIEW MP3 15.00 SEK
NO - THANKS!

Hung Up (Verse)
Madonna
NO - THANKS!

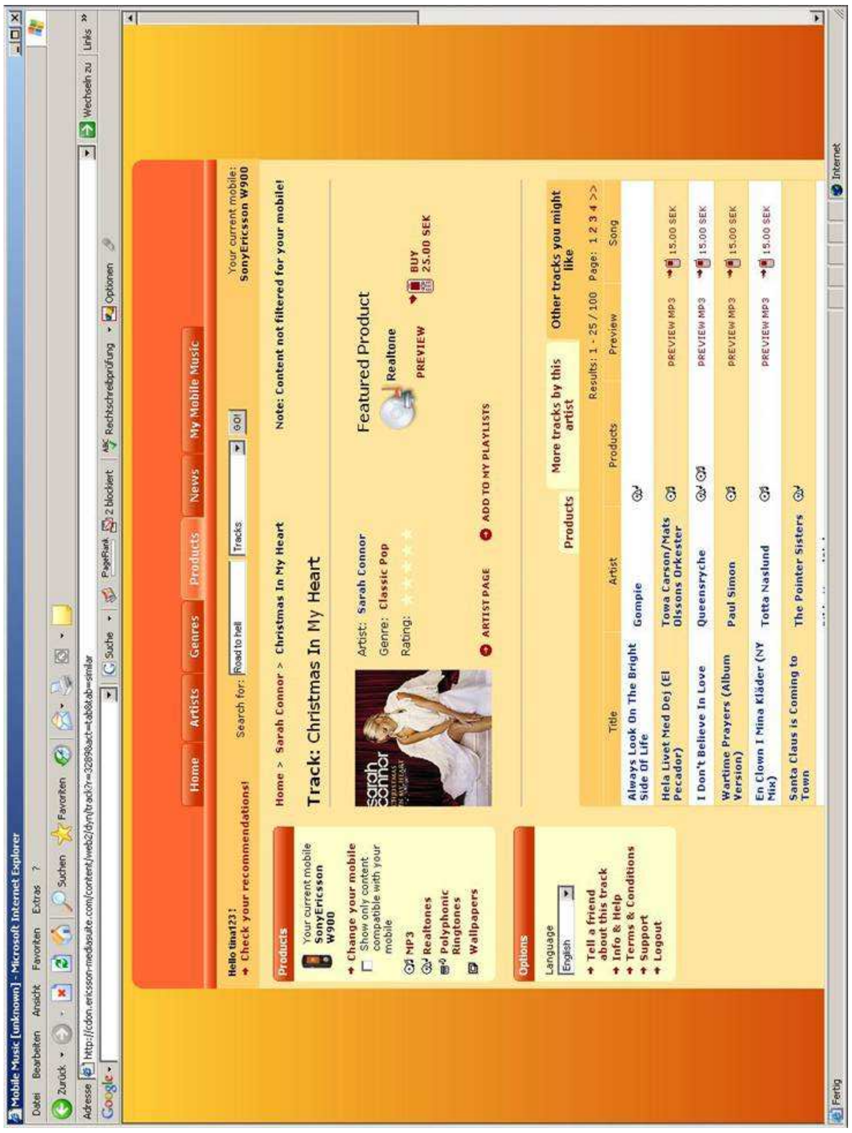
Boten Anna
NO - THANKS!

(Noch 5 Elemente) Bild http://cdon.ericsson-mediasuite.com/content/web2/img/default/myarea_tab_l_s.gif wird geladen

Figure B.2.: Similar Users Like Recommendation

B.2. Item-Based Recommendations

The *Other tracks you might like* recommendation, as presented in Figure B.3, is based on different item similarities and was provided to registered and non-registered users.


Mobile Music [unknown] - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse http://cdon.ericsson-mediasuite.com/content/web2/dyn/track?r=3289&act=tab&tab=similar
Wechseln zu Links
Home Artists Genres Products News My Mobile Music
Products
Options
Fertig
Internet

Hello tina123 !
Check your recommendations!

Search for: Road to hell | Tracks | GO!

Your current mobile: SonyEricsson W900

Home > Sarah Connor > Christmas In My Heart

Note: Content not filtered for your mobile!

Track: Christmas In My Heart

Artist: Sarah Connor
Genre: Classic Pop
Rating:

Featured Product
Realtone
PREVIEW
BUY 25.00 SEK

ARTIST PAGE
ADD TO MY PLAYLISTS

Your current mobile SonyEricsson W900
Change your mobile
Show only content compatible with your mobile
MP3
Realtones
Polyphonic Ringtones
Wallpapers

Language: English
Tell a friend about this track
Info & Help
Terms & Conditions
Support
Logout

Products | More tracks by this artist | Other tracks you might like

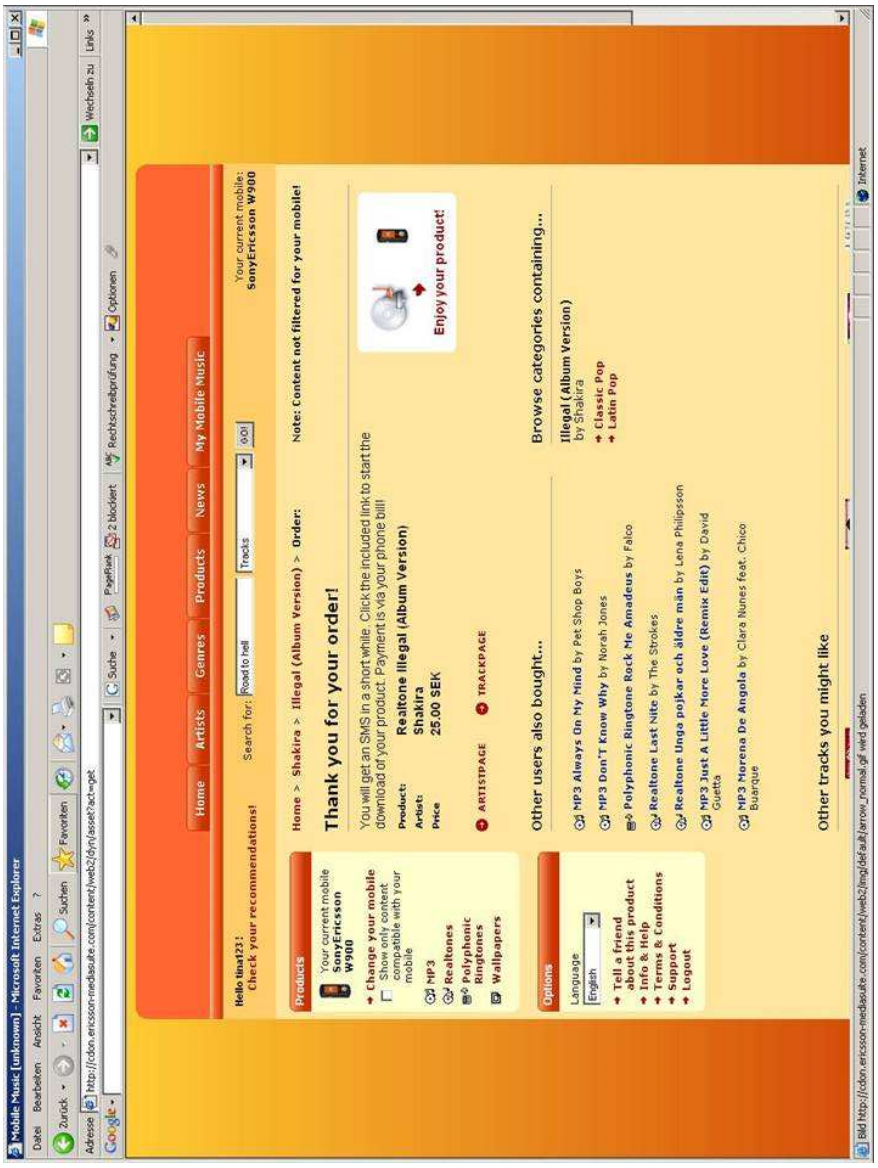
Results: 1 - 25 / 100 Page: 1 2 3 4 >>

Title	Artist	Products	Preview	Song
Always Look On The Bright Side Of Life	Gompie			
Hela Livet Med Dej (El Pecador)	Towa Carson/Mats Olssons Orkester		PREVIEW MP3	15.00 SEK
I Don't Believe In Love	Queensryche		PREVIEW MP3	15.00 SEK
Wartime Prayers (Album Version)	Paul Simon		PREVIEW MP3	15.00 SEK
En Clown I Mina Kläder (NY Mix)	Totta Naslund		PREVIEW MP3	15.00 SEK
Santa Claus is Coming to Town	The Pointer Sisters			

Figure B.3.: Other Tracks a User Might Like

B.3. Statistic-Based Recommendations

The *Other users also bought* recommendation, as presented in Figure B.4, is based on shopping cart analysis algorithms and was provided to registered and non-registered users.


Mobile Music [unknown] - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück Suchen Favoriten
Adresse http://cdon.ericsson-mediasuite.com/content/web2/dyn/asset?act=get
Wechseln zu Links
Google Suche PageRank 2 blockiert Rechtschreibprüfung Optionen
Home Artists Genres Products News My Mobile Music
Search for: Road to hell Tracks GO!
Products
MP3
Realtones
Polyphonic Ringtones
Wallpapers
Options
Language English
Tell a friend about this product
Info & Help
Terms & Conditions
Support
Logout
ARTISTPAGE
TRACKPAGE
Enjoy your product!
Bild http://cdon.ericsson-mediasuite.com/content/web2/img/default/arrow_normal.gif wird geladen
Internet

Hello tina123 !
Check your recommendations!

Your current mobile:
SonyEricsson W900

Your current mobile
SonyEricsson
W900

Change your mobile
Show only content compatible with your mobile

Home > Shakira > Illegal (Album Version) > Order:

Note: Content not filtered for your mobile!

Thank you for your order!

You will get an SMS in a short while. Click the included link to start the download of your product. Payment is via your phone bill!

Product:	Realtone Illegal (Album Version)
Artist:	Shakira
Price	25.00 SEK

Other users also bought...

- MP3 Always On My Mind by Pet Shop Boys
- MP3 Don'T Know Why by Norah Jones
- Polyphonic Ringtone Rock Me Amadeus by Falco
- Realtone Last Nite by The Strokes
- Realtone Unga pojkar och äldre män by Lena Philipsson
- MP3 Just A Little More Love (Remix Edit) by David Guetta
- MP3 Morena De Angola by Clara Nunes feat. Chico Buarque

Browse categories containing...

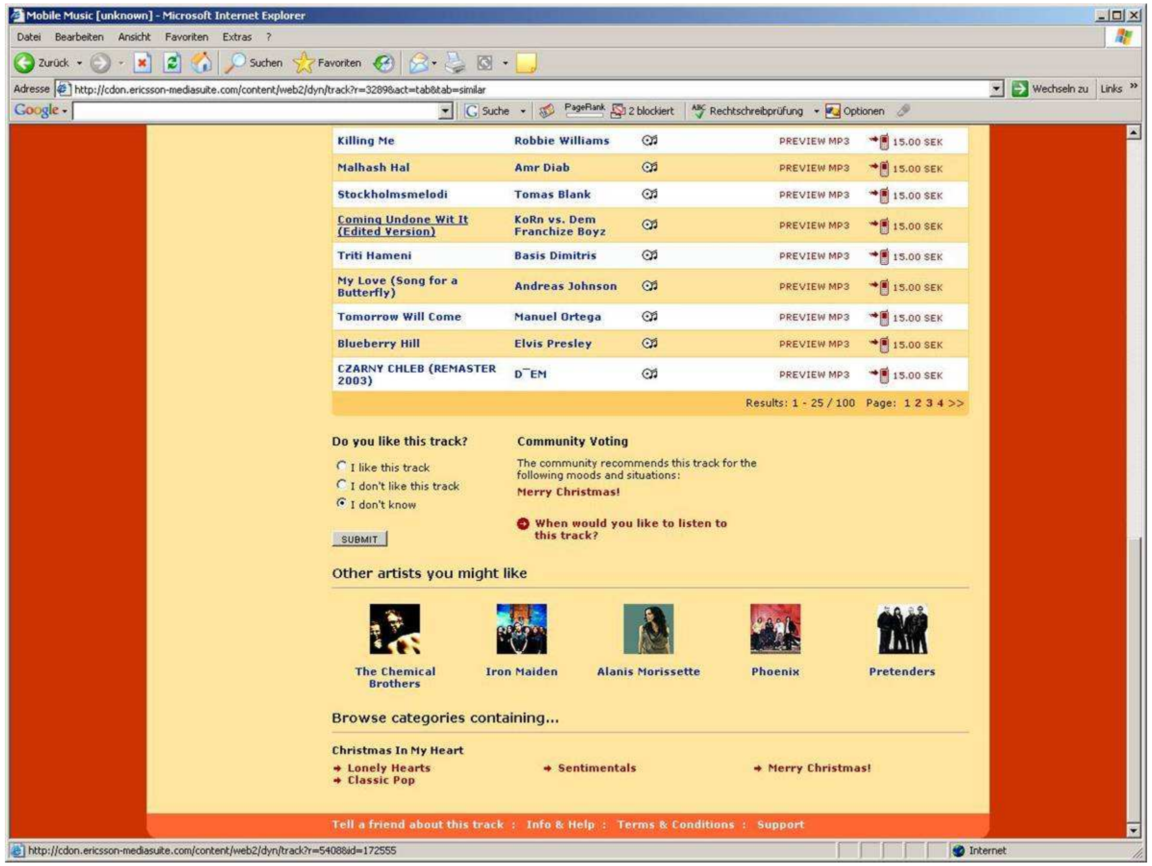
Illegal (Album Version)
by Shakira
- Classic Pop
- Latin Pop

Other tracks you might like

Figure B.4.: Products also Bought

B.4. User Feedback

As presented in Figure B.5, a user could rate an item (e.g., a track, an artist) by choosing one of three alternatives, e.g., *I like this track*, *I don't like this track*, *I don't know*.


Mobile Music [unknown] - Microsoft Internet Explorer
Datei Bearbeiten Ansicht Favoriten Extras ?
Zurück
Suchen
Favoriten
Adresse http://cdon.ericsson-mediasuite.com/content/web2/dyn/track?r=3289&act=tab&tab=similar
Wechseln zu
Links
Google
Suche
PageRank
2 blockiert
Rechtschreibprüfung
Optionen

Killing Me	Robbie Williams	PREVIEW MP3	15.00 SEK
Malhash Hal	Amr Diab	PREVIEW MP3	15.00 SEK
Stockholmsmelodi	Tomas Blank	PREVIEW MP3	15.00 SEK
Coming Undone Wit It (Edited Version)	KoRn vs. Dem Franchize Boyz	PREVIEW MP3	15.00 SEK
Triti Hameni	Basis Dimitris	PREVIEW MP3	15.00 SEK
My Love (Song for a Butterfly)	Andreas Johnson	PREVIEW MP3	15.00 SEK
Tomorrow Will Come	Manuel Ortega	PREVIEW MP3	15.00 SEK
Blueberry Hill	Elvis Presley	PREVIEW MP3	15.00 SEK
CZARNY CHLEB (REMASTER 2003)	D¯EM	PREVIEW MP3	15.00 SEK

Results: 1 - 25 / 100 Page: 1 2 3 4 >>

Do you like this track?

- I like this track
- I don't like this track
- [✓ I don't know]

SUBMIT

Community Voting

The community recommends this track for the following moods and situations:
Merry Christmas!

When would you like to listen to this track?

Other artists you might like

The Chemical Brothers · Iron Maiden · Alanis Morissette · Phoenix · Pretenders

Browse categories containing...

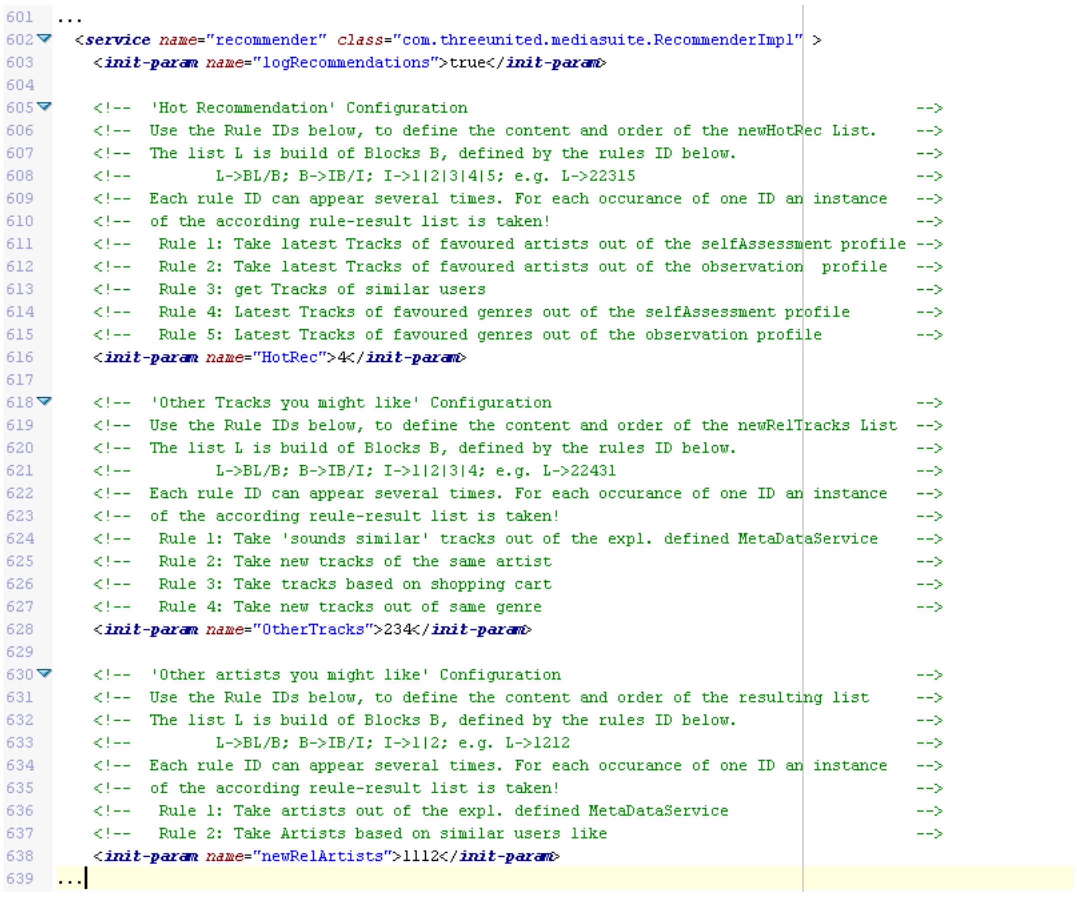
Christmas In My Heart
- Lonely Hearts
- Classic Pop
- Sentimentals
- Merry Christmas!

Tell a friend about this track : Info & Help : Terms & Conditions : Support

http://cdon.ericsson-mediasuite.com/content/web2/dyn/track?r=5408&id=172555
Internet

Figure B.5.: User Feedback on a Track

B.5. Configuration



```
...
  <service name="recommender" class="com.threeunited.mediasuite.RecommenderImpl" >
    <init-param name="logRecommendations">true</init-param>

    <!--  'Hot Recommendation' Configuration                                          -->
    <!--  Use the Rule IDs below, to define the content and order of the newHotRec List.    -->
    <!--  The list L is build of Blocks B, defined by the rules ID below.              -->
    <!--         L->BL/B; B->IB/I; I->1|2|3|4|5; e.g. L->22315                        -->
    <!--  Each rule ID can appear several times. For each occurance of one ID an instance   -->
    <!--  of the according rule-result list is taken!                                 -->
    <!--   Rule 1: Take latest Tracks of favoured artists out of the selfAssessment profile -->
    <!--   Rule 2: Take latest Tracks of favoured artists out of the observation  profile   -->
    <!--   Rule 3: get Tracks of similar users                                        -->
    <!--   Rule 4: Latest Tracks of favoured genres out of the selfAssessment profile       -->
    <!--   Rule 5: Latest Tracks of favoured genres out of the observation profile          -->
    <init-param name="HotRec">4</init-param>

    <!--  'Other Tracks you might like' Configuration                                 -->
    <!--  Use the Rule IDs below, to define the content and order of the newRelTracks List  -->
    <!--  The list L is build of Blocks B, defined by the rules ID below.              -->
    <!--         L->BL/B; B->IB/I; I->1|2|3|4; e.g. L->22431                          -->
    <!--  Each rule ID can appear several times. For each occurance of one ID an instance   -->
    <!--  of the according reule-result list is taken!                                -->
    <!--   Rule 1: Take 'sounds similar' tracks out of the expl. defined MetaDataService    -->
    <!--   Rule 2: Take new tracks of the same artist                                 -->
    <!--   Rule 3: Take tracks based on shopping cart                                 -->
    <!--   Rule 4: Take new tracks out of same genre                                  -->
    <init-param name="OtherTracks">234</init-param>

    <!--  'Other artists you might like' Configuration                                -->
    <!--  Use the Rule IDs below, to define the content and order of the resulting list     -->
    <!--  The list L is build of Blocks B, defined by the rules ID below.              -->
    <!--         L->BL/B; B->IB/I; I->1|2; e.g. L->1212                               -->
    <!--  Each rule ID can appear several times. For each occurance of one ID an instance   -->
    <!--  of the according reule-result list is taken!                                -->
    <!--   Rule 1: Take artists out of the expl. defined MetaDataService              -->
    <!--   Rule 2: Take Artists based on similar users like                           -->
    <init-param name="newRelArtists">1112</init-param>
...
```

Figure B.6.: Extraction of the Recommender Configuration

C. Data from Online Systems

C.1. Profile Views

Table C.1 contains statistics concerning the three profile views, where SELF is the placeholder for *self-assessment*, OBSERV for *observation view* and COMB stands for *combined profile*.

	EU	**MY**	**Description**
# SELF profiles (%of active users)	9.172 (1.7%)	70.782 (15%)	
# OBSRV profiles (% of active users)	424.804 (82%)	401.764 (85%)	
# COMB profiles (%of active users)	63.074 (12%)	151.282 (32%)	
AVG size SELF	8	8	# genre ids
AVG size OBSRV	5	8	# genre ids
AVG size COMB	9	8	# genre ids
AVG DIFF(SELF-OBSERV)	3	6	# genre ids (not over-lapping)
# size SELF == size COMB (%SELF)	6.476 (70%)	32.963 (47%)	
# size SELF < COMB (%SELF)	2.685 (29%)	30.072 (42%)	

Table C.1.: Profile Statistic

C.2. Purchase Statistic

Table C.2 contains the data how many users purchased how many items in EU and MY.

# of purchased assets	# of customers EU	# of customers MY
1	94.792	86.533
2	23.059	29.221
3	9.125	12.817
4	4.462	6.592
5	2.633	3.741
6	1.722	2.340
7-10	3.176	3.896
11-20	1.756	1.974
21-50	666	637
51-100	91	92
100 -	20	25

Table C.2.: Purchase Frequency

C.3. Artist Viewing Statistic

Table C.3 contains the information how many users viewed how many different artists in EU and MY

# of viewed artists	# of customers EU	# of customers MY
1	137.636	165.806
2	21.880	49.938
3	7.559	24.037
4	3.662	14.198
5	1.853	9.211
6	1.217	6.487
7-10	1.852	14.578
11-20	1.003	13.650
21-50	312	10.237
51-100	49	3.516
100 -	6	1.449

Table C.3.: Artist Viewing Frequency

C.4. Track Viewing Statistic

Table C.4 contains the information how many users viewed how many different tracks.

# of viewed tracks	# of customers EU	# of customers MY
1	243.895	139.700
2	72.283	65.045
3	30.255	37.414
4	15.294	23.822
5	8.808	16.792
6	5.347	12.492
7-10	9.039	28.981
11-20	4.626	25.551
21-50	1.498	17.700
51-100	210	6.314
100 -	63	3.419

Table C.4.: Track Viewing Frequency

C.5. Asset Viewing Statistic

Table C.5 contains the information how many users viewed how many different assets.

# of viewed assets	# of customers EU	# of customers MY
1	193.151	133.784
2	61.102	64.173
3	27.609	38.293
4	14.574	25.575
5	8.831	18.292
6	5.753	13.489
7-10	10.376	31.641
11-20	5.997	28.432
21-50	2.169	19.652
51-100	310	6.442
100 -	66	2.975

Table C.5.: Asset Viewing Frequency

C.6. Asset Pre-Viewing Statistic

Table C.6 contains the information how many users viewed how many different assets.

# of viewed assets	# of customers EU	# of customers MY
1	1426	
2	399	
3	141	
4	75	
5	52	
6	27	
7-10	35	
11-20	18	
21-50	3	
51-100	0	
100 -	0	

Table C.6.: Asset PreViewing Frequency

C.7. Genre Distribution of Tracks

The distribution of tracks concerning genres is shown in the following diagrams. For reason of clarity the diagrams were separated into two parts where the Y-scaling of the second diagram was reduced by a factor of 100!

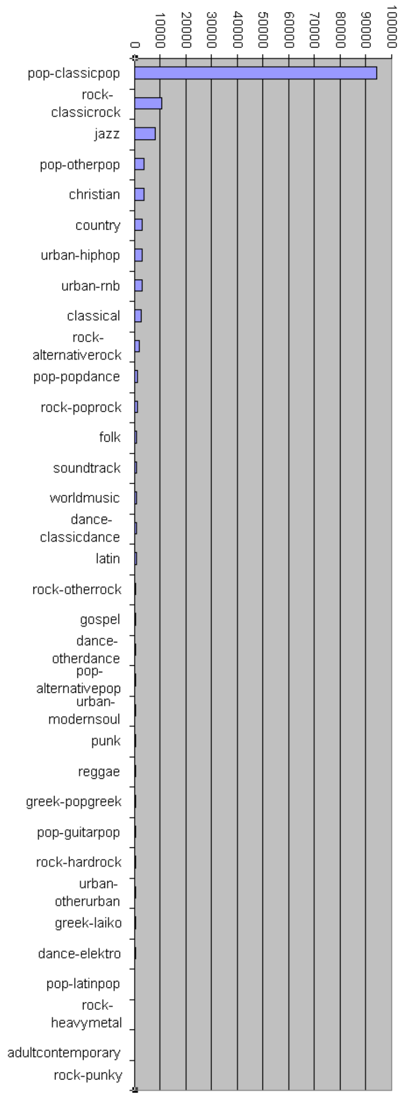


Figure C.1.: Track – Genre Distribution in Europe (1st part)

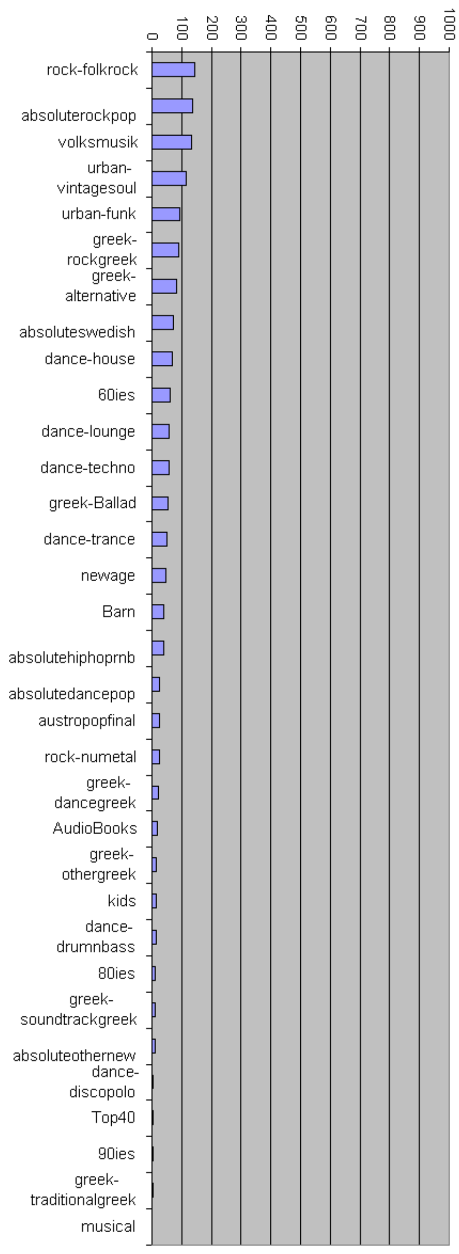


Figure C.2.: Track – Genre Distribution in Europe (2nd part)

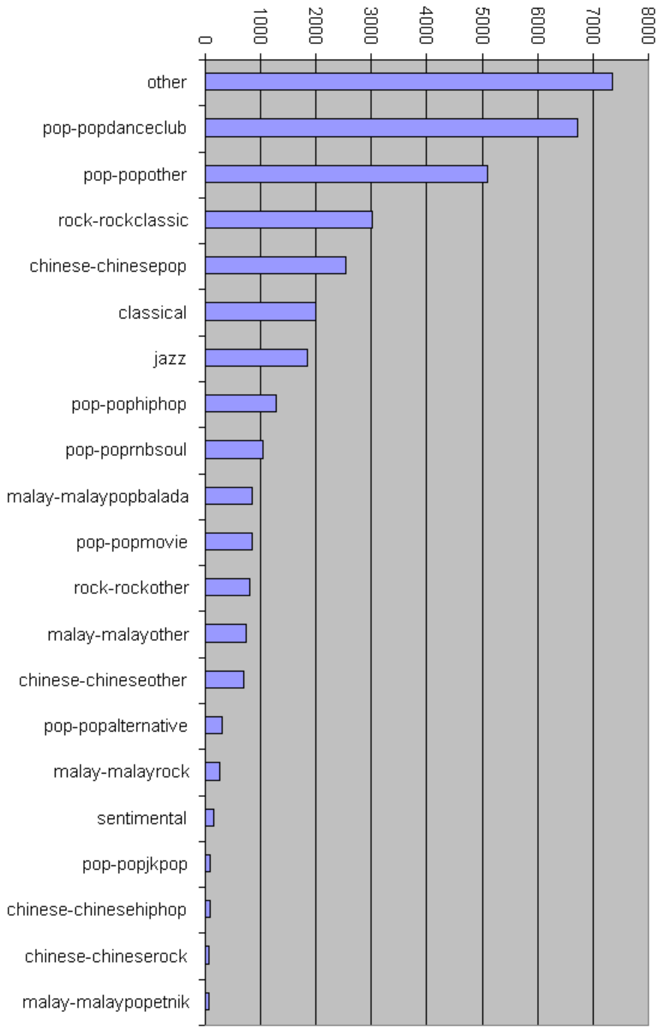


Figure C.3.: Track – Genre Distribution in Malaysia (1st part)

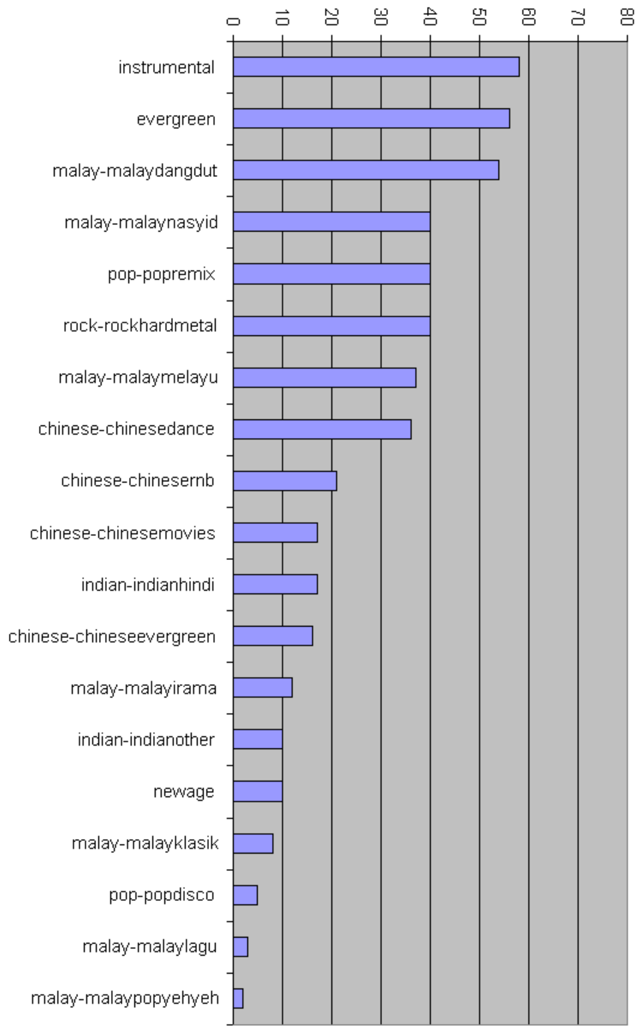


Figure C.4.: Track – Genre Distribution in Malaysia (2nd part)

Bibliography

Adomavicius, G., Sankaranarayanan, R., Sen, S., and Tuzhilin, A. (2005). Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Transactions on Information Systems*, 23(1):103–145.

Adomavicius, M.-G. and Tuzhilin, M.-A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engeneering*, 17(6):734–749. Member-Gediminas Adomavicius and Member-Alexander Tuzhilin.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules. In Bocca, J. B., Jarke, M., and Zaniolo, C., editors, *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB*, pages 487–499. Morgan Kaufmann.

Alani, H., O'Hara, K., and Shadbolt, N. (2002). Ontocopi: Methods and tools for identifying communities of practice. In *Proceedings of the 2002 IFIP World Computer Congress*, pages 225–236.

Allgayer, J., Harbusch, K., Kobsa, A., Reddig, C., and Reithinger, N. (1989). Xtra: a natural-language access system to expert systems. *International Journal of Man-Machine Studies*, 31(2):161–195.

Aucouturier, J.-J. and Pachet, F. (2002). Music similarity measures: What's the use? In *Proceedings of the Third International Symposium on Music Information Retrieval (ISMIR02)*, pages 157–163, Paris, France.

Aucouturier, J.-J. and Pachet, F. (2004). Improving timbre similarity: How high is the sky? *Journal of Negative Results in Speech and Audio Sciences*, 1(1).

Balabanović, M. and Shoham, Y. (1997). Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72.

Baraglia, R., Lucchese, C., Orlando, S., Serrano', M., and Silvestri, F. (2006). A privacy preserving web recommender system. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 559–563, New York, NY, USA. ACM.

Belkin, N. J. and Croft, W. B. (1992). Information filtering and information retrieval: two sides of the same coin? *Communications of the ACM*, 35(12):29–38.

Bell, R. M. and Koren, Y. (2007). Lessons from the netflix prize challenge. *SIGKDD Explorations Newsletter*, 9(2):75–79.

Bennett, J. and Lanning, S. (2007). The netflix prize. In *Proceedings of KDD Cup and Workshop*, volume 2007.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web a new form of web content that is meaningful to computers will unleash a revolution of new possibilities. Scientific America. http://www.sciam.com/article.cfm?id=the-semantic-web (as of 04/08/2009).

Billsus, D. and Pazzani, M. J. (1998). Learning collaborative information filters. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 46–54, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Billsus, D. and Pazzani, M. J. (2000). User modeling for adaptive news access. *User Modeling and User-Adapted Interaction*, 10(2-3):147–180.

Bouza, A., Reif, G., Bernstein, A., and Gall, H. (2008). Semtree: Ontology-based decision tree algorithm for recommender systems. In Bizer, C. and Joshi, A., editors, *International Semantic Web Conference (Posters & Demos)*, volume 401 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Brajnik, G. and Tasso, C. (1994). A shell for developing non-monotonic user modeling systems. *Int. J. Hum.-Comput. Stud.*, 40(1):31–62.

Breese, J. S., Heckerman, D., and Kadie, C. (1998). Empirical analysis of predictive algorithms for collaborative filtering. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52. Morgan Kaufmann.

Brown, P. J., Bovey, J. D., and Chen, X. (1997). Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE [see also IEEE Wireless Communications]*, 4(5):58–64.

Brusilovsky, P. and Maybury, M. T. (2002). From adaptive hypermedia to the adaptive web. *Communications of the ACM*, 45(5):30–33.

Buriano, L., Marchetti, M., Carmagnola, F., Cena, F., Gena, C., and Torre, I. (2006). The role of ontologies in context-aware recommender systems. In *MDM '06: Proceedings of the 7th International Conference on Mobile Data Management*, page 80. IEEE Computer Society.

Burke, R. (1999). Integrating knowledge-based and collaborative-filtering recommender systems. In *AAAI Workshop on AI in Electronic Commerce*, pages 69–72. AAAI.

Burke, R. (2000). Knowledge-based recommender systems. In *Encyclopedia of Library and Information Systems*, volume 69, pages 180–200. Marcel Dekker.

Burke, R. (2002). Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370.

Burke, R. D., Hammond, K. J., and Young, B. C. (1997). The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40.

Carroll, J. M. and Rosson, M. B. (1987). Paradox of the active user. In *Interfacing thought: cognitive aspects of human-computer interaction*, pages 80–111. MIT Press, Cambridge, MA, USA.

Cerny, R. (2008). Design and implementation of a generic recommender and its application to the music domain. Master's thesis, Vienna University of Technology, Vienna, Austria.

Comer, D. (1979). Ubiquitous b-tree. *ACM Computing Surveys*, 11(2):121–137.

Deshpande, M. and Karypis, G. (2004). Item-Based Top-N Recommendation Algorithms. *ACM Transactions on Information Systems*, 22(1):143–177.

Dey, A. K. and Abowd, G. D. (2000). Towards a Better Understanding of Context and Context-Awareness. *CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness*.

Edwards, W. (1977). How to use multi-attribute utility measurement for social decision-making. *IEEE Transactions on Systems, Man and Cybernetics*, 7(5):326–340.

Edwards, W. (1994). Smarts and smarter: Improved simple methods for multiattribute utility measurement. *Organizational Behavior and Human Decision Processes*, 60(3):306–325.

Felfernig, A. and Burke, R. (2008). Constraint-based recommender systems: technologies and research issues. In *ICEC '08: Proceedings of the 10th international conference on Electronic commerce*, pages 1–10, New York, NY, USA. ACM.

Finin, T. and Drager, D. (1986). Gums: a general user modeling system. In *HLT '86: Proceedings of the workshop on Strategic computing natural* *language*, pages 224–230, Morristown, NJ, USA. Association for Computational Linguistics.

Ghosh, J. and Nag, A. (2001). An overview of radial basis function networks. *Radial basis function networks 2: new advances in design*, pages 1–36.

Goldberg, D., Nichols, D., Oki, B. M., and Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70.

Gruber, T. R. (1993). A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220.

Gstrein, E. (2003). The bpnd-tree: An efficient search structure for high dimensional profile based recommendation systems. Technical report, Austrian Research Centers.

Gstrein, E. et al. (2005). Adaptive personalization: A multi-dimensional approach to boosting a large scale mobile music portal. In *Fifth Open Workshop on MUSICNETWORK: Integration of Music in Multimedia Applications*, pages 1–8, Vienna, Austria.

Gstrein, E. and Krenn, B. (2006). Mobile personalization at work. In *Proceedings of the ECAI 2006 Workshop on Recommender Systems*, pages 122–124, Riva del Garda, Italy.

Guttman, R. H. (1998). Merchant differentiation through integrative negotiation in agent-mediated electronic commerce. Master's thesis, Media Arts and Sciences School of Architecture and Planning, Massachusetts Institute of Technology.

Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Computer Supported Cooperative Work*, pages 241–250.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., John, and Riedl, T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, 22:5–53.

Heyer, G., Quasthoff, U., and Wittig, T. (2006). *Text Mining: Wissensrohstoff Text*. W3L GmbH, ISBN 978-3-937137-30-8.

Hill, W., Stead, L., Rosenstein, M., and Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Huang, S.-L. (2008). Comparison of utility-based recommendation methods. *12th Pacific Asia Conference on Information Systems; July 3-7; Suzhou RPC*.

Jameson, A., Hoeppner, W., and Wahlster, W. (1980). The natural language system ham-rpm as a hotel manager: Some representational prerequisites. In *GI - 10. Jahrestagung*, pages 459–473, London, UK. Springer-Verlag.

John, B. M., Riedl, J. T., and Konstan, J. A. (1997). Experiences with grouplens: Making usenet useful again. In *Proceedings of the 1997 Usenix Winter Technical Conference*, pages 219–231.

Kay, J. (1990). Um: A toolkit for user modelling. In *Second International Workshop on User. Modeling.*, Honolulu, HI. 1-11.

Kim, S. and Kwon, J. (2007). Effective context-aware recommendation on the semantic web. *IJCSNS International Journal of Computer Science and Network Security*, 7:154–159.

Kleedorfer, F. (2008). Automatic topic detection in song lyrics. Master's thesis, TU Vienna.

Kobsa, A. (1990). Modeling the user's conceptual knowledge in bgp-ms, a user modeling shell system. *Computational Intelligence*, 6(4):193–208.

Kobsa, A. (1995). Editorial. *User Modeling and User-Adapted Interaction*, 4(2):iii–v. DOI 10.1007/BF01099427.

Kobsa, A. (2001). Generic user modeling systems. *User Modeling and User-Adapted Interaction*, 11(1-2):49–63.

Kobsa, A. and Fink, J. (2006). An ldap-based user modeling server and its evaluation. *User Modeling and User-Adapted Interaction*, 16(2):129–169.

Kobsa, A. and Pohl, W. (1995). The user modeling shell system bgp-ms. In *User Modelling and User-adapted Interaction*, pages 59–106.

Kobsa, A. and Wahlster, W. (1989). *User models in dialog systems*. Springer-Verlag New York, Inc., New York, NY, USA.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., and Riedl, J. (1997). Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87.

Koren, Y. (2009). Collaborative filtering with temporal dynamics. In *KDD '09: Proceedings of the 15th ACM SIGKDD international conference on*

Knowledge discovery and data mining, pages 447–456, New York, NY, USA. ACM.

Koren, Y., Bell, R., and Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

Krenn, B. and Gstrein, E. (2006). On female and male avatars: Data from a web-based flirting community. *Proceedings of the AVI 2006 Workshop on Gender and Interaction. Real and virtual women in a male world*.

Krenn, B., Gstrein, E., Neumayr, B., and Grice, M. (2002). What can we learn from users of avatars in net environments? In *Proceedings of AA-MAS 2002 Workshop:Embodied Conversational Agents - Let's Specify and Evaluate Them!, July 15-16*, Bologna, Italy.

Krenn, B., Neumayr, B., Gstrein, E., and Grice, M. (2004). Life-like agents for the internet: A cross-cultural case study. In Payr, s. and Trappl, R., editors, *Agent Culture: Human-Agent Interaction in a Multicultural World*, pages 197–229. Lawrence Erlbaum Associates.

Krenn, B., Skowron, M., Sieber, G., Gstrein, E., and Irran, J. (2009). Adaptive mind agent. *9th International Conference on Intelligent Virtual Agents , Amsterdam 14-16 September 09*.

Krulwich, B. (1997). Lifestyle finder: Intelligent user profiling using large–scale demographic data. *Artificial Intelligence Magazine*, 18(2):37–45.

Lam, S. K. and Riedl, J. (2004). Shilling recommender systems for fun and profit. In *Proceedings of the 13th international conference on World Wide Web*, pages 393–402. ACM Press.

Lemire, D., Boley, H., McGrath, S., and Ball, M. (2005). Collaborative filtering and inference rules for context-aware learning object recommendation. *International Journal of Interactive Technology and Smart Education*, 2(3):179 –188.

Lemire, D. and Maclachlan, A. (2005). Slope one predictors for online rating-based collaborative filtering. In *Proceedings of SIAM Data Mining (SDM'05)*, pages 471–475.

Limbu, D. K., Connor, A., Pears, R., and MacDonell, S. (2006). Contextual relevance feedback in web information retrieval. In *IIiX: Proceedings of the 1st international conference on Information interaction in context*, pages 138–143, New York, NY, USA. ACM.

Linden, G., Smith, B., and York, J. (2003). Amazon.com recommendations: item-to-item collaborative filtering. *Internet Computing, IEEE*, 7(1):76–80.

Littlestone, N. (1987). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *SFCS ’87: Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, pages 68–77, Washington, DC, USA. IEEE Computer Society.

Lorenzi, F. (2007). A multiagent knowledge-based recommender approach with truth maintenance. In *RecSys ’07: Proceedings of the 2007 ACM conference on Recommender systems*, pages 195–198, New York, NY, USA. ACM.

Luft, J. and Ingham, H. (1955). The johari window, a graphic model of interpersonal awareness. In *Proceedings of the western training laboratory in group development*, Los Angeles: UCLA.

Massa, P. and Avesani, P. (2004). Trust-aware collaborative filtering for recommender systems. In *Proceedings of Federated Int. Conference On The Move to Meaningful Internet: CoopIS, DOA, ODBASE*, pages 492–508.

McSherry, F. and Mironov, I. (2009). Differentially private recommender systems: Building privacy into the Netflix Prize contenders. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining—KDD 2009*. ACM.

Mehrabian, A. (1996). Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292.

Middleton, S. E., Alani, H., Shadbolt, N., and Roure, D. D. (2002). Exploiting synergy between ontologies and recommender systems. In Frank, M., Noy, N. F., and Staab, S., editors, *Semantic Web Workshop*, volume 55 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Middleton, S. E., De Roure, D. C., and Shadbolt, N. R. (2001). Capturing knowledge of user preferences: ontologies in recommender systems. In *K-CAP ’01: Proceedings of the 1st international conference on Knowledge capture*, pages 100–107, New York, NY, USA. ACM.

Middleton, S. E., Shadbolt, N. R., and Roure, D. C. D. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems*, 22(1):54–88.

Myers, I. B. and Myers, P. B. (1995). *Introduction to Type: A Guide to Understanding Your Results on the Myers-Briggs Type Indicator*. Davies-Black Publishing. Original edition 1980; Reprint edition 1995.

Narayanan, A. and Shmatikov, V. (2008). Robust de-anonymization of large sparse datasets. In *SP '08: Proceedings of the 2008 IEEE Symposium on Security and Privacy*, pages 111–125, Washington, DC, USA. IEEE Computer Society.

Niederée, C., Stewart, A., Mehta, B., and Hemmje, M. (2004). A multi-dimensional, unified user model for cross-system personalization. In *Proceedings of Workshop On Environments For Personalized Information Access at Advanced Visual Interfaces*.

O'Donovan, J. and Smyth, B. (2005). Trust in recommender systems. In *IUI '05: Proceedings of the 10th international conference on Intelligent user interfaces*, pages 167–174, New York, NY, USA. ACM.

Orwant, J. (1994). Heterogeneous learning in the doppelgänger user modeling system. *User Modeling and User-Adapted Interaction*, 4(2):107–130.

Paiva, A. and Self, J. A. (1995). Tagus - a user and leamer modeling workbench. *User Modeling and User-Adapted Interaction*, 4(3):197–226.

Pampalk, E., Dixon, S., and Widmer, G. (2003). On the evaluation of perceptual similarity measures for music. In *Proceedings of the International Conference on Digital Audio Effects (DAFx-03)*, pages 7–12.

Park, J. S., Chen, M.-S., and Yu, P. S. (1995). An effective hash based algorithm for mining association rules. In Carey, M. J. and Schneider, D. A., editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 175–186, San Jose, California.

Pascoe, J., Ryan, N. S., and Morse, D. R. (1998). Human Computer Giraffe Interaction: HCI in the Field. In Johnson, C., editor, *Workshop on Human Computer Interaction with Mobile Devices*, GIST Technical Report G98-1. University of Glasgow.

Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. *Artificial Intelligence Review*, 13(5-6):393–408.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In Brusilovsky, P., Kobsa, A., and Nejdl, W., editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 325–341, Berlin, Heidelberg. Springer.

Pennock, D., Horvitz, E., Lawrence, S., and Giles, C. L. (2000). Collaborative filtering by personality diagnosis: A hybrid memory- and model-based approach. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 473–480. Morgan Kaufmann.

Perrault, C. R., Allen, J. F., and Cohen, P. R. (1978). Speech acts as a basis for understanding dialogue coherence. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, pages 125–132, Morristown, NJ, USA. Association for Computational Linguistics.

Potter, G. (2008). Putting the collaborator back into collaborative filtering. In *Proceedings of the Second KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*, pages 9–12, Las Vegas, Nevada, USA.

Procopiuc, O., Agarwal, P. K., Arge, L., and Vitter, J. S. (2002). Bkd-tree: A dynamic scalable kd-tree. In *Proceedings of the International Symposium on Spatial and Temporal Databases*, pages 46–65.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, 1(1):81–106.

Rack, C., Arbanowski, S., and Steglich, S. (2007). A generic multipurpose recommender system for contextual recommendations. In *ISADS '07: Proceedings of the Eighth International Symposium on Autonomous Decentralized Systems*, pages 445–450, Washington, DC, USA. IEEE Computer Society.

Ramakrishnan, N., Keller, B. J., Mirza, B. J., Grama, A. Y., and Karypis, G. (2001). Privacy risks in recommender systems. *IEEE Internet Computing*, 5(6):54–62.

Rentfrow, P. J. and Gosling, S. D. (2003). The do re mi's of everyday life: The structure and personality correlates of music preferences. *Journal of Personality and Social Psychology*, 84(6):1236–1256.

Resnick, P., Iacovou, N., Suchak, M., Bergstorm, P., and Riedl, J. (1994). GroupLens: An Open Architecture for Collaborative Filtering of Netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina. ACM.

Resnick, P. and Varian, H. R. (1997). Recommender systems. *Communications of the ACM*, 40(3):56–58.

Rich, E. (1979). User modeling via stereotypes. *Cognitive Science*, 3:329–354.

Rich, E. (1983). Users are individuals: individualizing user models. *International Journal of Man-Machine Studies*, 18:199–214.

Riedl, J. and Dourish, P. (2005). Introduction to the special section on recommender systems. In *ACM Transactions on Computer-Human Interactions*, pages 371–373.

Robinson, J. T. (1981). The k-d-b-tree: a search structure for large multidimensional dynamic indexes. In *SIGMOD '81: Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pages 10–18, New York, NY, USA. ACM.

Rocchio, J. (1971). Relevance feedback in information retrieval. In Salton, G., editor, *The SMART Retrieval System*, pages 313–323. Prentice Hall.

Samet, H. (2005). *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Sarwar, B., Karypis, G., Konstan, J., and Reidl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 285–295, New York, NY, USA. ACM.

Schafer, J. B., Konstan, J., and Riedi, J. (1999). Recommender systems in e-commerce. In *EC '99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, New York, NY, USA. ACM.

Schein, A. I., Popescul, A., Ungar, L. H., and Pennock, D. M. (2002). Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA. ACM.

Schäfer, R. (2001). Rules for using multi-attribute utility theory for estimating a user's interests. In *Workshop on Adaptivity and User Modelling*, Dortmund, Germany. Cambridge University Press.

Schmidt, A., Aidoo, K. A., Takaluoma, A., Tuomela, U., Van Laerhoven, K., and Van de Velde, W. (1999). Advanced interaction in context. *Lecture Notes in Computer Science*, 1707:89–101.

Schnabel, S. (2007). Investigating web usage data. Master's thesis, TU Vienna.

Shardanand, U. and Maes, P. (1995). Social information filtering: Algorithms for automating "word of mouth". In *Proceedings of ACM CHI'95 Conference on Human Factors in Computing Systems*, volume 1, pages 210–217.

Shortliffe, E. (1974). *MYCIN: A rule-based computer program for advising physcians regarding antimicrobial therapy selection*. PhD thesis, Stanford University.

Shortliffe, E. (1976). *Computer-Based Medical Consultations: MYCIN*. Artificial intelligence series ; 2 Elsevier computer science library. Elsevier, New York.

Steglich, S., Räck, C., and Arbanowski, S. (2005). Amaya: A recommender system for ambient-aware recommendations. In *International Conference on Internet Computing*, pages 389–394.

Swearingen, K. and Sinha, R. (2001). Beyond algorithms: An hci perspective on recommender systems. In *ACM SIGIR. Workshop on Recommender Systems*, volume Vol. 13, Numbers 5-6, pages 393–408.

Swearingen, K. and Sinha, R. (2002). Interaction design for recommender systems. In *Proceedings of the Conference on Designing Interactive Systems (DIS'02)*, London, England.

Terveen, L. and Hill, W. (2001). Human-computer collaboration in recommender systems. In Wesley, A., editor, *HCI on the new Millennium*, pages 223–242. J. Carroll.

Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J. (1997). Phoaks: a system for sharing recommendations. *Communications of the ACM*, 40(3):59–62.

Thomas, F. B. (1992). How to interpret yourself/johari window. In *SIGUCCS '92: Proceedings of the 20th annual ACM SIGUCCS conference on User services*, pages 225–230, New York, NY, USA. ACM.

Towle, B. and Quinn, C. (2000). Knowledge based recommender systems using explicit user models. In *Papers from the AAAI Workshop, AAAI Technical Report WS-00-04*, pages 74–77. Menlo Park, CA: AAAI Press.

Tupes, E. C. and Christal, R. E. (1992). Recurrent personality factors based on trait ratings. *Journal of Personality*, 60(2):225–251.

Uitdenbogerd, A. and van Schnydel, R. (2002). A review of factors affecting music recommender success. In *Proceedings of 3rd International Conference on Music Information Retrieval*, Paris, France.

Ujjin, S. and Bentley, P. J. (2001). Building a lifestyle recommender system. In *Poster Proceedings of the Tenth International World Wide Web Conference, WWW 10*, Hong Kong, China.

von Hahn, W., Wahlster, W., and Hoeppner, W. (1982). Ham-ans (hamburg application-oriented natural-language system): U. of hamburg, frg. *SIGART Bulletin*, (79):57–57.

W3C (2004a). Composite capability/preference profiles (cc/pp): Structure and vocabularies 1.0. http://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/. (04/08/2009).

W3C (2004b). Rdf primer. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. (04/08/2009).

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition (Morgan Kaufmann Series in Data Management Systems)*. Morgan Kaufmann.

D. Curriculum Vitae

Address
Erich Gstrein
Lorenz Mandlgasse 46/9
1160 Vienna, Austria
gstrein@acm.org

Date and Place of Birth
Mai 16, 1963; Lienz, Osttirol

Education
1982-1989
M.S. studies in Information Science at the Vienna University of Technology

2008-2009
Ph.D. studies in Information Science at the Vienna University of Technology

Positions Held
2003-today
Director of the Smart Agent Technologies studio of the Research Studios Austria Forschungsgesellschaft mbH.

2006-2008
Manager Research at Verisign Communications GmbH. Responsible for the development of personalization solutions and the coordination of corresponding R&D projects.

2004-2006
Head of the Personalization & Data mining group at Three United mobile solutions ag.

2000-2004
Head of the artificial intelligence group at SYSIS interactive simulations ag. Development of *NetLife* an artificial life platform for simulating need driven agents. Project manager of several commercial and research projects. General technical manager and quality officer of NECA, an research project founded by the European Union.

1999
Trainer for Object Oriented Software Development and C++ at Integrata Österreich GmbH

1993-2000
Chief architect and the head of the implementation team of a large mission critical client-server project for the Austrian saving banks at SPARDAT.

1993
Senior architect at AT&T, responsible for defining and implementing an object oriented software developing process.

1989-1993
Software architect of N/Joy at Vienna Software Publishing, an award winning office automation system for OS/2. Best International Product at Comdex 1992.

Publications

Krenn, B., Gstrein, E., Goetzloff, I.: Vitalmonitoring für ein langes Leben in den eigenen vier Wänden. (Accepted at) 3. Deutscher AAL-Kongress 2010, Berlin 26-27 Janurary, 2010.

Krenn, B., Skowron, M., Sieber, G., Gstrein, E., Irran, J.: Adaptive Mind Agent. 9th International Conference on Intelligent Virtual Agents, Amsterdam 14-16 September, 2009.

Gstrein, E., Krenn B.: Mobile Personalization at Work. Workshop on Recommender Systems, ECAI 06, Riva del Garda, 18-19 August , 2006.

Gstrein, E., Kleedorfer, F., Krenn, B.: Automated Meta Data Generation for Personalized Music Portals. Technical Report, Austrian Research Centers, 2006.

Hlavac, P. , Krenn B, Gstrein, E.: SOUNDSCOUT: A SONG RECOMMENDER BASED ON SOUND SIMILARITY FOR HUGE COMMERCIAL MUSIC ARCHIVES. Technical Report, Austrian Research Centers, 2006.

Krenn, B., Gstrein, E.: On Female and Male Avatars: Data from a Web-Based Flirting Community. In Proceedings of the AVI 2006 Workshop on Gender and Interaction. Real and virtual women in a male world, 2006.

Gstrein, E., Kleedorfer, F., Mayer R., Schmotzer, C., Widmer G., Holle O. and Miksch S. (2005). Adaptive Personalization: A Multi-Dimensional Approach to Boosting a Large Scale Mobile Music Portal. In Fifth Open Workshop on MUSICNETWORK: Integration of Music in Multimedia Applications, Vienna, Austria

Roemmer-Nossek, B., Peschl, M.F., Gstrein, E., Oswald, M.: Knowledge Elicitation and Navigation in Knowledge Spaces; Knowtech 04, Oct. 2004

Oswald, M., Roemmer-Nossek, B., Gstrein, E., Peschl, M.F.: Enhancing Blogs with a Dual Interaction Design; BlogTalk 2.0; 5-6. July 2004

Krenn, B., Gstrein, E., Grice, M.: Lifelike Agents for the Internet: A Cross-Cultural Case Study. In Payr S., Trappl R. (eds). Agent Culture: Human-Agent Interaction in a Multicultural World. Lawrence Erlbaum Associates, New Jersey, 2004.

Gstrein, E.: The BPnD-Tree: An efficient search structure for high dimensional profile based recommendation systems. Technical Report, Researchstudios Austria , 2003

Krenn, B., Grice, M., Piwek, P., Schröder, M., Klesen, M., Baumann, S., Pirker, H., van Deemter,K., Gstrein, E.: Generation of Multi-modal Dialogue for Net Environments. Proceedings of KONVENS-02, 30 September - 2 October 2002, Saarbrücken, Germany.

Krenn, B., Gstrein, E., Neumayr, B., Grice, M.: What can we learn from users of avatars in net environments? Proceedings of the Workshop "Embodied conversational agents - let's specify and evaluate them!," held in conjunction with AAMAS-02, July 16 2002, Bologna, Italy.

Gstrein, E., Nejdl, W.: The follow-graph: A deterministic finite automaton for solving recursive queries. Technical report, Technische Universität Wien, January 1989.

Awards
easyrec: Multimedia & e-business Staatspreis 2009; Jury Auszeichnung Innovationspreis.

Manata Island: Finalist – Van Dusseldorp Award for Excellence in Digital Media 2003.

Xmas Agent: Staatspreis Multimedia 2001.

N/JOY: Best International Product; Comdex 1992.