UNIVERSITÄT
DUISBURG
ESSEN

Cross System Personalization: Enabling personalization across multiple systems

Von der Fakultät für Ingenieurwissenschaften
der Universität Duisburg-Essen
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften
genehmigte Dissertation

von

Bhaskar Mehta
M.Tech., B.Tech.

aus Faridabad, Indien

Referent: Prof. Dr. Norbert Fuhr
Korreferent: Dr. Thomas Hofmann

Tag der mündlichen Prüfung: 7. February 2008

Dedication

This thesis is dedicated to my parents and my wife Divu, without whose love and encouragement, none of this would have been possible.

Acknowledgment

"In time of test, family is best."
— Burmese Proverb

I began this journey of knowledge many years ago, when as an inquisitive child, my father introduced me to science and mathematics. Reading all the books he had collected, made me want to know it all; it was an exercise where I miserably failed. However, the desire to learn more remained, like a fire buring within, only which reading and knowing more could quench. For that, I shall be eternally thankful to my father.

They say, to achieve something, *You have to believe in yourself, that's the secret.*. This was a lesson my mother taught me, always believing in me, in my talents, and always standing by me. For instilling self-belief and confidence in me, I am grateful to my mother.

For protecting me from all negative influences, their belief in me, and unconditional support, I am thankful to my sisters.

To my wife, who waited patiently as I came home late day after day, and for making it so much easier than it would have been otherwise, I am full of gratitude. For being my friend, my companion, and my well of support, I owe a lot to you.

I also thank all the colleagues and friends who helped me in this journey. I am thankful to Claudia (Niederee) who first mentored me and gave me direction. I am thankful to the colleagues who were supportive: Claudio Muscoguiri, Avare Stewart, Predrag Knezivic, Thomas Risse, Martin Leissler, Gerald Jaschke, who patiently listened to me and gave their frank opinions.

I am especially thankful to Peter Fankhauser, who mentored me in aspects beyond research: his friendship and support are greatly cherished and helped me significantly in this journey. His willingness to discuss all topics, and the ability to *cut out the noise* and find the real issues, are skills I hope to acquire some day. I learnt a lot from Peter, and continue to do so.

Importantly, I am indebted to Thomas Hofmann, who's interest in my research changed the direction my life has taken. I feel honored to have worked with a man as bright as him, and was able to learn so many things by just being around. Your gentle support, and willingness to think about my ideas, greatly helped me to produce work of significantly higher quality than I would have otherwise. I hope to live to your high standards for all my research career. Thanks for being the best guide and mentor I could wish for, for being my *Dronacharya*.

I am also very grateful to Ingo Frommholz, and Eelco Herder for patiently reading early versions of my thesis and providing me comments. And finally, I feel fortunate to receive guidance from Prof Norbert Fuhr, and for his generous acceptance of my candidature as a PhD student. I am very thankful to him for his support in this process and his gentle advice on various aspects fundamental to a PhD defense.

Abstract

The World Wide Web provides access to a wealth of information and services to a huge and heterogeneous user population on a global scale. One important and successful design mechanism in dealing with this diversity of users is to *personalize* Web sites and services, i.e. to customize system content, characteristics, or appearance with respect to a specific user. Each system independently builds up user profiles and uses this information to personalize the service offering. Such isolated approaches have two major drawbacks: firstly, investments of users in personalizing a system either through explicit provision of information or through long and regular use are not transferable to other systems. Secondly, users have little or no control over the information that defines their profile, since user data are deeply buried in personalization engines running on the server side.

Cross system personalization (CSP) (Mehta, Niederee, & Stewart, 2005) allows for sharing information across different information systems in a user-centric way and can overcome the aforementioned problems. Information about users, which is originally scattered across multiple systems, is combined to obtain maximum leverage and reuse of information. Our initial approaches to cross system personalization relied on each user having a *unified profile* which different systems can understand. The unified profile contains facets modeling aspects of a multidimensional user which is stored inside a "*Context Passport*" that the user carries along in his/her journey across information space. The user's *Context Passport* is presented to a system, which can then understand the context in which the user wants to use the system. The basis of 'understanding' in this approach is of a semantic nature, i.e. the semantics of the facets and dimensions of the unified profile are known, so that the latter can be aligned with the profiles maintained internally at a specific site. The results of the personalization process are then transfered back to the user's Context Passport via a protocol understood by both parties. The main challenge in this approach is to establish some common and globally accepted vocabulary and to create a standard every system will comply with.

Machine Learning techniques provide an alternative approach to enable CSP without the need of accepted semantic standards or ontologies. The key idea is that one can try to learn dependencies between profiles maintained within one system and profiles maintained within a second system based on data provided by users who use both systems and who are willing to share their profiles across systems – which we assume is in the interest of the user. Here, instead of requiring a common semantic framework, it is only required that a sufficient number of users cross between systems and that there is enough regularity among users that one can learn within a user population, a fact that is commonly exploited in *collaborative filtering*.

In this thesis, we aim to provide a principled approach towards achieving cross system personalization. We describe both *semantic* and *learning* approaches, with a stronger emphasis on the learning approach. We also investigate the privacy and scalability aspects of CSP and provide solutions to these problems. Finally, we also explore in detail the aspect of robustness in recommender systems. We motivate several approaches for robustifying collaborative filtering and provide the best performing algorithm for detecting malicious attacks reported so far.

Contents

1 Introduction

Every great advance in science has issued from a new audacity of the imagination.

(John Dewey)

Across the world, 24 hours a day, friends and families chat, exchange letters and pictures - all via electronic communications; business negotiates multi-million-dollar deals; products and services are bought and sold; banks process millions of financial transactions; travel agents organize business and holiday trips, and students research assignments. Increasingly, people are using the web for retrieving information instead of conventional sources of like books, magazines and libraries. Phone directories, newspapers and shopping stores are being increasingly replaced with electronic versions. Search engines like *Google* are used by millions of people to search for information that helps them in their work and day-to-day life. The *World Wide Web* (WWW) has become a very important source of information and communication. From the users' point of view, obtaining the 'right' information, which is needed to solve a problem or accomplish a task carries tremendous value. One important and successful design mechanism in dealing with this requirement from a diverse set of users is to *personalize* Web sites and services, i.e˙ to customize system contents, characteristics, or appearance with respect to a specific user. The ultimate goal is to optimize access to relevant information or products by tailoring search results, displays, etc. to a user's presumed interests and preferences. More specifically, this optimization may aim at increasing the efficiency of system usage or improving the quality and relevance of results. Given the huge and rapidly growing amount of data available online as well as an ever growing user population that uses the WWW, the relevance of personalized access has reached a critical point and is likely to further increase in the future.

Personalization today has wide spread use on many eCommerce sites. Applications store preferences and other information about users in order to provide personalized access. Lately, web stores like Amazon have started using recommender systems extensively, which additionally profile information about user interests and skills, typically implicitly by observing and analyzing user behavior. Each system independently builds up information about a user's likes and dislikes, interests, and further characteristics and uses this information to personalize the system's content and service offer (Riecken, 2000; Neuhold, Niederée, & Stewart, 2003). There are various personalization techniques (Neuhold et al., 2003; Pretschner & Gauch, 1999); most of these rely on either the implicit collection of information about users by tracking their system usage behavior or the users putting in effort to explicitly providing information about themselves or giving feedback to the system. Such techniques often need careful investment from the user's point of view, as the end system analyzes the collected information and learns more about the user in time; this is called as the *training phase*. When a user uses multiple electronic systems

which offer personalization, the user has to go through similar *training* phases every individual system often providing the same, or similar information. From the user's point of view, there are several drawbacks with such isolated personalization approaches:

- Investments of users in personalizing a system either through explicit provision of information or just through long and regular use are not transferable to other systems. However, complex tasks like booking a travel or preparing a proposal require people to obtain information from multiple sources, and to switch between different information systems. In such a scenario, users would clearly benefit from transferring personalization information between systems.
- Users have little or no control over the information that defines their *profile*, since user profiles are deeply buried in personalization engines.
- Given the current trend towards *Service-oriented Architectures* and *P2P technology*, electronic systems are making a transition from centrally controlled systems to dynamic federations of service and content resources. Services which dynamically join such a federation-based system can benefit greatly from the personalization information already built up and hence immediately provide personalized access.

Cross System Personalization (CSP), i.e. personalization that shares personalization information across different system in a user-centric way, overcomes the aforementioned problems. Information about users that is scattered across multiple systems is combined to obtain maximum leverage. This enables new users to immediately experience a level of personalization which is usually possibly only after a long interaction. Moreover, existing users can also benefit greatly by reusing their profile data present in other electronic systems to experience more effective personalization. The development of an approach for CSP is the core problem addressed in this thesis.

1.1 Problem Statement

The objective of this thesis is *'to enable sharing and combining of user profile information spread across multiple electronic systems to provide an enhanced personalization experience for end users, and provide control to the end users over their profiles'.*

1.2 Contributions

Cross-System Personalization (CSP) is a new direction in personalization, which explores the effectiveness of reusing user profile information spread across multiple systems. The first contribution of this thesis is in formulating the CSP problem and demonstrating that Cross System Personalization indeed benefits end users. Further, this thesis provides a thorough analysis of the solution space, developing both knowledge based, and knowledge-poor solutions. In the *semantic approach* to CSP, we propose a unified user context model(UUCM), which models the unified profile of a multitasking user using multiple electronic systems. Further, we develop a protocol called CSCP which can be used by two parties to exchange information about a user's profile.

The *learning solution* to CSP builds on the observations made during the design of the semantic solution: specifically, deploying semantic solutions using CSCP requires that multiple electronic systems agree on a common vocabulary which is mutually understood. This means a

significant effort has to be made in the direction of standardization. The learning solution aims to make as few assumptions as possible, relying instead on the fact that user profiles from many users using multiple systems can be used to learn a mapping between use profile formats. Using this key idea, we use techniques in machine learning to enable CSP.

The contribution in the learning solution to CSP is to suitably modify existing statistical and graph learning methods to deal with the peculiarities of the CSP task: namely *sparsity*, *using data correspondence* and *privacy preservation*. We use dimensionality reduction methods (both linear and non linear) in a novel way, suitably extending them as required by our problem. We also use PLSA, a well known Latent Semantic Model technique, and apply it successfully to the CSP task. We also suitably enhance PLSA to add distributivity and privacy, and describe a protocol to deploy PLSA for CSP in a peer to peer setting. We also validate our hypothesis, and provide experimental proof that CSP results in a significant measurable improvement in personalization.

Finally we also explore the *robustness* of collaborative filtering methods towards malicious attacks, and propose unsupervised learning techniques for *detecting spam* profiles inserted into the user database. The PCA based spam detection method proposed in this thesis is a novel usage of PCA and provides the most accurate detection of spam user profiles so far. We also provide detailed experimental results on the performance of our proposed methods and compare it with existing methods to report a significant improvement.

Structure of Work

In this thesis, we look in depth into the issues and challenges that arise in achieving Cross System Personalization, discussing scenarios and suggesting approaches that can be used to achieve this objective in information systems. After discussing related work in Chapter 2, we describe the conceptual approach and the underlying Machine Learning and Semantic methods in Chapter 3. Chapter 4 discusses the evaluation and experimentation. Finally conclusions are drawn in Chapter 5.

2 State of the Art and Related Work

> There is nothing like looking, if you want to find something. You certainly usually find something, if you look, but it is not always quite the something you were after.
>
> (J.R.R. Tolkien)

The problem discussed in this thesis lies in the area of personalization, drawing on related work in a number of sub-areas: *Recommender Systems*, *User Modeling*, *Collaborative Filtering* and *Semantic Web Personalization*. In addition, the *Machine Learning* approach to Cross System Personalization uses techniques in the area of *Dimensionality Reduction*. Therefore, we describe the techniques used in Chapter 3 also as a part of related work.

The organization of this chapter is as follows: first we describe concepts like Personalization and Recommender systems; next, User Modeling is discussed in detail, followed by an overview of relevant Machine Learning and statistical techniques. Lastly, we discuss Collaborative Filtering and various algorithms for producing recommendations based on collaborative data.

2.1 Personalization

Service providers on World Wide Web operate in a cut-throat environment where even satisfied customers and growth do not guarantee continued existence. As users become ever more proficient in their use of the web and are exposed to a wider range of experiences, they are becoming more demanding, and their definition of what constitutes good service is rapidly changing and being refined. Given the user population of the web, it is difficult to come up with a *one size fits all* approach. A successful mechanism to deal with the demands of such a heterogeneous user population is to modify the contents, characteristics, or appearance of web based systems with respect to a specific user. This is referred to as *Personalization*, and is distinguished from customization by the use of implicit and assumed preferences. While personalization is a broad term, which can also be applied to activities like choosing the color of one's car, or the filtering of TV channels based on the current viewer, we refer to Personalization in the context of software systems and electronic services like those based on the Internet. This is known as *Web Personalization*: However, we use the broader term *Personalization* in this thesis since the concepts behind web personalization are applicable to personalization as a whole.

To measure the user perception of personalization and its effectiveness, surveys were conducted

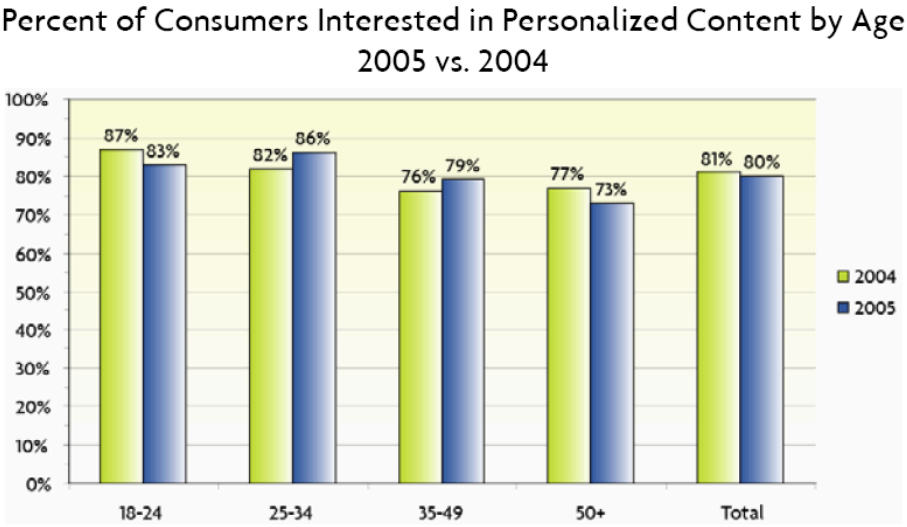

Figure 2.1: ChoiceStream Personalization Survey: the survey shows a continued preference of users towards customized services.

by Choicestream[1] in 2004, 2005 and 2006. According to this study, in 2004, 81% of consumers in the survey were interested in receiving personalized content and service. In follow-up surveys in 2005 & 2006, a similar 80% and 79% said they were interested in personalized content. To get personalized content, 60% of respondents indicated they would be willing to spend a minimum of two minutes answering questions, up from 56% in 2004; the trend continued in successive surveys. Over a quarter (26%) reported they would be willing to spend at least 6 minutes answering questions, up from 21% the year before. Only 12% said they wouldn't be willing to spend any time answering personalization question, down from 14% in 2004. In particular, the study notes that 37% of respondents of all ages reported they would have bought more DVDs/videos if they had found more of what they liked. A third (34 %) reported a similar incongruity with music. Overall, the trends in 2006 are in line with those in 2005. The results of the survey clearly point to the fact that customers realize the value of personalized content; moreover, they are willing to spend more effort and money to get a better service customized according to their individual preferences.

How does Personalization work?

Personalization dynamically adapts a system's service or content offer in order to enhance the quality of a users interaction with the system. Having a *closer customer relationship* as its goal, personalization provides support to satisfy the needs, preferences or goals of individuals and specific target groups (Riecken, 2000). There are various existing techniques for personalization on the Web (cf. (Neuhold et al., 2003; Pretschner & Gauch, 1999)); however, three basic steps are common to these techniques:

Step I collecting data about the user;

Step II inferring the need of the user, and other user characteristics based on collected information, or by interpretation of interaction data, and;

[1] http://www.choicestream.com

Step III customizing or adapting the system to fit the user's needs.

Generally speaking, personalization is achieved through explicit user involvement mechanisms like questionnaires, where users select different content types and services from a list of predefined choices, or provide feedback on the content they have already received. Questionnaire-based personalization does not take into account the dynamic nature of user preferences: users must manually update their profiles when their interests change. To overcome this shortcoming, machine learning and statistical techniques are being used to recognize regularities in the behavior of users and to infer a model of the interests of a user, referred to as *user model*. An instantiation of a user model for a particular user is called a *user profile*.

User profiles can be used to match appropriate content and services to individual users. User profiles model a user's actions possibly in the form of inferred rules and are obtained by recording the navigational history and/or the preferences of each user, resulting in dynamic user profiles. Examples of inferred rules about customers stored in profiles can be: "*When purchasing books from an online store, John Doe usually buys blank CDs*" and "*On weekends, John Doe usually spends more than $100 on online shopping.*" The rules could be exploited for personalized marketing campaigns suggesting items of interest. However, each item contains further information that could be exploited to compute the level of user interest for that item. For example, the description of an item (book) at Amazon.com consists of a set of features such as title, authors, price, editorial reviews, etc. *Content-based filtering systems* suggest items based on their associated features. A pure content-based recommender system is one in which recommendations are made for a user based solely on a profile built by analyzing the content of items (e. g. WebWatcher (Joachims, Freitag, & Mitchell, 1997)) which the user has shown explicit interest in the past, either explicitly, or implicitly. Content-based filtering approaches then find relevant documents or items based on content similarity between the Web documents and the personal profiles obtained explicitly or implicitly from users. *Collaborative filtering systems* on the other hand typically take explicit information in the form of user ratings or preferences, and through a correlation engine, return information that is predicted to closely match the users' preferences. This approach relies on collective judgment of a group of people who are similar to the current user based on commonly rated items. Examples include Firefly (Shardanand & Maes, 1995), and Net Perceptions [2]. *Web mining* (Srivastava, Cooley, Deshpande, & Tan, 2000) is another technique for personalization based on data mining. It is a natural application of data-mining techniques to the Web as a very large and unstructured information source and has a great impact on Web personalization. Through Web mining techniques, such as the discovery of association rules or sequential patterns, clustering, and classification, one is able to gain a better understanding of Web-user preferences, a knowledge that is crucial for mass customization. At this point in the process, the results of the pattern discovery can be tailored toward several different aspects of Web usage mining. For example, Spiliopoulou et al. (Spiliopoulou, Pohle, & Faulstich, 1999), Mobasher et al. (Mobasher, Cooley, & Srivastava, 2000) have applied data mining techniques to extract usage patterns from Web logs for the purpose of deriving marketing intelligence. Shahabi et al. (Shahabi, Zarkesh, Adibi, & Shah, 1997) and Nasraoui et al. (Nasraoui, Frigui, Joshi, & Krishnapuram, 1999) have proposed clustering of user sessions to predict future user behavior.

[2] http://www.netperceptions.com

Drawbacks of current Personalization methods

In order to tailor content and services, systems typically require a representation of the characteristics of its users including for example, the user's needs, goals, environment, cognitive patterns - interests, skills, expertise or preferences. Such representations are typically captured with a *user model*. Current systems typically model a user along a single dimension (e. g. interests) and suffer from a limited view of users, causing a loss of significant amount of potentially useful information about the user. There is no agreed-upon unified theory which systematically integrates all dimensions; instead, different personalization techniques focus on different aspects of a user, and a user's context. Thus, there is a need for more robust, or generic models.

The effectiveness of a personalization system generally improves in the long run, as more data is available about users. Every time a user interacts with a recommendation service, the personalization process collects new data about his/her preferences, so that an improved service can be offered. However, in the e-commerce area, moving from one provider to a competitor is often unfavorable for a customer. Even if a competitor uses a personalization system, it has to learn a lot of information about the new customer to be able to offer the same level of service satisfaction as the previous provider. This problem could be tackled by using cross-system personalization: the knowledge about users could be shared among different systems by keeping the user profile information closer to the user, and each system could separately contribute to enrich that knowledge.

Privacy and control over personal data and its usage is another issue with personalization. In most personalization approaches, the user has no control over his/her user profile. Users cannot see what a system has inferred about their needs and preferences, and what information is collected by the system. Figure 2.2 shows that the ChoiceStream survey also reports privacy as a major user concern, with over 68% of the survery respondants indicating their concern. Initiatives to provide a higher level to privacy include standardizations like P3P (Cranor et al., 2004) and CC/PP (Klyne et al., 2003), which are steps towards giving more power to the user. P3P-compliant web sites can express their privacy practices and a P3P-compliant Web browser could store the user's preferences about those practices. The browser can then make automatic negotiations on behalf of the user over the level of privacy and what information the user is willing to provide. CC/PP is a way to specify precisely what a user agent (e. g. web browser) is capable of doing. This allows for sophisticated content negotiation techniques between web servers and clients, to produce optimized XML-based markup for display and use on a wide variety of web user agents. However, there is a gulf between the adoption of these standards and their integration with existing systems and software. Most websites do not support these standards, relying instead on their own privacy policies who's enactment is legally binding, but rarely enforcement. Customer data is routinely released (e. g. the AOL query log) into the public space, and leakages happen from time to time. Thus the adoption of privacy preservation is a fundamental requirement in new technology and research for personalization.

2.2 User Modeling

A *user model* is a data model which captures different characteristic of a human user when in interaction with an electronic system. User models can be used to describe the interests and preferences of a user, so that these user-specific characteristics are taken into account by an electronic system. Based on our literature survey, we find 4 major types of user models:

Figure 2.2: ChoiceStream Personalization Survey: the survey shows a user perception that personalization may lead to misuses of their personal information

Cognitive Pattern-based models, Task-based models, Environmental models and Relation-based models.

Traditional models of users are based on a mentalist paradigm (Pohl, 1997), using characteristics of the user which we collectively refer to as *Cognitive patterns*. These patterns represent user-specific aspects and include for example: interests, knowledge, preferences, misconceptions, or abilities. Systems incorporating models of user interests (Fink & Kobsa, 2002; Kobsa, 2001) have been widely used to selectively filter information on behalf of users from a large, possibly dynamic information source (Baudisch, 2001). A common example of an interest-based model is a collaborative filter which infers a user's interest and preferences from the ratings that the user gives to an information item and from similarities between other users' interests (Konstan et al., 1997; Pazzani, 1999). Despite studies which suggest that cognitive pattern models such as interest are insufficient data for accurate models of the user, it seems likely that these systems will continue to be adopted in the future; therefore we consider this traditional modeling dimension as a significant indication of the user characteristics.

Task models of users are considered important (Kaplan, Fenwick, & Chen, 1993) based on the assumption, that the goals of users (who participate in a task) can influence their information needs. When these needs are known in advance, a system can better adapt to its users (Tyler & Treu, 1989; Vassileva, 1994). Based on these goal-driven theories for information related-activity, we consider tasks an important dimension in modeling users and their context.

Environmental models are considered a key issue with respect to the interaction between human and computer because they describe the surrounding facts or assumptions which provide a meaningful interpretation (context) to a user's computer usage when their physical environment varies (Schmidt, Beigl, & Gellersen, 1999). Furthermore, researchers have suggested that future usage scenarios will require more sophisticated support for changes that occur in a user's location and infrastructure. Such scenarios include: multi-computer usage: (e. g. a PC at work, a laptop on the go, and a PC at home); mobile computing: where a user carries a small information devices that can be temporarily connected to a network or ubiquitous information: where the information space can be accessed from information walls, kiosks, or desktops (Fink & Kobsa, 2002) and federated services: where collective information is dispersed among information sources. Given the aforementioned trends and scenarios, environmental models are an important dimension in

adequately supporting aspects of the users' situation and environment.

Relation-based models of a user are information and community models that take into account the salient interrelationships of individuals in a cooperation or community context (McDonald, 2003). Having its roots in social theories, these systems use graph-based, or complex network structures to model interactions between human beings. Specifically, social network analysis (SNA) (Wasserman & Galaskiewicz, 1994) extends and complements traditional social science by focusing on the causes and consequences of relations between people and among sets of people (e. g. EgoNet [3]). Another approach to SNA is ego-centric network analysis. This approach focuses on an individual (or ego) and uses this individual's network of relations to understand the diverse factors contributing to his/her behavior and attitude (Newman, 2003).

More general than social networks, relations as well relation types are considered of high significance in modeling users and information. Relation types describe common properties for a class of relation and include, for example, containment relations such as part-whole and class inclusion (Artale, Franconi, Guarino, & Pazzi, 1996) as well as non-containment relations such as thematic roles (or case relations). The *thematic-roles* of a user are important in modeling a user's interaction with his environment because they represent a function, behavior, or assigned characterization that a participants plays in an association (Shapira, Shoval, & Hanani, 1997). One type of participant from these classifications includes a determinant: an entity which is an active participant who initiates or determines the direction of process. Other types include immanent and recipient. In addition to ontology-based classifications, relational elements theories have been used to describe inherent properties of the relations themselves . One important application area of relation-based models is *bibliometric analysis* of scientific data (Barabási et al., 2002).

Generic User Modeling

Given that the current user models are mostly one-dimensional (see Sec. 2.2), there is a need for a unified approach to user modeling. A number of factors contribute to the proposal in support of generic or unified user models. On the one hand, given the number of aforementioned dimensions that is possible when modeling users, researchers have considered a generic approach to modeling users, because at present, there is no unified theory which systematically integrates all dimensions. On the other hand, current systems which typically model a user along a single dimension suffer from a limited view of users and a significant amount of potentially useful information about the user may be lost; thereby demanding a need for more robust models. A unified user modeling approach would take into account the domain knowledge that might be required for various applications. In order to support personalization across multiple systems, a broader understanding of the user is required as is also discussed in (Niederée, Stewart, Mehta, & Hemmje, 2004; Kobsa, 2001). Section 3.2.4 discusses this aspect in detail and propose a solution in the form of a *unified user context model* (UUCM). Finally, similar work to build unified user-related models for dynamic information spaces in RDF and a standardized RDF vocabulary has been proposed as a part of the CC/PP framework (Composite Capabilities/Preferences Profile) (Klyne et al., 2001).

2.2.1 Representation Formats and Standards for User Profiles

Besides these more generic aspects of user modeling, there are also some efforts in standardizing user model related aspects, mostly in application-specific areas. The vCard specifica-

[3] http://survey.bebr.ufl.edu/EgoNet/

tion (Dawson & Howes, 1998) from the Internet Mail Consortium is a means of Personal Data Interchange (PDI), which automates the traditional business card. Another related standard is X.500, commonly known as LDAP. LDAP provides directory services for querying, as well as an information model based on object classes defined in the standard. The *IMS Learner Information Package (LIP)* (Colin Smythe & Robson, 2001) specification offers a data model that describes characteristics of a user needed for the general purpose of recording and managing learning related history, goals and accomplishments. In addition, the *IEEE Public And Private Information (PAPI)* (IEEE, 2000) specification was created to represent student records. Its development is moving towards harmonization with IMS. (Rousseau, Browne, Malone, & ÓFoghlú, 2004) discusses some of the above standards in more detail. Privacy and control over personal data and its usage is also an issue in user modeling. In most personalization approaches, the user has no or little control over his/her user profile. P3P (Cranor et al., 2004) is a step towards giving more power to the user. P3P-compliant web sites can express their privacy practices and a P3P-compliant agent (e. g. Web browser) can automatically negotiate on behalf of the user over the level of privacy.

- *vCard (version 3)* - The vCard specification from the Internet Mail Consortium is a means of Personal Data Interchange (PDI) (Dawson & Howes, 1998), which automates the traditional business card. It can be used to store vital directory information (name, addresses, telephone numbers, email, URLs), geographic and time zone information, and can include graphics and multimedia (photo, logos, audio clips). The vCard has multiple language support, is standards based and the specification (based on RFC 2425 and RFC 2426) is transport and operating system independent.

- *LDAP* - LDAP provides directory services for querying, as well as an information model based on object classes defined in the standard. The LDAP information model structures data as a tree - *the Directory Information Tree* (DIT). An entry in the DIT corresponds to a node in the tree, and contains information about an object Class. *ObjectClasses* have both required and optional attributes, and attribute typing defines the encoding and matching rules to be used during searching. The LDAP information model is also called the LDAP schema. There is also a standard text-based format for describing directory entries called LDIF.

- The *IMS Learner Information Package (LIP)* (Colin Smythe & Robson, 2001) specification offers a data model that describes characteristics of a user needed for the general purpose of recording and managing learning related history, goals and accomplishments; for engaging the user in a learning experience and for discovering learning opportunities for users. The main elements are: Accessibility, Activity, Affiliation, Competency, Goal, Identification, Qualifications and certifications, Relationship, Security key and Transcript.

- The *IEEE Public And Private Information* (PAPI) (IEEE, 2000) specification was created to represent student records. Its development is moving towards harmonization with IMS. It specifies data interchange formats, facilitating communication between cooperating systems. User records cover personal information and performance information. The current specification is well structured and splits the learner information into the following areas: personal information, relations' information, security information, preference information, and portfolio information.

- *CC/PP* (Klyne et al., 2003) provides a way to specify precisely what a user agent (e. g. web browser) is capable of doing. This allows for sophisticated content negotiation techniques

between web servers and clients, to produce optimized XML-based markup for display and use on a wide variety of web user agents.

The above standards are well known, but suffer from some drawbacks. vCard is too simple a format to store user profiles and is best suited for light weight user profiles like contact information or directories. While LDAP allows storing user information as entries made up of attributes, the directory schemas place restrictions on the attribute types that must be or are allowed to be contained in an entry. LDAP does not address problems such as the classification of user interests, but does provide a widely implemented standard for representing name, address and contact detail information. IMS and PAPI are more generic and based on standards like XML. However, they are not conceptually extensible. Moreover, applications today require user profiling which takes into account the domain knowledge, e. g. a book site's user profiling requirements focus on transactional and browsing information and classification of interests, which is different from the requirement of a UI centric application like *My Yahoo!* [4] where the user specifies what s/he is interested in seeing on his/her personalized homepage.

2.2.2 Personalization Engines and User Modeling servers

User modeling servers are systems that (at least partially) factor personalization related functionality out of the rest of the system and provide this functionality as a separate component or layer within the system, ideally in an application independent form. In the following, we review selected user modeling servers that are available as standalone products. For a detailed analysis of these servers, we refer to (Kobsa, 2001).

GroupLens: Net Perceptions [5] has its roots in work on collaborative filtering systems developed at the University of Minnesota with the GroupLens project (Konstan et al., 1997). Their personalization product called NetP consists of a recommendation engine and a set of APIs to access it. With these APIs, applications can send ratings to, and receive predictions from the recommendation engine. This has evolved from the GroupsLens toolkit which was the earliest product of Net Perceptions. For user input, GroupLens could deal with numerical ratings provided explicitly by the users, or implicit ratings provided by the applications using, e. g. browsing patters or shopping card analysis. Whereas the user ratings and navigations data can be processed at runtime, past purchase data as well as past ratings can only be taken into consideration during bootstrapping. Since 2004, Net Perceptions has closed operations and Tornago, a company formed by former NP employees continues development & support of the NetP product line.

Personalization Server At end of 1998, Art Technology Group (ATG)[6] released its product named *Personalization Server* as a complement to their previously released Application Server. Personalization Server extends the functionality of the Application Server by profile management and a rule-based development and runtime personalization environment. Rules in the personalization server are defined on user groups which consist of users with similar profiles w.r.t. some attributes. Group profiles comprise relevant characteristics (e. g. age, gender) of user subgroups (e. g. family, father). Rules that are associated with group profiles allow Personalization Server to assign an individual user to one or more user groups. These

[4] http://my.yahoo.com
[5] now renamed as Tornago (http://www.tornago.com)
[6] http://www.atg.com

rules can take user data (e. g. demographic data like gender and age), implicit information about system usage (e. g. pages visited, products bought) as well as environmental information into account (e. g. domain name, browser type, operating system, available bandwidth). The recommendations made by the engine are a result of the rules applicable to a given user.

FrontMind FrontMind (from Manna) provides a rule-based development, management and simulation environment for personalized information and personalized services on the Web. FrontMind distinguishes itself from other rule-based products like Personalization Server by having Bayesian networks for modeling users' behavior integrated into their product. Several major differences become apparent when comparing FrontMind with Personalization Server regarding data acquisition and representation: FrontMind maintains dynamic models of users' behavior, which can take arbitrary user and usage related information into account, whereas Personalization Server relies on rather static group profiles and associated acquisition and activation rules. FrontMind employs rules mainly for adaptation purposes, whereas Personalization Server also utilizes rules for acquiring assumptions about the user and for assigning profiles of user groups to individual users. Besides static user and usage related information, FrontMind's adaptation rules can also take advantage of users' behavior models.

Learn Sesame Learn Sesame relies on applications for collecting implicit and explicit user, usage, and environmental data. Relevant usage characteristics (e. g. keywords of requested hypermedia pages, ratings of products, keywords entered in a search form) have to be collected by applications and sent to the user modeling server along with relevant user characteristics (e. g. user id, age, gender, sex, income). Learn Sesame analyzes this stream of time-stamped events for recurrent patterns, and supplies applications with evidences for regularities (e. g. a user's presumed interest in outdoor clothing, a correlation between the amount of money spent and suitable product categories, a correlation between product category and user demographics like age, gender, income, and formal education for a group of users). Learn Sesame's learning algorithms are based on incremental hierarchical clustering.

Overview and Discussion

In the following subsection, we discuss the current features of the above mentioned servers along the following dimensions: functionality, user data acquisition, quality of recommendations, and privacy. A more detailed analysis has been performed by Fink and Kobsa (Kobsa, 2001; Kobsa & Fink, 2003)

Functionality Compared to the restricted set of input data for GroupLens and the rather tight integration of Personalization Server with a single user-adaptive application (environment), FrontMind's configuration facilities for input data and Learn Sesame's domain modeling facilities with their inherent application independence and flexibility seem to be clearly superior. With Learn Sesame, application programmers can communicate information about the domain at hand and control the associated learning process at an appropriate level of abstraction.

Use Data Acquisition With regard to acquisition methods, GroupLens uses collaborative filtering, Personalization Server offers (simple) production rules that mainly operate on individual user profiles and stereotypes, FrontMind employs (simple) production rules that take advantage of Bayesian networks, and Learn Sesame employs hierarchical clustering.

Quality of Recommendations Business practices can often be implemented straightforwardly in rule-driven personalization environments. Moreover, rule-driven personalization allows businesses to be very explicit. From a user's point of view, however, the effects of a solely rule-driven personalization are often found to be quite deterministic. Unlike non-deterministic recommendations, rule-driven personalization leaves barely any room for users' serendipity. This is mainly due to the fact that the underlying representation system for user information can hardly deal with uncertainty and with changes in user behavior. Keeping track of changing user interests and preferences in real time is, however, a main motivation for user modeling from a marketing point of view. Even worse, rule design, update and management are primarily a manual process and therefore cumbersome and error-prone. Therefore, user modeling servers like Personalization Servers that solely rely on rule-based personalization and stereotypes seem to have severe shortcomings. Systems like FrontMind that exhibit both deterministic and non-deterministic personalization behavior seem to have a significant competitive advantage.

Privacy ATG, and to some extent FrontMind, seem to be rather careless regarding privacy, compared for example to the efforts undertaken by Net Perceptions. This is somewhat surprising since many tool vendors, their customers and the (online) marketing industry actively propagate and contribute to self-regulation with regard to privacy, in order to prevent governments from requiring to pass more restrictive privacy laws. For more details on the privacy issue, see (Schreck, 2003).

2.3 Machine Learning and Statistical Techniques

A drawback of traditional electronic systems is their inability to cope sensibly with new or unexpected situations, leading to sudden crashes or unexpected outcome. Anticipating every possible scenario and defensively programming computer systems is possible only in restricted scenarios. Clearly, to operate autonomously in a real world setting, electronic systems have a key requirement to learn from new situations and adapt accordingly. The field of *Machine Learning* has evolved from this requirement in the Artificial Intelligence community. In this field, one considers the important question of how to make machines able to learn. Learning in this context can be of different types, one of which is *inductive inference*, where one observes examples that represent samples of some statistical phenomenon. In *unsupervised* learning one typically tries to discover inconsistencies, anomalies in observed data, similar conceptually to data mining. In *supervised learning*, one typically has input and output data of a given sample of observations, where one tries to infer functions which map the input to output with minimum error. An example of this is weather prediction, e. g. given parameters like precipitation, humidity, temperature etc, to guess the chances of rain. Output data is often one dimensional and each output is called a *label*. If labels are discrete and in a small range (say 1–5), then this task is called *classification problem*. Examples include classifying documents as either belonging to a topic, or not belonging to a topic. For real-valued labels, the term used is *regression*. In classification and regression, one is particularly interested in generalizing from observed examples and predicting the output for other cases for which only input data is observed. The usage of unlabeled input data for the purpose of learning a function in addition to observed input–output pairs is known as *semi-supervised learning*. A recent category of Machine learning techniques has emerged for learning problems where the structure of the output is known apriori; this class of problems are known as *structural classification*.

In this thesis, we have extensively used techniques in dimensionality reduction, which broads falls under the category of unsupervised learning methods.

2.3.1 Dimensionality Reduction

Advances in data collection and data storage technologies has lead to a large amount of information which cannot be humanly analyzed. Many data analysis techniques also do not scale to the size of data now available, since traditional techniques cannot deal with the dramatic increase in the number of observations. Even more problematic is the increase in number of variables per observation. The dimensionality of data is the number of variables associated with every observation. As an example, consider a set of photographs taken by a 2 megapixel camera. Each image is an observation, with every pixel as a variable. Such data thus has 2 million dimensions associated with each observation (image).

One important observation about high dimensional data is that a large number of variables in observed data do not provide interesting information. As an example, consider the problem of detecting whether an image has a human face or not, given that images are taken against a black background. In this case, we can expect a large number of pixels to be black, therefore not providing any additional information. This observation is fundamental and inspires the field of dimensionality reduction where variables with redundancy and low information are discarded and a low dimensional representation is created for every observation. Mathematically, we motivate the problem as follows: consider n observations of a p-dimensional random variable $\mathbf{x} = \{x_1, x_2, ..., x_p\}$. We are interested in finding a lower dimensional representation of $\mathbf{x}$, represented as $\mathbf{z} = \{z_1, z_2, ..., z_k\}$, where $k << p$, and where $\mathbf{z}$ exhibits almost the same statistical properties as the original data $\mathbf{x}$.

2.3.2 Linear Methods for Dimensionality Reduction

Dimensionality reduction is often performed by making certain assumptions about the underlying data. One such assumption is that variables in observed data are linearly related. Under this assumption, *linear methods* such as Principal Component Analysis (PCA) (Jolliffe, 2002) and Factor Analysis (Everitt, 1984) as used. Both PCA and FA have been used in various domains. Both these methods also perform a dimensionality reduction of the following form, although with different assumptions:

$$\mathbf{x} = \Lambda \mathbf{z} + \eta\,, \tag{2.1}$$

The goal of both approaches is to estimate the loading matrix Λ and the covariance matrix ψ of the additive noise η, under the assumption that the lower dimensional data has a fixed dimensionality k, with each dimension representing an unobserved *factor*. In PCA, the factors and loading matrix are chosen under the assumption that the factors have unit variance. In Factor Analysis, the covariance is also modeled, but only diagonal variance is assumed.

Principal Component Analysis

Principal component analysis (PCA) is the simplest, and the best (in the mean-square error sense) linear dimension reduction technique. Being based on the covariance matrix of the variables, it is a second-order method. In various fields, it is also known as the *Karhunen-Loeve transform*, or the *Hotelling transform*. In essence, PCA seeks to reduce the dimensionality of the data by finding a few orthogonal linear combinations (called the *Principal Components*)

of the original variables with the largest variance. The first principal component is $s_1 = \mathbf{x}^T w_1$, where the p-dimensional coefficient vector $\mathbf{w}_1 = (w_{1,1}, ..., w_{1,p})^T$ solves

$$\mathbf{w}_1 = \underset{\|\mathbf{w}\|=1}{\operatorname{argmax}} \; Variance\{\mathbf{x}^T\mathbf{w}\}, \tag{2.2}$$

The second PC is the linear combination with the second largest variance and orthogonal to the first PC, and so on. There are as many PCs as the number of the original variables. Principally, this is equivalent to performing an eigen-decomposition of the *covariance* matrix of the original data. Suppose the data is represented by the matrix $\mathbf{X}^{m\times n}$, where each column corresponds to an observation $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, ..., x_{i,m}\}$. For the sake of simplification, we assume that the data is zero-centered[7]. Now the covariance matrix $\mathbf{C}$ is computed. $\mathbf{C}$ is defined as:

$$\mathbf{C} = \frac{1}{n-1}\mathbf{X}.\mathbf{X}^T, \tag{2.3}$$

Using the Spectral decomposition theorem (Jolliffe, 2002), we can write the symmetrical matrix $\mathbf{C}$ as:

$$\mathbf{C} = \mathbf{U}\Lambda\mathbf{U}^T, \tag{2.4}$$

where $\mathbf{U}$ is an Unitary Normal Matrix[8] and Λ is a diagonal matrix containing eigenvalues of $\mathbf{C}$. It can be shown that the Principal components (PCs) can be given by the rows of the matrix $\mathbf{S}$ where

$$\mathbf{S} = \mathbf{U}^T\mathbf{X}, \tag{2.5}$$

By ordering the rows of $\mathbf{U}$ in the order of eigenvalues of $\mathbf{C}$ (which comes from the Spectral decomposition theorem), we get the PCs in ascending order (i.ethe first row represents the first PC, etc.). An important property of this order is that PCs model the overall variance of data $\mathbf{X}$ in proportion of the corresponding eigenvalues.

Note: We use PCA in Sec. 3.4.4 for developing a spam detection procedure for Collaborative filtering.

Probabilistic PCA

Principal component analysis is a popular technique for data analysis and processing, but it is based on assumptions of complete data. In case data is missing, an underlying probability model has to be assumed. However, a pure probabilistic method like Factor Analysis which models noise systematically as well produces a subspace which does *not* corresponding to the principal subspace. (Tipping & Bishop, 1999) bridge this gap by modeling isotropic noises with variances $\psi_i = \sigma^2$ being the same for all variables, a model called *Probabilistic Principal Component Analysis* (PPCA). The interested reader is referred to (Tipping & Bishop, 1999) for further details: the following section on Factor Analysis provides an analytical insight into PPCA as well since FA is a more general form of PPCA.

Factor Analysis

Factor analysis is used to uncover the latent structure underlying a set of variables and as such is a *non-dependent* procedure that does not require to explicitly specify dependent variables.

[7] If data is not zero centered, a simple linear transform can be used by subtracting the mean of every dimension.
[8] A matrix $\mathbf{A}$ is normal if $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T$, and unitary normal if $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I}$

It can be used to analyze the patterns of relationships between observed variables, eventually discovering the underlying (fewer and fundamental) independent variables that may not be directly observed. The inferred variables are called factors. A typical application of factor analysis suggests answers to following questions:

1. What are the latent factors underlying the data?
2. In which way do these factors explain correlations between observed variables?
3. How much of the observed variability is accounted for by latent factors, how much should be considered noise?

Factor analysis is also a generative model for *high dimensional data*, which is actually based on a small set of factors. Factor analysis is used to uncover the latent structure of a set of (observed) variables within such data, and to reduce the attribute space from a larger number of variables to a smaller number of factors.

Factor analysis is a latent variable model in which dependencies and correlations between multiple observable (dependent) variables $\mathbf{x}$ are explained by virtue of a typically much smaller number of latent variables or *factors* $\mathbf{z}$. The functional relationship between the observed random vector $\mathbf{x}$ and the unobserved $\mathbf{z}$ is assumed to be linear with some additive zero mean Gaussian noise added to each dimension of $\mathbf{x}$ independently. The fundamental equation that relates observables and latent factors can thus be described as

$$\mathbf{x} = \mu + \Lambda\mathbf{z} + \eta, \quad \eta \sim \mathcal{N}(0, \Psi), \tag{2.6}$$

where $\mu \in \mathbb{R}^m$ is a constant offset vector (mean), $\Lambda \in \mathbb{R}^{m\times k}$ is the matrix of factor loadings, and $\Psi = \mathrm{diag}(\psi_1, \ldots, \psi_m)$ is a diagonal matrix modeling the variance of the additive Gaussian noise η. To complete the model, one usually assumes that *a priori* $\mathbf{z} \sim \mathcal{N}(0, I)$, i.e. the k latent factors follow an isotropic normal distribution with unit variance. The key assumption in factor analysis as in many latent class models is that conditioned on the latent classes, the observables are rendered independent; hence the crucial requirement on Ψ to be diagonal. It can be shown by integrating out the latent variables $\mathbf{z}$ that the distribution induced by factor analysis on the observables is a multivariate normal of the form

$$\mathbf{x} \sim \mathcal{N}(\mu, \Lambda\Lambda' + \Psi)\,. \tag{2.7}$$

This shows that factor analysis can be thought of as a multivariate normal model in which certain constraints are imposed on the co-variance matrix.

Note: We use FA in Sec. 3.3.3 for developing a learning solution for CSP.

2.3.3 Non-Linear Methods for Dimensionality reduction

As opposed to a linear relationship between variables, a non linear relationship can also be observed. A common example of such data is shown below in Figure 2.3, where data lies on a spiral, which can be unrolled to 2-D plane, thus with an intrinsic dimensionality of 2. In such cases, linear dimensionality reduction methods fail to identify the independent variables correctly and non-linear techniques have been developed to fill this gap. Non Linear methods do not generalize data globally, instead looking for local properties such as distances which are nearly linear. Methods such as Laplacian Eigenmaps (Belkin & Niyogi, 2003) and Locally Linear Embedding (LLE) (Saul & Roweis, 2003) aim to discover a d-dimensional subspace, given m-dimensional data, such that local distances are preserved. Other methods of non linear dimensionality include Multi Dimensional Scaling (MDS) and Hessian LLE.

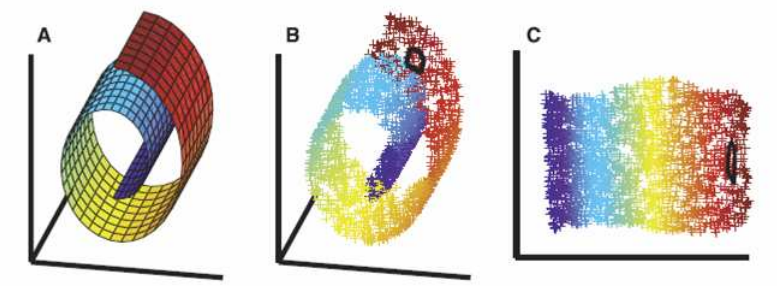

Figure 2.3: A synthetic example of data that lies on a manifold

Laplacian Eigenmaps

Suppose we are given n data points x_i in $\mathbb{R}^m$. When these data lie approximately on a low-dimensional manifold embedded in the n-dimensional Euclidean space, manifold learning methods such as *Laplacian Eigenmaps* (Belkin & Niyogi, 2003) and *Locally Linear Embeddings* (LLE) (Saul & Roweis, 2003) can be used to recover the manifold from a sample set $\mathcal{S} = \{x_1, x_2, ..., x_n\}$. Here we explain the Laplacian Eigenmap approach, for which rigorous convergence results exist in the large sample limit (Hein, Audibert, & Luxburg, 2005).

The starting point in Laplacian eigenmaps is the construction of a weighted graph whose nodes are the sample points and whose edges connect the nearest neighbors of each node. Neighborhoods may consist of the k-nearest neighbors of a sample point or the set of all points that are within an ϵ-ball. We write $i \sim j$ as a shorthand for sample points $\mathbf{x}_i$ and $\mathbf{x}_j$ that are neighbors. The weights W_{ij} between neighbors are usually assumed to be non-negative and symmetric, $W_{ij} = W_{ji} \geq 0$ and are summarized in an affinity matrix $\mathbf{W}$. There are several alternatives on how to define these weights when starting from a vector-valued representation over $\mathbb{R}^m$, one popular choice being the Gaussian kernel,

$$W_{ij} \equiv \exp\left[-\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2\right], \tag{2.8}$$

where $\beta > 0$ is a suitably chosen bandwidth parameter. Another choice is to compute weights based on a local affine approximation over neighbors, as discussed in the following subsection on LLE.

The heart of the Laplacian eigenmap approach is the generalized graph Laplacian $\mathbf{L}$ defined as,

$$\mathbf{L} = (L_{ij})_{i,j=1}^{n}, \quad L_{ij} = \begin{cases} \sum_{j \sim i} W_{ij}, & \text{if } i = j \\ -W_{ij}, & \text{if } i \sim j \\ 0, & \text{otherwise}. \end{cases} \tag{2.9}$$

An Laplacian eigenmap is a function $f : \mathcal{S} \to \mathbb{R}$ for which $\mathbf{L}f = \lambda f$ and $\|f\|^2 = 1$, where we think of f as a vector of function values for convenience. Moreover, in order to remove the trivial solution with $\lambda = 0$ one can add the constraints $(1, \ldots, 1)f = \sum_{i=1}^{l} f_i = 0$. It can be shown that the eigenmap corresponding to the smallest eigenvalue $\lambda > 0$ minimizes the criterion

$$f^T \mathbf{L} f = \sum_{i,j} W_{ij}(f_i - f_j)^2. \tag{2.10}$$

The eigenmaps corresponding to the d smallest eigenvalues span a d-dimensional coordinate system on the low-dimensional data manifold.

In the case of semi-supervised learning one may utilize $f^T\mathbf{L}f$ as a regularizer and combine it with supervised information about target values t_i that may be available at some subset $\mathcal{S}' \subseteq \mathcal{S}$ of the nodes of the graph to define the regularized solution (cf. (Belkin, Matveeva, & Niyogi, 2004))

$$f^* = \arg\min_f \sum_{\mathbf{x}_i \in \mathcal{S}'} (f_i - t_i)^2 + \lambda f^T\mathbf{L}f\,. \tag{2.11}$$

Locally Linear Embedding

The Locally Linear Embedding algorithm has the same conceptual basis as Laplacian Eigenmaps, hence we will continue to use the same symbols. The crucial difference between LLE and Laplacian eigenmaps is in the choice of the weights W_{ij} for neighboring nodes in the graph Laplacian (see Eq. (2.9)). The method used in LLE is to compute W_{ij} based on a local affine approximation. For a sample of l data points $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^m\colon\ i = 1, \ldots, l\}$, LLE proceeds as follows:

- For each data point $\mathbf{x}_i$, compute the K nearest neighbors in $\mathcal{S}$ which are closest to $\mathbf{x}_i$ in Euclidean distance.
- Compute for each $\mathbf{x}_i$ the optimal approximation weights for an affine local regression over the neighbors. This is equivalent to approximating the nonlinear manifold at $\mathbf{x}_i$ by the linear hyperplane that passes through the neighboring points. This step of the algorithm amounts to solving a quadratic optimization problem:

  $$W^*_{ij} = \arg\min_W |\mathbf{x}_i - \sum_{j\sim i} W_{ij}\mathbf{x}_j|^2\,, \text{s.t.} \sum_j W_{ij} = 1\,, \tag{2.12}$$

  where $j \sim i$ indicates that $\mathbf{x}_j$ is a neighbor of $\mathbf{x}_i$ (notice that the relation is in general not symmetric).
- Finally, a low-dimensional representation $\hat{\mathbf{x}}_i$ is computed by solving the minimization problem

  $$\hat{\mathbf{X}}^* = \arg\min_{\hat{\mathbf{X}}} \sum_i \|\hat{\mathbf{x}}_i - \sum_{j\sim i} W_{ij}\hat{\mathbf{x}}_j\|^2 \tag{2.13}$$

  This can be shown to be equivalent to an eigenvector decomposition problem involving the matrix

  $$\mathbf{M} = (\mathbf{I} - \mathbf{W}^*)^T(\mathbf{I} - \mathbf{W}^*) \tag{2.14}$$

  where I is the $l \times l$ identity matrix. The bottom $d+1$ eigenvectors of M (excluding the smallest, which is 1) form a co-ordinate system for the low dimensional data manifold.

Please note that the matrix $\mathbf{I} - \mathbf{W}^*$ corresponds to the graph Laplacian $\mathbf{L}$ (defined in Eq. (2.9)) for a graph with $\sum_{j\sim i} W_{ij} = 1$ for all graph nodes. Also note that the graph Laplacian thus formed is not symmetric and the weights can be negative. Multiplying the $\mathbf{L}$ with its transpose gives a symmetric matrix $\mathbf{M}$ (see Eq. 2.14). (Belkin & Niyogi, 2003) explains that under some conditions, the matrix $\mathbf{M}$ is approximately the same as $\mathbf{L}^2$, which has the same eigenvectors as $\mathbf{L}$, with eigenvalues which are the square of the eigenvalues of $\mathbf{L}$. It has been shown in (Ham, Lee, & Saul, 2005) that the matrix $\mathbf{M}$ can be substituted for the graph Laplacian $\mathbf{L}$ in the aligned manifold method.

Note: We use LLE & Laplacian Eigenmaps in Sec. 3.3.1 for developing a learning solution for CSP.

2.4 Collaborative Filtering

Collaborative Filtering (Shardanand & Maes, 1995; Konstan et al., 1997) is one of the most popular and successful filtering techniques that has been used to date. It is applied in a setting where users have a choice between a number of items (say, all books in a book store) and provide votes to items that they know about. Collaborative Filtering helps users to make choices based on the opinions of similar users in a system and find relevant items that they may not have explored so far. The basic idea employed is that users who agree with each other on some items based on their ratings are likely to agree or disagree on future items. To make predictions for a given user, collaborative filtering algorithms typically find similar users in a system, and assign weights to the level of similarity. The preferences of this set of similar users is combined and weighted with the assigned weights. This technique has its basis in every-day life where people consider the opinions of similar minded people in order to decide what they want to buy next (e. g. a music CD). Collaborative filtering is therefore an algorithmic form of *word-of-mouth* process.

Collaborative filtering algorithms are now widely used in Internet applications, with considerable success. For example, *Amazon.com* and *CDNow.com*, the largest online book and music stores respectively on the web, use collaborative filtering to provide personalized information filtering for users. Many other recommender systems have been developed using this technology, such as MovieFinder.com, Belcore Video Recommender (movie recommendation sites), Levis Style Finder (www.levis.com, a clothing recommender system), and lately, NetFlix, an online DVD rental store [9].

		Airplane	**Matrix**	**Room with a View**	**...**	**Hidalgo**
		comedy	**action**	**romance**	**...**	**action**
Joe	*27, M, 70k*	1	1	0		1
Carol	*53, F, 30k*	1		1		0
...						
Kumar	*25, M, 62k*	1	0	0		1
U_a	*48, M, 81k*	1	1	?	?	?

Figure 2.4: A synthetic example of Collaborative Filtering data with the task of predicting values for the user U_a

[9] A list of more than 70 websites using Collaborative Filtering is available at http://en.wikipedia.org/wiki/Collaborative_filtering

2.4.1 Types of Collaborative Filtering Algorithms

Collaborative filtering algorithms have been classified into two general categories, commonly referred to as *memory-based* and *model-based* algorithms (Breese, Heckerman, & Kadie, 1998). *Memory-based algorithms* are the more prevalent of the two categories and use all available data in order to make a prediction for the selected user. The system database contains sets of user preferences, recording the transactions that are made by all users of the system. Memory based CF algorithms retain all relevant data in memory and compute the required prediction on demand in real. The advantage of this approach is that new data provided by a user can immediately be taken into account. Typically, this provides a better usability experience, as the user can see how his/her actions are immediately utilized by the system. However, the scalability of such systems is not arbitrary; using memory based algorithms for real-world systems requires optimizations and some approximations have to be made, which can counter the accuracy of the original method. Nonetheless, several algorithms have been proposed for memory based CF due to their high accuracy and simplicity of implementation.

Model-based collaborative filtering algorithms operate in a different manner to memory based algorithms by abstracting from the observed data and creating a statistical model of observed data. This model is learnt based on known ratings and is subsequently used in the recommendation process. Most model based algorithms model the collaborative filtering problem as a missing value problem: the user-item matrix which records known ratings is very sparse, and the objective is to learn to find appropriate values for the unobserved values. Model based methods use techniques from the field of Machine Learning.

The different strategies employed in memory-based and model-based collaborative filtering algorithms have an impact on the performance and running time. Memory algorithms tend to be completely online, where computations are performed when a particular recommendation is required, and these values are not stored. While such approaches are often more accurate and take into account only the most recent data, the computational time is very high. To scale these algorithms to millions of items and users, optimizations are required. Such optimizations include user sampling, pre-computation of similarity and/ or neighbors, and caching of pre-computed results.

Model-based algorithms in contrast are generally small, efficient and involve a large offline phase for model training. However, once the model has been learnt, computing a recommendation is very quick, often taking $O(1)$ time. Successful model based methods deal with data sparsity in a principled way and use global trends in data rather than a small set of neighbors. Due to this, the coverage of model-based algorithms tends to be 100%. However the accuracy of early model based methods was a little worse than memory based algorithms until recently. Newer approaches like PLSA (Probabalistic Latent Semantics Analysis) (Hofmann, 2004), and Sparse Factor Analysis (Canny, 2002b) outperform traditional memory based collaborative filtering algorithms.

To summarize, model-based systems offer the advantage of fast and efficient recommendation generation, at an additional cost of a time consuming offline computation and can scale to large datasets. However, model based CF approaches are suitable for applications where data is infrequently updated, as models are not rebuilt frequently. In scenarios of rapid data influx, or frequent updates of user data, memory based algorithms are far more accurate, and with optimizations, can scaled to larger datasets.

2.4.2 Relevant Collaborative Filtering Algorithms

In this thesis, we use some well known CF algorithms as a basis for developing our techniques. Some of these algorithms are also used as baseline and the Gold standard. These algorithms are

1. Popular voting
2. 'k-Nearest Neighbor' based algorithms for collaborative filtering
3. Factor Analysis
4. Probabilistic Latent Semantics Analysis

Popular voting

Popular voting uses mean rating of every item and recommends the mostly highly rated items to the active user. This form of recommendation is non-personalized and every user receives the same recommendations. While the performance of such a strategy is clearly suboptimal, the difference between this simple-minded strategy and the best available methods is usually the order of 10-15%. Since this algorithm is the only one which can be used for a new user about whom no data is known, we use this algorithm as a baseline.

k-NN based algorithms for collaborative filtering

Basic collaborative filtering systems use a weighted reconstruction of the votes of users similar to the current user to predict the likely rating for a previously unrated item. Various improvements have been made to the basic mechanism of predicting votes using Pearson's correlation, but they mostly comply to the following scheme: assume the user database consists of a set of votes $v_{i,j}$ corresponding to the vote for user i on item j. The predicted vote for an active user for item j, $p_{a,j}$ is a weighted sum of the votes of other users:

$$p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \overline{v}_i) \tag{2.15}$$

where $w(a,i)$ is the weight given to every user i from active user a, $\overline{v}_i$ and $\overline{v}_a$ are the average rating given by users i and a, and κ is a normalization factor.

Pearson's Correlation based Collaborative Filtering: The most popular memory-based algorithm uses a similarity measure called Pearson's Correlation. This is a standard measure in statistics, which is applied here with only a small modification: similarity is measured based only on items where votes are available for both users. Predicted votes $v(i,j)$ are computed as defined in Eq. (2.15) with similarity weights $w(a,i)$ defined as follows:

$$w_{PC}(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}} \tag{2.16}$$

Various modifications to the above scheme have been proposed in literature (cf. (Karypis, 2001; Herlocker, Konstan, & Riedl, 2002)) which can lead to better coverage and higher accuracy. The principle behind these enhancements is better neighborhood selection and weighting similarity measures by the number of items that are co-voted by pairs of users.

Factor Analysis for Collaborative Filtering

Factor Analysis was introduced in the previous section on techniques in Dimensionality reduction. Here, we continue to use the same notation for continuity. To cast the collaborative filtering

scenario in the factor analysis framework, we assume the following: Let the user database for n users and m items be represented by a $m \times n$ matrix $\mathbf{X}$ with columns $\mathbf{x}_i$ corresponding to the profile for user u_i. We assume that each $\mathbf{x}_i$ is a random vector in Eq. (2.6) drawn i. i. d. from a factor analysis model with k factors and (unknown) parameters μ, Λ, and Ψ. We then can use the observed user ratings in $\mathbf{X}$ to learn the parameters of the model, e. g.ũsing maximum likelihood estimation. (Canny, 2002b) has shown how factor analysis can also deal with missing data without the need for imputing values, by using a mixture of factor analyzers (Ghahramani & Hinton, 1997) (discussed in Sec. 3.3.3).

Given sample data which has to be fitted into a Factor Analysis framework, the most commonly used solution is to learn the model parameters using *maximum likelihood estimation* (MLE). A standard approach for performing maximum likelihood estimation in a factor analysis model is the expectation maximization (EM) (Dempster, Laird, & Rubin, 1977) algorithm. In the EM approach, maximum likelihood estimation is performed by maximizing the expected complete data log-likelihood with respect to the parameters of the model, i.e˙ one needs to perform the maximization

$$(\hat{\Lambda}, \hat{\Psi}) = \underset{\Lambda,\Psi}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{E}_{\mathbf{z}} \left[\log p(\mathbf{x}_i, \mathbf{z}; \Lambda, \Psi) \right], \tag{2.17}$$

where the expectation for $\mathbf{z}$ is computed with respect to the posterior distribution of $\mathbf{z}$ given a particular profile $\mathbf{x}_i$. Note that the latter will also depend on the parameters Λ and Ψ, so that both steps, the computation of the posteriors (E-step) and the re-estimation of the parameters (M-step) needs to be alternated until (guaranteed) convergence. The posterior distribution of the $\mathbf{z}$ is a multivariate normal for which the mean vector and co-variance matrix can be calculated as

$$\mathbf{E}[\mathbf{z}|\mathbf{x}] = \langle \beta, \mathbf{x} \rangle, \text{ where } \beta = \Lambda'(\Psi + \Lambda\Lambda')^{-1} \text{ and} \tag{2.18}$$

$$\mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}\right] = \mathbf{I} - \beta\Lambda + \beta\mathbf{x}\mathbf{x}'\beta'. \tag{2.19}$$

Using these, maximizing the expected complete data log-likelihood results in the equations:

$$\Lambda = \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{E}(\mathbf{z}|\mathbf{x}_i)' \right) \left(\sum_{i=1}^{n} \mathbf{E}(\mathbf{z}\mathbf{z}'|\mathbf{x}_i) \right)^{-1} \tag{2.20}$$

$$\Psi = \frac{1}{n} \operatorname{diag} \left[\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i' - \Lambda \mathbf{E}(\mathbf{z}|\mathbf{x}_i) \mathbf{x}_i' \right]. \tag{2.21}$$

A detailed derivation can be found in (Ghahramani & Hinton, 1996).

By replacing $\mathbf{X} = \{\mathbf{x}_i\}_{i=0}^{n}$, we can rewrite the above recurrences compactly using the entire user matrix $\mathbf{X}$, and the entire latent space matrix $\mathbf{z}$, in a readable form as follows:

$$\begin{aligned} \beta &= \Lambda'(\Psi + \Lambda\Lambda')^{-1} \\ \mathbf{Z} &= \beta\mathbf{X} \\ \Lambda^{[t]} &= \mathbf{X}\mathbf{Z}'(\mathbf{Z}\mathbf{Z}' + \Psi(\Psi + \Lambda\Lambda')^{-1})^{-1} \\ \Psi^{[t]} &= \frac{1}{n} \operatorname{diag}(\mathbf{X}\mathbf{X}' - \Lambda^{[t]}\mathbf{Z}\mathbf{X}') \end{aligned} \tag{2.22}$$

Note: We use FA in Sec. 3.3.3 for developing a learning solution for CSP.

Probabilistic Latent Semantic Analysis Model

Latent Semantic Analysis (LSA) (Furnas et al., 1988) is an approach to identify hidden semantic associations from co-occurrence data. It is mostly used in automatic indexing and information retrieval, where LSA usually takes the (high dimensional) vector space representation of documents based on term frequency as a starting point and applies Singular Value Decomposition (SVD) to generate a reduced latent space representation. LSA has been applied with remarkable success in different domains. PLSA (Hofmann, 2003, 2004) is a probabilistic variant of LSA. The core of PLSA is a statistical model which has been called the *aspect* model. The aspect model is a latent variable model for general co-occurrence data which associates a hidden (unobserved) factor variable $\mathbf{z} \in Z = \{z_1, z_2, ..., z_k\}$ with each observation. In the context of information retrieval or document analysis, the observations usually correspond to the occurrences of words in documents. In the context of Collaborative filtering, each observation corresponds to a vote by a user to an item in a collaborative filtering setting. The space of observations is normally represented as an $m \times n$ co-occurrence matrix (in our case) of m items $\mathcal{Y} = \{y_1, y_2, .., y_m\}$ and n users $\mathcal{U} = \{u_1, u_2, .., u_n\}$. The aspect model can be described as a generative model:

- select a data item y from $\mathcal{Y}$ with probability $P(y)$,
- pick a latent factor z with probability $P(z|y)$,
- choose a user u from $\mathcal{U}$ with probability $P(u|z)$.
  As a result we obtain an observed pair (u, y), while the latent factor variable z is discarded.

Translating this process into a joint probability model results in the following

$$P(u, y; z) = \sum_z P(u, y, z) = \sum_z P(y|z)P(z|u)P(u) \tag{2.23}$$

This model is based on the following independence assumption: conditioned on the latent factor z, data item u (user) is assumed to be generated independently of the specified item y. Since in collaborative filtering we are usually interested in predicting the vote for an item for a given user, we are interested in the following conditional model:

$$P(y|x) = \sum_z P(y|z)P(z|x) \tag{2.24}$$

The process of building a model that "explain" a set of observations $(\mathcal{X}, \mathcal{Y})$ can be reduced to the problem of finding values for $P(z), P(y|z), P(u|z)$ that maximize the (log)likelihood $L(\mathcal{X}, \mathcal{Y})$ of the observations, where

$$L^{lg1}(z) = -\frac{1}{N} \sum_{\langle u,y \rangle} \log P(y|u; z) \tag{2.25}$$

Expectation Maximization for Gaussian PLSA

Expectation-Maximization (EM) algorithm is a standard procedure for maximum likelihood estimation in latent variable models. It alternates two steps: (1) an expectation E step where posterior probabilities are computed for latent variables z, based on the current estimates of the parameters, (2) a maximization M step, where parameters are updated for given posterior probabilities computed in the previous E-step. Beginning with some arbitrary values of $P(z)$, $P(y|z)$ and $P(z|u)$, EM algorithm is guaranteed to reach a locally optimal solution.

E-step: We compute the posterior probabilities for each observed rating triple (u, y, v) according to

$$p(z|u, y, v) = \frac{p(z|u)p(v|z, y)}{\sum_{z'} p(z|u)p(v|z, y)} \tag{2.26}$$

M-step, part I: We re-compute the community membership probabilities

$$p(z|u) = \frac{\sum_{(u',y,v):u=u'} p(z|u, y, v)}{\sum_{(u',y,v):u=u'} 1}, \tag{2.27}$$

where the denominator is the number of ratings by user u.

M-step, part II: We re-estimate the Gaussian parameters (means and variances)

$$\mu_{z,y} = \frac{\sum_{(u,y',v):y'=y} p(z|u, y, v)\, v}{\sum_{(u,y',v):y'=y} p(z|u, y, v)} \tag{2.28}$$

$$\sigma^2_{z,y} = \frac{\sum_{(u,y',v):y'=y} p(z|u, y, v)\, (v - \mu_{y,z})^2}{\sum_{(u,y',v):y'=y} p(z|u, y, v)} \tag{2.29}$$

Note: We use PLSA in Sec. 3.3.4 for developing a learning solution for CSP.

2.4.3 Evaluation in Collaborative Filtering

The task of evaluating predictions in collaborative filtering is easily described as the measurement of the *deviation* from observed values. Given that the user database can be compactly represented as a Matrix $\mathbf{X}$, with a user u_i forming a row with m items, the objective is to predict missing values in this matrix. Since only a small percentage of the matrix is observed, a portion of the observed data is artificially removed, and predicted using the remaining values. Generally, the user population is divided into two categories: training users, and test users. Usually all the data of the training users is made available to the algorithm being evaluated, which either learns a statistical model, or pre-computes user similarities. Data is artificially removed from the test set only.

There are a commonly used protocols for data removal from the test set. These include the *All-But-1* and the more general *All-But-n* protocols, where 1 (or n) vote(s) are randomly removed. The prediction task is to then estimate the missing value using the training data and the available data from the test users.

To measure the success of the prediction task, metrics which capture deviation from actual values are used. These include the *mean* and *root mean error*. An additional metric called the *ranking score* rates the ranking generated by the predicted user votes.

1. *Mean Average Error* $= \frac{1}{m}\sum_v |p_v - a_v|$, where p_v is the predicted vote, a_v is the actual vote, and m is the number of votes over which MAE is being calculated. The average is taken only over known values (assume the active user has provided m votes).Normalization may also be done by dividing the MAE by the scale of the rating (which is $rating_{max} - rating_{min}$): this metric is called the *Normalized Mean Average Error* (NMAE).

2. *Root Mean Average Error* $= \sqrt{\frac{1}{m}\sum_v |p_v - a_v|^2}$, where p_v is the predicted vote, a_v is the actual vote, and m is the number of votes over which RMSE is being calculated. The average is taken only over known values (assume the active user has provided m votes). This metric is useful in finding out the ability of a CF algorithm to generalize and highlights larger errors.

3. *Ranking score of top-20 items* $= R_{score} = \frac{100 * \sum R}{\sum R_{max}}$. This metric gives a value between 0 and 100; it was introduced by (Breese et al., 1998) and remains popular in the recommender systems community. The rank score R is defined using a ranking function τ, where $\tau(y)$ provides the rank for the item y. The top ranked item gets $\tau = 1$, the next as $\tau = 2$ etc. Using this definition, the ranking score is defined as

$$R(u,\tau) = \sum_{<u',v,y>:u'=u} 2^{-\frac{\tau(y)-1}{\alpha-1}} \cdot max(v-\overline{v},0) \tag{2.30}$$

where α is a decay factor, typically set to 5, and $\overline{v}$ is the mean vote for the user u. R_{max} is the ranking score achieved using the actual top-n votes of the user, typically using all available data. Higher values indicate a ranking with top items as the most highly rated ones. One big advantage of this metric is that it gives the same score for permutations of items with the same score. Thus if a user has rated 5 items with the maximum score 5, then the R_{score} is the same for any permutation of the ranking. This takes away the problem of breaking ties.

2.4.4 Privacy in Collaborative Filtering

Collaborative Filtering is one of the most successful mechanisms for generating recommendations. Widespread adoption of CF technology has lead to large data collection about users which companies often consider as a valuable asset. However, the current technology is a serious threat to individual privacy. Most online vendors collect buying information about their customers, and make reasonable efforts to keep this data private. However, customer data is routinely leaked either as stolen/auctioned old hardware, or via unindented public exposure (e. g. the AOL query log). Recent work (Frankowski, Cosley, Sen, Terveen, & Riedl, 2006) describes the privacy risk that users face of being identified in a collaborative filtering setting, even among hundreds of thousands of other users, when user data is released in the public domain. A second disadvantage is that server-based systems encourage monopolies. There are correlations between customer purchase choices across product domains. So companies that can acquire preference data for many users in one product domain have a considerable advantage when entering another one.

Privacy in Recommender systems is a topic which has been studied extensively in the last few years. Canny's work (Canny, 2002b, 2002a) in privacy preserving collaborative algorithms is the most well known approach for this problem. Canny proposes a peer to peer model-based approach for computing recommendations in a scenario where each user has access to his profile and no central server is involved. Canny argues that a decentralized approach has the advantage that user data is not available to any single person or system, however data still has to be communicated between such nodes to compute an effective model. Therefore any message exchange between peers should also be protected. Canny proposes the use of encryption on these individual messages, with model computation done using encrypted data. Due to homomorphic properties of encryption schemes like RSA, operations like addition can be performed even if only encrypted data is available. Encrypted contributions from all users can be combined meaningfully in the form of a model which can be decrypted and made available to all peers. This model can be used by every peer to compute its own recommendations. We follow this idea as well in the development of a protocol to computing a CSP solution with distributed users in Sec. 3.3.4.

2.4.5 Trust in Collaborative Filtering

Collaborative Filtering systems are essentially social systems which base their recommendation on the judgment of a large number of people. However, like other social systems, they are also vulnerable to manipulation by malicious social elements. *Lies and Propaganda* may be spread by a malicious user who may have an interest in promoting an item, or downplaying the popularity of another one. By doing this systematically, with either multiple identities, or by involving more people, a few malicious user votes and profiles can be injected into a collaborative recommender system. This can significantly affect the robustness of a system or algorithm, as has been studied in recent work (Lam & Riedl, 2004; M. O'Mahony, Hurley, Kushmerick, & Silvestre, 2004).

The study of attack models and detection in recommender systems is fairly recent as the relevance of such attacks has increased rapidly. One recent example is when a loosely organized group who did not like evangelist Pat Robertson managed to trick the Amazon recommender into linking his book *Six Steps to a Spiritual Life* with a book on sex for men[10]. Collaborative filtering technology is being widely used on the web as an approach to information filtering and recommendation by commercial service providers like Amazon and Yahoo!. For malicious attackers, or a group interested in popularizing their product, there is an incentive in biasing the collaborative filtering technology to their advantage. Since collaborative filtering is based on social networking, it is also vulnerable to social attacks, i.eà group of users working together to bias the system. A lot of electronic systems, especially web-enabled ones provide free access to users via simple registration processes. This can be exploited by attackers to create multiple identities for the *same* system and insert ratings in a manner that manipulates the system. *Profile injection attacks* add a few profiles (say 3% of the total profiles) which need to be identified and protected against. Such attacks have been refered to as *shilling* attacks, while we see this as a specific form of spam. Further, profile injection attacks can be classified in two basic categories: inserting malicious profiles which rate a particular item highly are called *push* attacks, while inserting malicious profiles aimed at downgrading the popularity of an item are called *nuke* attacks (M. O'Mahony et al., 2004).

Research in the area of *shilling attacks* (M. O'Mahony et al., 2004) has made significant advances in the last couple of years. Early work identified the threat of shilling attacks and the types of attack (*nuke* and *push*). Various attack strategies were then discovered and appropriate metrics were developed to measure the effectiveness of an attack. Attack strategies include (Mobasher, Burke, Williams, & Bhaumik, 2005):

1. *Random attacks*, where a subset of items is rated randomly around the overall mean vote.
2. *Average attacks*, where a subset of items is rated randomly around the mean vote of every item
3. *Bandwagon attacks*, where a subset of items is rated randomly around the overall mean vote, and some highly popular items are rated with the maximum vote.

Note that Gaussian distributions ($\mathcal{N}_{\mu,\sigma}$) have been used for generating the random votes rather than the uniform random distribution. This implies that attack profiles have votes near, or equal to the mean vote with a very high probability. Also, standard deviation of the complete set of votes is used for random and bandwagon attacks, while the standard deviation of the each individual item is used for the average attack.

The most commonly used metric for measuring the effect of shilling attacks is *prediction shift* (M. O'Mahony et al., 2004), which models the difference between average predicted rate

[10]The news story is at http://news.com.com/2100-1023-976435.html.

of the targeted item, before and after the attack. It is defined as the difference in the predicted value of an item before and after a shilling attack.

$$\mathcal{P} = \sum_{u} \hat{v}_{u_i,y} - v_{u_i,y} = \sum_{u} \mathcal{P}_u, \tag{2.31}$$

where $\hat{v}_{u_i,y}$ denotes the predicted value of item y for user u_i after an attack and $\mathcal{P}_u$ denotes the prediction shift in user u, and $v_{u_i,y}$ is the predicted value without attack profiles inserted. Thus the aim of the shilling user s is to maximize the prediction shift $\mathcal{P}$.Using this metric, it was discovered that *Average* attacks are more powerful than *Random* attacks (Mobasher et al., 2005). Further, it was discovered that k-NN based algorithms for collaborative filtering (e. g. based on Pearson's correlation) were very vulnerable towards such shilling attacks.

In the last couple of years, research in this area has focused on detection of shilling attacks. It was discovered that item based recommendation algorithms, which measure similarity between items rather than users (B. M. Sarwar, Karypis, Konstan, & Riedl, 2001; Mobasher et al., 2005) were more robust to such manipulations. However, newer attack models like bandwagon attacks and segment attacks were quite successful against item based recommendation algorithms. The earliest spam detection algorithm based on features of spam profiles was invented by Chirita et al. (Chirita, Nejdl, & Zamfir, 2005). While this algorithm was successful in detecting shilling attacks with dense attacker profiles, it was unsuccessful against attacks, which are small in size or have high sparsity. Mobasher et al. (Mobasher et al., 2005) compare their feature-based classification algorithm which performs significantly better than the Chirita et al. (Chirita et al., 2005) algorithm by taking more features into account. The Mobasher et al. algorithm trains a classifier given example spam and authentic profiles and is fairly accurate in detecting spam attacks of varying sizes and density. Two disadvantages of their approach come to mind: firstly, a supervised approach needs a large number of examples, and can detect only profiles similar to the examples profiles. Secondly, these algorithms perform badly when the spam profiles are obfuscated. Adding noise, shifting targets, or shifting all user ratings differently makes the attack profiles more difficult to detect for existing feature based detection algorithms. Williams et al. (Williams, Mobasher, Burke, Sandvig, & Bhaumik, 2006) discusses these obfuscation strategies and their effect on detection precision. Recent algorithms (M. P. O'Mahony, Hurley, & Silvestre, 2006) have taken up more principled approaches where signal processing theory is used to detect noise which is artificially inserted into the data. We find this direction promising, however the accuracy remains low (15-25%). One principle reason for the low accuracy of this and other approaches that all existing algorithms consider users indivually, instead of looking at the collective effect.

The current state w.r.t. shilling detection is that feature based classifiers can be learnt and used to classify users as trusted or untrusted. These algorithms work by correctly identifying the goal of shillers to affect the predicted vote for the attacked item, and identifying profiles which affect the predicted vote significantly. However, shilling profiles can now be constructed such that the obvious signatures which make them stand out are masked, at the cost of lesser impact. This is like a 2-player game, with the recommender system on one side against the attackers. Current algorithms fail in detecting obfuscated (Williams et al., 2006) and small attacks and have to evolve in order to win this game.

2.5 Final comments on the Literature Survey

This chapter represents the most relevant topics to the problem of cross system personalization. Since each of the areas mentioned is an active topic of research, there is a growing body of related

work; a case in example is the area of collaborative filtering algorithms. We have concentrated on the best performing, or the best known approaches in such cases; there are several other approaches which we are aware of, however, these are not mentioned. Also, while giving a survey of commercial personalization servers, we describe those which are well known, or for which technical details are available. Several others, like the one used by Amazon, while more popular, are not described in enough detail by the developers. Also, for very recent servers like CleverSet[11] and Rocketinfo[12], technical details are unavailable; hence we do not mention them as related work.

[11] http://www.cleverset.com
[12] http://www.rocketinfo.com

3 Conceptual Model and Methods

A problem worthy of attack proves its worth by hitting back.

(Paul Erdos)

Cross system personalization(CSP) allows for sharing information across different information systems in a user-centric way and can overcome the problem of using distributed user data. Information about users, which is originally scattered across multiple systems, is combined to obtain maximum leverage and reuse of information. We have identified two principal approaches for enabling cross system personalization: a *semantic* approach and a *learning* approach.

The *semantic* approach to CSP relies on understanding the meaning of user data; such approaches are commonly adopted by researchers (Davies, Fensel, & Van Harmelen, 2003) in the area of *Semantic Web* . The base of this approach is a generic user model which can be used under a large variety of circumstances called the Unified User Context Model (UUCM). The UUCM is a meta model using which can describe other detailed/specific models user models. However the vocabulary of the UUCM is shared, meaning that different models defined using the UUCM are still partially inter-operable. Systems can chose to use preexisting ones or define a set of aspects from their own which can be partially mapped to existing aspects. Users maintain a large user profile on their side which is in the UUCM format and can be used by many different systems. There is an active agent on the users' side which interacts with systems via an open protocol and negotiates on the aspects of the UUCM which both the system and agent can understand. This process can be used with many different systems and their feedback combined to update aspects of the user profile which are reflected in the usage across all systems.

The *learning* approach relies on detection of patterns across many users who use multiple systems to predict profiles of users at one system given profile(s) of the same users on other systems. This method uses example user profile data to learn mappings between profile formats to enable cross system personalization without the need to rely on accepted semantic standards or ontologies. The key idea is that one can try to learn dependencies between profiles maintained within one system and profiles maintained within a second system based on data provided by users who use both systems and who are willing to share their profiles across systems. Here, instead of requiring a common semantic framework, it is only required that a sufficient number of users cross between systems and that there is enough regularity among users that one can learn within a user population, a fact that is commonly exploited in social or *collaborative filtering*.

In this chapter, we outline both of these approaches, pointing out the pros and cons of each of them, and how to use the concept of a decentralized unified user profile which acts as a *Passport* identifying users during their journey in information space. Towards the end, we also

present some contributions to collaborative filtering in general: we propose novel unsupervised algorithms to detect shilling attacks which outperform existing detection approaches. We also propose a robust collaborative filtering algorithm which offers partial resistance to spam.

3.1 A Semantic Approach to Cross System Personalization

Humans are often better than computers in dealing with situations where there is no precedent: also, they can think beyond syntax. A human can look at a document which states the annual salary of a person as $20 and understand that this does not make sense, while a computer would not identify this. This knowledge, which humans understand as *common sense* is missing in computers and thus limits how far computer programs can go. In recognizing this problem, researchers set out to codify the common sense knowledge possessed by an average person into a rule based framework, which a computer program can understand. Semantic reasoning is thus the area of computer science which deals with knowledge representation in a manner that can be automatically exploited by a software agent. By expressing knowledge in a formal manner based on *predicate logic*, traditional logic techniques can be used to reason and infer derived knowledge. This vision has been extended to the web by Sir Tim Berners-Lee, the inventor of the WWW, who has coined the term *Semantic Web* for an intelligent form of the WWW as a universal medium for data, information, and knowledge exchange.

Clearly, how to represent knowledge is an important part of the reasoning process. Casting the Cross system personalization problem in the Semantic Web context, representation of user data in a standard machine understandable format is important. Formats like *Resource Description Framework* (RDF) and *Web Ontology Language* (OWL) are intended to formally describe concepts, terms, and relationships within a given problem domain. In a user centric world, we would aim to describe in detail all concepts and relationships that a user is part of. Clearly, there is the need to define a large vocabulary to model heterogeneous users who may have different interests and skills. Moreover, different systems that a user interacts with may model different aspects; therefore one needs a framework where many aspects of a variety of users can be expressed. This framework should also be extensible to incorporate new systems or new aspects of a person. Using logic based representation techniques, we expect to be able to reason with rules defined on concepts used by the framework. In the next subsection, we introduce the UUCM as representation framework for modeling users in an extensible fashion.

3.1.1 The Unified User Context Model

We now introduce the core of our semantic solution to CSP in the form a unified user model, first described by us in (Niederée et al., 2004). The *Unified User Context Model* (UUCM) is an extensible user model that can be used for modeling characteristics of the user and his situation, i. e. the user context, along different dimensions. For this purpose, a vocabulary for the description of domain-model and ontology-based user profiles was developed and is represented as an RDF Schema. Existing user profile-related standards and proprietary user profile formats provided input for the adequate design of the vocabulary. The construction of ontology-based user profiles, taking into account domain relationships, goes beyond the state of the art of describing user profiles (with respect to interests and preferences).

The extensible set of UUCM facets describes not only the characteristics of the users themselves (like interests and skills), but also aspects of the users' situation and environment. UUCM incorporates several dimensions such as Task, Relationship, and Cognitive Patterns. This

supports flexibly modeling aspects of the user like the tasks of a user and information objects related to the user. For the interpretation of user profiles based on the UUCM, we rely on Semantic Web Technologies for the representation of the user context model as well as the concrete user context profiles. The UUCM can not only be used to represent a user context model within a system, but also provides an intermediate format to exchange user profiles between legacy personalization systems.

Two levels are distinguished in our approach for unified user context modeling. On the abstract level, the basic building blocks for the UUCM are defined as follows: *user context*, *user model facets*, core properties for *facet description*, and user model *dimensions*. We use the term *facet* here to represent the different characteristics of the user. This level defines a meta-model for the concrete dimensions and facets used in the description of the user context model. For the cross-system personalization approach that we are aiming for, it is assumed that this user context *meta-model* is published as a shared ontology and all participating systems rely on this model.

On the concrete level, an extensible set of UUCM dimensions and facets is defined. This is not restricted to just users' interests, but also includes tasks and relations to other entities in the information space and respective user communities. UUCM facets and dimensions are described as part of an additional ontology that is shared by the components committing to the UUCM. The UUCM meta-model, thus, can be combined with different UUCM facet and dimension ontologies to form concrete user context models that provide the schema for the construction of user context profiles. This is supported by the fact that the UUCM is encoded as an RDF Schema augmented with OWL (OWL, 2004) expressions. This technology enables simple exchange within the (Semantic) Web context, reasoning over user characteristics for value adding services and URIs provide a systematic support for the qualification of facets and facet values.

Structure of the Unified User Context Model

The structure of the UUCM is summarized in Fig. 3.1. A simple but flexible and extensible way of modeling the different facets of the user can be accomplished by the use of name/value pairs (cf. modeling of context by parameter value pairs in (Benerecetti, Bouquet, & Ghidini, 2000)). Following this approach, a name/value pair is used to capture each facet of the user context model (e. g. user preferences) and new facets can be easily added.

The UUCM can be exploited in an open or cross-system environment by binding facet names and values to vocabularies or ontologies, easing interpretation of user profiles in a global context. In summary, each UUCM facet is described by the following properties:

facet name: Name of the UUCM facet to be described;

facet qualifier: A qualifier for the facet itself; this qualifier can be used to bind the facet to a defining vocabulary;

facet value: The values of the facet; The structure of the value may depend on the respective facet. In the general case, it is a value or a reference to another resource. (The use of the term resource is comparable to that of RDF [RDF/XML].)

value qualifier: A qualifier for the value(s) of the facet; this qualifier can be used to bind the value of the facet to a defining vocabulary (in contrast to the facet qualifier that qualifies the facet itself); In the case of a UUCM facet area-of-interest, for example, one might state that the ACM classification schema is used to specify the user's research interests.

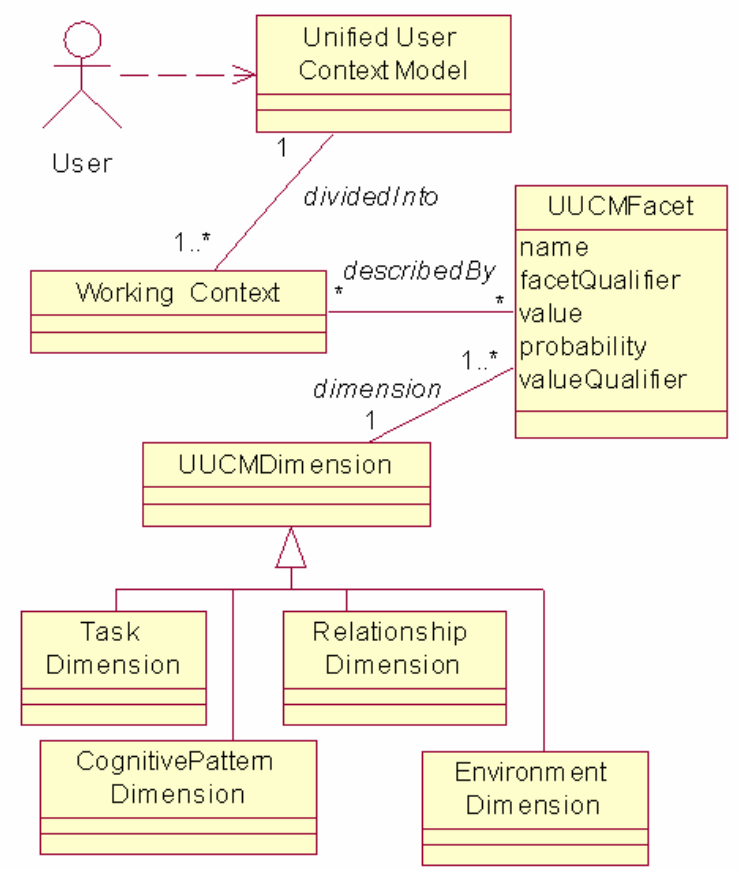

Figure 3.1: Building Blocks of the UUCM

value probability: A weight reflecting the probability of the facet value; this property of a facet can be used to express the reliability of facet values that are computed from analyzing user behavior.

facet dimension: Each facet is assigned to one of the dimensions covered by the UUCM; for example, the UUCM facet *area-of-interest* can be modeled as part of the UUCM cognitive pattern dimension. Two aspects are used for structuring the UUCM. First, the UUCM is structured into several working contexts taking into account the different tasks and roles of a user. Secondly, the UUCM is structured along a set of UUCM dimensions that are discussed in more detail below. The UUCM structure can, thus, be summarized as follows: A user profile is divided into a set of working contexts, where each context is described by a set of UUCM facets. Each facet is assigned to one of the UUCM dimensions. We encode the UUCM as an RDF Schema augmented with OWL expressions.

User Context Dimensions and Facets

The UUCM just defines the principled way in which a user context profile is described and structured. However, for the description of concrete user context profiles, the UUCM relies on ontologies (or vocabularies) for the UUCM dimensions, the UUCM facets and for the facet values.

As a starting point, we pick 4 dimensions along which a user can be modeled: the *Cognitive Patterns*, *Task*, *Relationship*, and finally, the *Environment* dimension. The selection of the dimensions is based on user models in existing personalization approaches and on context

modeling approaches. For each of the UUCM dimensions, our UUCM facet ontology defines a set of UUCM facets, which describe the aspects of the respective UUCM dimension. These facets are presented together with the respective dimensions in the following. However, the UUCM is independent of the selected facet and dimension ontology. It can be combined with other sets of facets and dimensions. It is not the goal of this work to give the ultimate set of facets and dimensions, but to discuss a flexible and extensible approach for unified user context modeling.

Within the facet ontology, the concrete facets are defined as subclasses of the general class `UUCMFacet` defined in the UUCM (see Fig. 3.1). They inherit all properties of the class `UUCMFacet` and define facet-specific restrictions like e. g. for the types of resources that are valid facet values. With this approach, there is a large flexibility with respect to which aspects are fixed for all instances of one facet (e. g. the facet name) and which can be selected individually for each facet instance (e. g. the value qualifier, if one wants to allow the use of values from different vocabularies). An alternative modeling approach is to make all facets instances of the general class `UUCMFacet`. This, however, gives fewer options for a systematic definition of specific types of facets.

The Cognitive Patterns Dimension The cognitive patterns dimension describes cognitive characteristics of the user. It contains the facets that are traditionally used in personalization approaches. Based on an analysis of existing personalization approaches, we selected the following facets to be included into our facet ontology:

- The facet *areas-of-interest* describing the interests of a user typically based on a controlled vocabulary or ontology of subjects (specified by the value qualifier).
- The facet *competence* with two facet subclasses *skill* and *expertise*.
- The facet *preference* that can be used to model preferences of the user.

Each of these facets may have several values. In this case the same facet is contained several times in the user context profile. Alternatively one may also enable the use of multi-value facets within the facet ontology. However in this case, all values have to share the same value qualifier.

The Task Dimension When interacting with an information system, the user is involved in a task that determines his/her information needs and the goals of the performed activities. Tasks are described in (domain-specific) task models that structure tasks into subclass hierarchies. The user profile may refer to such task models.

The following useful facets for the task dimension have been identified:

- *Current Task*: This facet describes the task the user is currently involved in. This facet has a facet qualifier referring to a task model description, and the value qualifier refers to a concrete domain task model, whereas the facet value is a reference to a concrete task instance based on this task model. Using this approach, any appropriately described task model can be fitted into the UUCM.
- *Task Role*: This facet describes the role of the modeled user in the current task. This facet has a value qualifier referring to an ontology of roles in the current task domain, and the facet value refers to a node in the chosen ontology.
- *Task History*: This facet points to a history of tasks completed so far within the current working context. The task history helps to keep track of completed tasks and subtasks. This facet again is based on a task model (typically the same as the current task) and refers to a sequence of interrelated tasks.

Further task properties can also be considered for inclusion in the set of facets of the Task Dimensions. Since considerable work has been done in task modeling (Motta, 1999; Schreiber et al., 2000), the challenge here is not to identify adequate properties to describe tasks, but to decide, which of these properties are required as integral parts of the user context model.

The Relationship Dimension The requirements and information needs of a user are also determined by the entities the user is related to. Therefore, the facets of the relationship dimension are based on the relationships the user is involved in. These facets are thus based on one (or more) relationship type ontology (for e. g. example relationship ontologies from the scientific research community domain). The facet names are names of relationship types, the facet qualifier points to the respective relationship ontology and the facet value refers to the resource the user is related to via this relationship. The value probability, finally, gives a probability for the existence of this relationship.

The Environment Dimension The environment dimension refers to those parameters which are typically used for context-awareness approaches. Facets like current time, location, device, language etc are parameters which influence and, thus, are important in understanding the interaction between the user and the computer. These aspects are also important in understanding the user's changing requirements in different scenarios. These facets include:

- *Time*: Every working context would be valid in a certain time frame
- *Location*: This facet refers to the physical location of the user
- *Device*: The device the user is using, e. g. PC, PDA, etc.
- *Language*: The language of choice for the user

These are only the most central facets of this dimension. Many other facets describing the environment might be important depending upon the specific application. However, the environment dimension is not in the focus of our work. We rely on existing and upcoming work in this area.

Working Context, and Context-of-Use In principle, the user context can be described by a large set of facets. However, the user interacts with systems in different roles and is involved in different tasks in parallel, each of which is associated with a specific subset of the user context facets. Therefore, to reflect this structuring the user context is divided into multiple working contexts, grouping together user context facets that are related to and relevant for the same task and/or role of the user.

While accessing an information system and performing an activity to complete a task, a part of the current working context is extracted based on the relevance of the working context's facets for the planned activity (or activities). This subset of the working context is called the context-of-use.

Illustrative Example

As an example, lets consider the case of two popular Instant Messaging Applications: MSN[1] and Yahoo messenger[2]. The profile of a user in these applications contains a list of people that

[1] http://messenger.msn.com
[2] http://messenger.yahoo.com

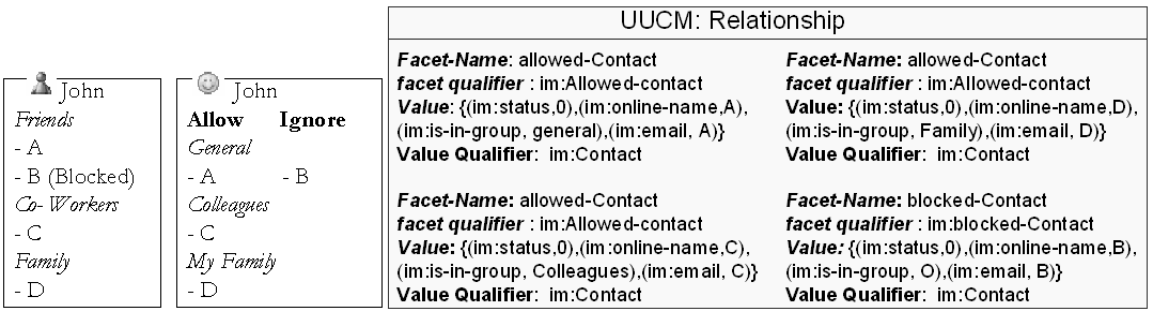

Figure 3.2: (a) John's Profile for MSN and Yahoo (b) John's Yahoo Profile in UUCM

this user can directly contact. Therefore, the profiles of these applications focus on modeling relationship aspects. However, both of these applications have a different structure to model this user information. MSN Messenger allows user to have custom names, but has predefined categories for classifying contacts (Friends, Family, co-workers etc). Yahoo Messenger does not allow custom names, but allows creation of new categories. Furthermore, with MSN Messenger, it is possible to have a person on your list, yet to 'block' them from contacting you. Yahoo has different structure, and to 'block' or 'ignore' a contact, this person has to be deleted from the list and then added to a separate 'ignore list'. Therefore, it is obvious that even for similar systems in the same domain, there are variances in how the same or similar knowledge about the user is modeled.

In this case, user profile formats like PAPI (IEEE, 2000) and IMS (Colin Smythe & Robson, 2001) will fail to store the profile completely. Thus, to make these systems inter-operable, a more powerful unified user model is required. For the messaging domain, cross-over applications like Trillian[3] or Gaim[4], which connect to both these networks, in addition to others like AOL and ICQ, could benefit from the possibility of storing all profiles in one common format or of translating the different user profile format into a shared format.

Lets take up the example of a user *John* who uses both the MSN and Yahoo Messenger. Essentially the user profiles of these IM networks are lists of 'contacts' that a person is explicitly connected with. While the communication protocols and networks influence the behavior, the basic usages and the required user information remains much the same. However the different structure of the user profiles means they have to be represented in different formats. Concretely, let John's MSN and Yahoo profiles be as described in Fig. 3.2 (a).

Both these models use common concepts, which can be represented by a common vocabulary consisting of concepts *im:Contact, im:Contact-List, im:Group* related by relationships, as shown in Fig. 3.2(a). Analyzing the domain for IM user profiles, we reach a common model shown in Fig. 3.2(b), composed of only two facets. Please note that the same common model can be used for AOL, ICQ and other IM applications with minor modifications to the common vocabulary. Thus this model represents the domain model for the IM domain. Further additions to the model are possible, but for this example, we assume the profile to be composed of only a categorized list of contacts. Using this understanding, we can represent both the profiles in the UUCM format. Fig. 3.2(b) shows how the Yahoo profile can be represented. The MSN profile can be similarly represented. We note that the profiles for these applications lie completely in the Relationship dimension. Similar models can be constructed for eCommerce websites and

[3]http://www.trillian.cc/
[4]http://gaim.sourceforge.net/

Figure 3.3: The Context Passport conceptual architecture

personalized content providers.

By using a common format for representing users, these two applications can more easily interoperate, and one application can connect to both networks while maintaining a common profile. In a scenario where all the networks interoperate, one common profile could abstract which network a contact actually belonged to. The UUCM enables such a unified user profile and hence eases interoperatibility.

3.1.2 The Context Passport Metaphor

There are three objectives which Cross System Personalization needs to address. These are a) broader user models that can cope with the variety in user modeling, b) handling heterogeneous personalization systems and approaches, and c) giving more control to the user. In line with these objectives, we claim that user profiles should be stored closer to the user. However, maintaining user profiles on the user's side presents some challenges. Interacting with multiple information systems may lead to a large amount of interaction data. As a result, the unified user profile may become quite large, and transferring the entire user profile data when interacting with an Information System may be impractical. The first step of reducing this amount is interpretation of this data to extract user characteristics from it. Since the individual system best understands the local interactions, this should be done within the individual personalization engine and only higher level descriptions of users should be exchanged between the information system and the unified user profile. In (Niederée et al., 2004), we introduced the concept of a *Context Passport* which accompanies the user in his/her travel through information space. The Context Passport is a unified user profile based on the multi-dimensional user context model (UUCM), which can be used with multiple systems.

The *Context Passport* is a compact representation of the user's current context for cross system personalization. It also contains the activities chosen by the user to be performed in order to fulfill the tasks allotted. It contains ontologically-arranged information about the user's current tasks and related activities, the user's cognitive patterns (skills, area-of-interest etc), the environment (time, place, device used), and the user's personal web of the people and relationships involved, following the UUCM model for user context modeling. In order to use the context passport, the user takes the context passport and *presents* it to an information system (IS) that the user wants to use. Since the context passport is bound to a shared ontology, there is a chance the IS can partially interpret the context passport using a mediator architecture. As a result of this partial interpretation, two flows of information are possible: one from the context passport to the IS, and secondly from the IS to the context passport. The first flow helps the IS

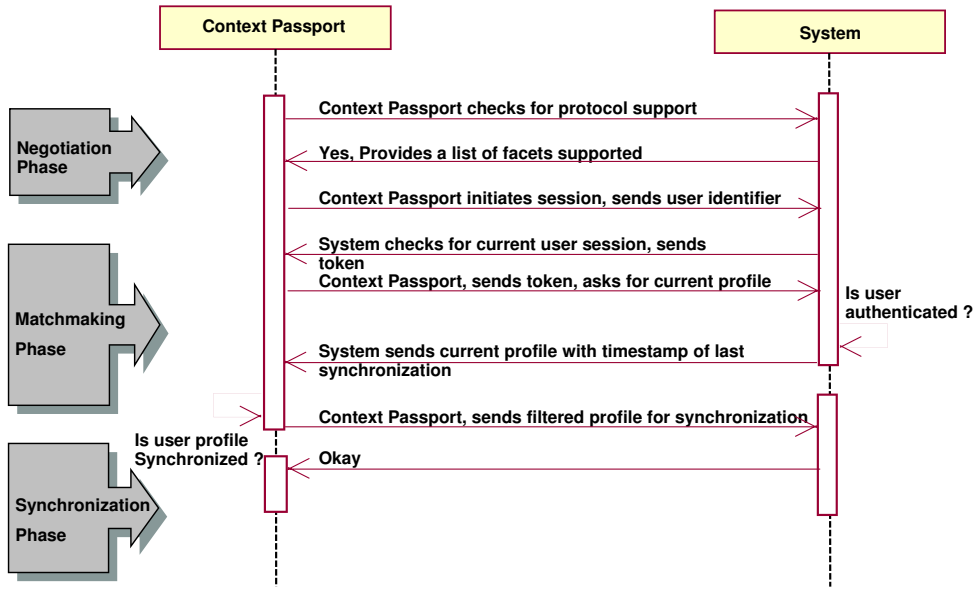

Figure 3.4: Cross System Communication Protocol

to better understand what the user requires from the IS, since the context passport refers to the task model, activities and also other information about the user context model. The second flow arises due to the interaction between the IS and the user which changes the state of the user's context. The purpose of this flow is to update the context passport with the feedback from the interaction.

3.1.3 The Cross System Communication Protocol

For the Context Passport to communicate with an information system (see Fig. 3.3), a standardized protocol is required. This protocol should allow negotiations on behalf of the user for sending relevant user information, and also allow synchronization with the existing user profile within the system. With these objectives in mind, we now describe the Cross-System Communication Protocol (CSCP).

As shown in Fig. 3.1.3, the CSCP protocol operates in three phases a) negotiation phase, b) personalization phase, and c) synchronization phase. In the negotiation phase, the Context Passport and the system agree on activities to be performed and information to be exchanged. The main goal to be achieved is a common understanding on the type of information that the other system can understand and use. In our approach, the UUCM provides the common vocabulary for negotiating about the available user information (dimensions, facets about the user, etc.) In order to perform an automatic negotiation about what activities can be supported, there needs to be an agreement on a machine understandable common vocabulary. An ontology of a particular domain (say Travel) can provide this common vocabulary for this purpose. After an agreement has been reached on the activity to be performed and the available user information, the Context Passport needs to extract information relevant to this activity (context selection). There is also a simpler possibility of simply exchanging the facets in Context Passport, and the facets supported by the end system. However, this would limit the transfer of information from one system to another if they do not use the same set of facets to describe a used. This is

Figure 3.5: Context Passport as an Internet Explorer Toolbar

communicated to the system in the personalization phase of the CSCP. After the activity has been performed on the system and the user has completed the related transactions (booked some tickets, shown interest in some other offers, etc), the information system has a slightly changed understanding of the user. Finally in the synchronization phase, the system changes the user's profile on its side and synchronizes these changes with the Context Passport. Thus, the Context Passport is kept up-to-date.

In an online environment, the Context Passport can be a browser toolbar which stores information on the client side in a data format using the UUCM ontology. In order to communicate with the Context Passport, existing systems have to understand the CSCP protocol and the UUCM as a part of it. This requires different architectures depending on the underlying system. An adaptor will be required for systems that do not inherently have a personalization engine (e. g. Google). A wrapper for Google, for example, could also do query translation in accordance with the user context. System providers can also choose to support CSCP and interface it with the personalization engine built in. New systems built can support CSCP natively (much like P3P/compliant websites) and use the user context information in their personalization engines. However the essence of the CSCP remains the same in all these scenarios, whether a system needs an adaptor, or can support CSCP directly.

3.1.4 Implementation

In our test bed, initial versions of CSCP and Context Passport have been implemented. One practical aspect in addition to the discussed protocol that came up was authentication. In order for the system to uniquely identify the user that the Context Passport is interacting on behalf of, authentication is necessary. Therefore in the first part of the protocol, the Context Passport provides the authentication token and then proceeds to the negotiation procedure. In order to support different systems, a small identity management component has been implemented which maps the user identifier (GUID) to different authentication tokens used by the user (e.g pairs of systems and login/password).

The Context passport, which is implemented as a browser toolbar, reads the UUCM-based user profile and gets input from the identity management component to interact with different systems in an online scenario. More over, the component is designed to take into account that multiple users may use the same computer, so the context passport further loads the profile relevant to the currently logged-on user.

CSCP is an XML encoded protocol which functions similarly to stateful conversational protocols like SMTP. Most popular protocols like SMTP, FTP, POP, LDAP etc. have their own daemons running on pre-specified ports. For security reasons however, a lot of online systems keep only port 80 available for HTTP based interaction. In this scenario, and inspired by the success of Web services and the ease of use of WS development tools, the CSCP is implemented as a set of stateful web-services. Although statefulness is not a feature of web

services, persistence layers based on databases can be used to maintain state. The advantage of using a web-service based interface is platform independence, as well as easy migration to other scenarios like on the GRID. For e. g. next generation libraries could be service-oriented and cross-service personalization can work in the same way as described for cross-system personalization. In addition to the toolbar and CSCP implementation, two test systems have been developed which implement a book store and a movie store respectively and also have a simple personalization engine built-in. These systems support the CSCP protocol natively as a set of web-services which plug into the personalization engine within these systems. Using a simple categorization scheme, one system is used to build up the user profile. The other system is then able to synchronize its empty profile with the existing user profile and to immediately provide recommendations relevant to the user requirements.

3.1.5 Discussion and Conclusion

Clearly, user models and profile formats in different systems tend to be different in how they model a user. For multiple systems to be able to inter-operate, a common vocabulary has to be used which requires a standardization process. This process might define a controlled vocabulary, e.g. default facet names, or the range of values a facet can take. While individual systems might still define their own facets, they would be required to make mapping available to existing facets. However, a minimum level of compliance is necessary, and a set of standards has to be maintained and enforced by a standards body (e. g. W3C).

Another drawback of this method is misinterpretation of a facet containing data about the user which is interpreted differently from various systems e. g. a movie provider using the facet *comedy* for storing all movie preferences. Clearly, this could lead to false information becoming a part of the user profile. Using an aggregate of user profiles can be used to sort out such inconsistencies automatically: this is the key idea in the next section which introduces CSP based on machine learning.

3.2 A Learning Approach to Cross System Personalization

One of the objectives of intelligent computing to enable machines to learn from their experience and improve over time, much like humans do over their lifetime. This vision lead to the development of Artificial Intelligence, which simulates the human intelligence by enabling computers programs to recognize patterns over time and learn to distinguish between them. Recent emphasis in AI to use probability theory and statistics has spawned the field of *Machine learning*. Techniques in Machine Learning focus on *learning* from example data and to infer rules and identify hidden patterns. When applied to personalization, the objective would be to learn underlying categories of users and how this can be used to predict their interests.

In the context of CSP, machine learning techniques can be used to overcome the drawbacks of the lack of semantic standards of ontologies to understand user profiles maintained by different systems. Instead, one can try to learn dependencies between profiles maintained within one system and profiles maintained within a second system based on data provided by users who use both systems and who are willing to share their profiles across systems – which we assume is in the interest of the user. The major requirement for a learning machine is plenty of example data; in our setting, this would correspond to users who use multiple systems, and are willing to make this information available. Given that there is enough regularity among users that one can learn within a user population, a fact that is commonly exploited in social or *collaborative*

filtering (Konstan et al., 1997).

For simplicity, we consider a two system scenario in which there are only two sites or systems denoted by A and B that perform some sort of personalization and maintain separate profiles of their users; generalization to an arbitrary number of systems is relatively straightforward and is discussed later. We assume that there is a certain number of N_c common users that are known to both systems. For simplification, we assume that the user profiles for a user u_i are stored as vectors $\mathbf{x}_i \in \mathcal{X} \subseteq \mathbb{R}^n$ and $\mathbf{y}_i \in \mathcal{Y} \subseteq \mathbb{R}^m$ for systems A and B, respectively. Given the profile $\mathbf{x}_i$ of a user i in system A, the objective is to find the profile $\mathbf{y}_i$ of the same user in system B, so formally we are looking to find a mapping

$$F_{AB} : \mathbb{R}^n \to \mathbb{R}^m, \qquad \text{s.t.} \quad F_{AB}(\mathbf{x}_i) \approx \mathbf{y}_i \tag{3.1}$$

for users u_i. Notice that if users exist for which profiles in both systems are known, i.e˙ a training set $\{(\mathbf{x}_i, \mathbf{y}_i) : i = 1, \ldots, l\}$, then this amounts to a standard supervised learning problem. However, regression problems typically only involve a single (real-valued) response variable, whereas here the function F_{AB} that needs to be learned is *vector-valued*. In fact, if profiles store say rating information about products or items at a site, then the dimensionality of the output can be significant (e. g.i̇n the tens of thousands). Moreover, notice that we expect the outputs to be highly correlated in such a case, a crucial fact that is exploited by recommender systems. For computational reasons, it is inefficient and often impractical to learn independent regression functions for each profile component. Moreover, ignoring inter-dependencies can seriously deteriorate the prediction accuracy that is possible when taking such correlations into account. Lastly, one also has to expect that a large fraction of users are only known to one system (either A or B). This brings up the question of how to exploit data without known correspondence in a principled manner, a problem generally referred to as *semi-supervised learning*. Notice that the situation is symmetric and that unlabeled data may be available for both systems, i.e˙ sets of vectors $\mathbf{x}_i$ without corresponding $\mathbf{y}_i$ and vice versa. In summary, we have three conceptual requirements roe a machine learning method:

- Perform vector-valued regression *en bloc* and not independently
- Exploit correlations between different output dimensions (or response variables)
- Utilize data without known correspondences

In addition, the nature of the envisioned application requires:

- Scalability of the method to large user populations and many systems/sites
- Capability to deal with missing and incomplete data

3.2.1 Challenges in Automatic Cross System Personalization

As stated earlier, there are three objectives which Cross System Personalization needs to address. These are a) broader user models that can cope with the variety in user modeling, b) handling heterogeneous personalization systems and approaches, and c) giving more control to the user. The second challenge is the most difficult of the above: heterogeneity is difficult to solve using rule-based systems, since such rules have to be humanly created and may be prone to errors. One mechanism of overcoming this difficulty is to use instance data. In our case, this corresponds to user profiles of many different users, who are in a cross-system environment. While a person may use multiple information systems, there is likely to be a lot of consistency between profiles of the same person. Given enough instance data, it might be possible to discover the mapping from profiles at one system to profiles on another system. While this approach is likely to me

more successful against heterogeneity, it opens new challenges that a successful method needs to solve.

Accuracy in Learning Mappings

While machine learning has long been successful at dealing with learning mappings from example data, there are some particular challenges in the cross-system scenario. The first challenge is that both output and input domains are vector spaces, a problem which is called *vector valued learning*. A candidate method should be able to exploit the relationships between output dimensions to perform vector valued regression *en bloc* and not independently. Another basic requirement is to learn as accurately as possible using as less data as possible. Since the best we can do for a new user without any additional data is recommending the most popular items, the chosen method for CSP should be able to outperform popularity vote based recommendations with as few users crossing over from one system to another. Additionally, it is likely that a large part of the user profile is unobserved (e. g. unrated items); therefore a candidate method must be able to deal with sparsity in a principled fashion. Additional challenges include dealing with unlabeled data (users with no profiles on other systems) for training.

Privacy

One important aspect of cross system personalization is privacy. People and companies alike are likely to have reservations against sharing their data with our systems. Users fear the loss of their anonymity, while companies fear a loss of their competitive edge. With our method, the important thing is to discover the underlying social similarity between people and not their exact buying/rating patterns. Recent work (Frankowski et al., 2006) describes the privacy risk that users face of being identified in a collaborative filtering setting, even among hundreds of thousands of other users. Researchers of the privacy problem have long maintained that centralized data servers pose severe security risks, and user data should be distributed to prevent identification. One proposed solution building on this assertion is to introduce a privacy preserving protocol(cf. (Canny, 2002b)) where encrypted user data is used to learn a statistical model of the user data (which is assumed to be discrete numerical). We later describe our scheme for privacy preserving cross system personalization later in Sec. 3.3.4.

Synchronization with Multiple Systems

A practical aspect of cross system personalization would be changing user profiles. One simple case of this is a new user rating some items on the new system, which provides some information in addition to profiles from other systems. Clearly, the predicted profile on the new system can be improved using this information. Another category of change is addition of new items at a system's end. Since no previous information is available about the newly added items, no prediction can be provided (except if a mixture of content based features of the new items is used as well). However, as more ratings and information are added, the model can be updated to take this information into account to make more accurate predictions. We will address both these issues in the later sections of this chapter.

Robustness

The use of data from other systems to make predictions on the current system opens up the issue of trust: *is it possible to manipulate the current system by malicious users*? This question

has been explored by recent researchers who suggest that recommender systems can definitely be manipulated. Particularly vulnerable are collaborative filtering systems where every user is advised by a community of similar users. Carefully designed user profiles can be inserted to push a particular item into the list of recommended items for many end users. In this thesis, we address this issue as well and propose algorithms for detecting and robustifying recommender systems.

3.3 Learning Methods for enabling Cross System Personalization

There are some recent learning methods that can be utilized for vector-valued regression problem (for the CSP task, see previous section), but some of them do not fulfill the above requirements. Kernel dependency estimation (Weston, Chapelle, Elisseeff, Schölkopf, & Vapnik, 2002) (KDE) is a technique that performs kernel PCA (Schölkopf, Smola, & Müller, 1998) on the output side and then learns independent regression functions from inputs to the PCA-space. However, KDE can only deal with unlabeled data on the output side and requires to solve computationally demanding pre-image problems for prediction (Bakir, Weston, & Schölkopf, 2004). Another option is Gaussian process regression with coupled outputs (Keerthi & Chu, 2006). Here it is again difficult to take unlabeled data into account while preserving the computational efficiency of the procedure. The same is true for more traditional approaches like Multi-Layer-Perceptrons with multiple outputs. Instead of using regression methods, we thus propose the use of *manifold learning* in this context. Manifold learning methods generalize linear dimension reduction techniques that have already been used successfully in various ways for collaborative filtering. Moreover, they are usually motivated in an unsupervised setting that can typically be extended to semi-supervised learning in a rather straightforward manner. More specifically, we propose to use the *Laplacian Eigenmaps* (Belkin & Niyogi, 2003) and *Locally Linear Embedding* (LLE) approaches (Saul & Roweis, 2003) as our core method.

As presented in (Ham, Lee, & Saul, 2003; Ham et al., 2005), correspondences between data points can be exploited by using constrained LLE (CLLE) to learn mappings between two vector spaces by semi-supervised alignment of manifolds. The former work also provides empirical evidence that CLLE can outperform standard regression methods. The key idea is to embed user profiles from different systems in low-dimensional manifolds such that profiles known to be in correspondence (i.e˙ profiles of the same user) are mapped to the same point. This means the manifolds will be aligned at correspondence points. This idea can also be extended to linear dimensionality reduction, where data points from different vector-spaces can be constrained to have the same representation in the low-dimensional latent space. Another alternative is to cast the CSP task as a missing value problem for a sparse matrix, and learn a model from incomplete data: Sparse Factor Analysis and PLSA are two methods which we explore in this context. In the next section, we describe how CSP can be treated as matrix completion problem.

3.3.1 Manifold Alignment

Suppose we are given l data points in $\mathcal{S} = \{\mathbf{x}_i \in \mathbb{R}^n\colon \ i = 1,\dots,l\}$. When the data lie approximately on a low-dimensional manifold embedded in the n-dimensional Euclidean space, manifold learning methods such as Laplacian eigenmaps (Belkin & Niyogi, 2003), Hessian eigenmaps (Donoho & Grimes, 2003), Isomap (Tenenbaum, Silva, & Langford, 2000) or Locally Linear Embeddings (Saul & Roweis, 2003) can be used to recover the manifold from a sample $\mathcal{S}$. We pursue the Laplacian Eigenmaps approach, which has been used successfully in semi-

supervised learning (Ham et al., 2005) and for which rigorous convergence results exists in the large sample limit (Hein et al., 2005). LLE and Laplacian Eigenmaps construct a low-dimensional data representation for a given set of data points by embedding the points in a way that preserves the local geometry. Compared to other manifold learning and non-linear dimension reduction algorithms, such as Sammon's MDS (John W. Sammon, 1969) or Isomap (Tenenbaum et al., 2000), the LLE approach is computationally attractive and highly scalable, since it only relies on distances within local neighborhoods. Laplacian Eigenmaps provide an even simpler framework, where computations are performed on the graph Laplacian matrix, and only an eigenvalue decomposition is needed.

The starting point in Laplacian eigenmaps is the construction of a weighted graph whose nodes are the sample points and whose edges connect the nearest neighbors of each node. Sec 2.3.3 describes the Laplacian Eigenmap approach in detail.

Aligned Manifold Learning

Consider now the case where two sets of points are given $\mathcal{S}_x \equiv \{\mathbf{x}_i \in \mathbb{R}^n : i = 1,\dots,n\}$ and $\mathcal{S}_y \equiv \{\mathbf{y}_j \in \mathbb{R}^m : i = 1,\dots,m\}$ where we assume without loss of generality that the first $l \leq \min\{n,m\}$ points are in correspondence. In the case of cross system personalization, $\mathbf{x}_i$ will denote a user profile in system A, $\mathbf{y}_j$ will denote a user profile in system B and $\mathbf{x}_i \leftrightarrow \mathbf{y}_i$ for users u_i, $i = 1,\dots,l$, who are known in both systems. We will separately construct graphs $\mathbf{G}_x$ on $\mathcal{S}_x$ and $\mathbf{G}_y$ on $\mathcal{S}_y$ in order to find low-dimensional embeddings of the points in $\mathcal{S}_x$ and $\mathcal{S}_y$, respectively. In addition, we will follow the approach in (Ham et al., 2005) and utilize the correspondence information to enforce that embeddings of user profiles for the same user are close to one another. To that extend we compute a simultaneous embedding f of $\mathcal{S}_x$ and g of $\mathcal{S}_y$ by minimizing the objective

$$C(f,g) = \sum_{i=1}^{l} (f_i - g_i)^2 + \lambda \left(f^T \mathbf{L}_x f + g^T \mathbf{L}_y g . \right) \tag{3.2}$$

More specifically, in order to deal with simultaneous re-scaling of f and g, one minimizes the Rayleigh quotient

$$\tilde{C}(f,g) = \frac{C(f,g)}{f^T f + g^T g} . \tag{3.3}$$

By defining the combined graph $\mathbf{G} \equiv \mathbf{G}_x \cup \mathbf{G}_y$ with Laplacian $\mathbf{L}_{\mathcal{XY}}$ and combined functions $h = (f^T, g^T)^T$ the above objective can be rewritten as

$$\tilde{C}(h) = \frac{h^T \mathbf{H} h}{h^T h}, \quad \text{where } \mathbf{H} \equiv \lambda \mathbf{L}_{\mathcal{XY}} + \begin{pmatrix} \mathbf{U}^{nn} & \mathbf{U}^{nm} \\ \mathbf{U}^{mn} & \mathbf{U}^{mm} \end{pmatrix} \tag{3.4}$$

and $\mathbf{U}^{nm} \in \mathbb{R}^{n \times m}$ is diagonal with $U^{nm}_{ii} = 1$ for $1 \leq i \leq l$ and 0 otherwise. Again, a solution is obtained as before by finding the eigenvectors of the matrix $\mathbf{L}_{\mathcal{XY}}$.

One can also enforce the embeddings of points in correspondence to be the same on both manifolds (Ham et al., 2005). In this case, one identifies the first l points in $\mathcal{S}_x$ and $\mathcal{S}_y$, resulting in a combined graph $\mathbf{G}$ with $n+m-l$ nodes with a combined weight matrix. Notice that weights between pairs of nodes with indices $1 \leq i,j \leq l$ are simply given by the sum of the weights from $\mathbf{G}_x$ and $\mathbf{G}_y$. Introducing functions h one then minimizes

$$\tilde{C}(h) = \frac{h^T \mathbf{L}_{\mathcal{XY}} h}{h^T h}, \quad \text{s.t.} \sum_i h_i = 0 . \tag{3.5}$$

The solution to the above minimization is given by an eigenvalue decomposition of the symmetric $\mathbf{L}_{\mathcal{XY}}$, making the solution h an eigenmap of the combined Laplacian.

Using Locally Linear Embedding for Aligned Manifold Learning

One way to define the weights W_{ij} for neighboring nodes in the graph is to compute them based on a local affine approximation. This idea has originally presented in the context of the Locally Linear Embedding (LLE) method (Saul & Roweis, 2003). Its use as a preprocessing step in conjunction with Laplacian Eigenmaps has been proposed in (Ham et al., 2005).

While the LLE algorithm can be used in its own right for manifold learning, we have employed it here to compute the affinity matrix for the Laplacian eigenmap method. The weights can be computed by solving a quadratic optimization problem:

$$W_{ij}^{*} = \arg\min_{W} |\mathbf{x}_i - \sum_{j \sim i} W_{ij}\mathbf{x}_j|^2 \,, \text{s.t.} \sum_j W_{ij} = 1 \,,$$

Reconstructing Points from Alignments

The remaining problem we would like to discuss is how to map a point on the low-dimensional manifold back into the original data space. This is particularly relevant in the context of manifold alignment, where one ultimately may want to realize a mapping from $\mathbb{R}^n \to \mathbb{R}^m$. After mapping a point $\mathbf{x} \in \mathbb{R}^n$ to a k-dimensional representation $\hat{\mathbf{x}}$, we would thus like to compute an approximation $\mathbf{y} \in \mathbb{R}^m$ by finding a pre-image to $\hat{\mathbf{y}}$ and identifying $\hat{\mathbf{y}} = \hat{\mathbf{x}}$.

We do this in the following manner: For a point $\mathbf{x}_i \in \mathcal{S}_x$ with $i > l$ and manifold coordinates $\hat{\mathbf{x}}_i$ we first identify a set of k nearest neighbors $\hat{\mathbf{y}}_r$ on the manifold among the points that are images of points in $\mathcal{S}_y$, resulting in some set of image/pre-image pairs $\{(\mathbf{y}_r, \hat{\mathbf{y}}_r)\}$. We then compute the optimal affine combination weights w_r that optimally reconstruct $\hat{\mathbf{x}}_i \approx \sum_r w_r \hat{\mathbf{y}}_r$. Then the pre-image prediction is given by $F(\mathbf{x}_i) = \sum_r w_r \mathbf{y}_r$. Similarly, we can compute an inverse map by exchanging the role of the $\mathbf{x}_i$ and $\mathbf{y}_j$. Notice that one can also generalize this for arbitrary new samples $\mathbf{x} \in \mathbb{R}^n$ by generalizing the manifold mapping $\mathbf{x} \mapsto \hat{\mathbf{x}}$ to new points, which can be done along the lines presented in (Bengio et al., 2003).

The Manifold Alignment Collaborative Filtering Algorithm

It has been reported that dimensionality reduction techniques are effective for k-NN algorithms used typically for collaborative filtering (B. Sarwar, Karypis, Konstan, & Riedl, 2000). The manifold alignment algorithm essentially works as a k-NN algorithm as well. After projecting user profile vector from two (or more) systems on a low-dimension manifold, we are able to find nearest neighbors based on distance measures like Euclidean distance. The additional constraint of aligning profiles belonging to the same user aligns the two sub-manifolds and helps in finding more effective neighborhoods. The manifold alignment technique also provides privacy features since user profiles do not have to be exchanged between systems, as long as a weighted neighborhood of every user can be exchanged. Our algorithm for alignment learning has the following steps:

1. **Neighborhood identification**: For each point $\mathbf{x}_i \in \mathcal{X}$, we find the k-nearest neighbors. NLDR techniques usually use Euclidean distance to identify the nearest points. In our setting, data is sparse, therefore Euclidean distance on pure data is not necessarily effective unless missing data is imputed. Options here include mean imputation (with item mean), measuring distance only on commonly-voted items, or using a distance based on a similarity measure like Pearson's correlation coefficient. This procedure also has to be repeated for $\mathbf{y}_i \in \mathcal{Y}$. Note that choosing exactly k-nearest neighbors for every node may result in a graph Laplacian that is not symmetric.

Algorithm 1 ComputeManifold-NLDR $(\mathcal{X}, \mathcal{Y}, c, K, d)$

Input: Matrices $\mathcal{X}, \mathcal{Y}$ with the first c columns aligned. K is the number of neighbors, d is the dimensionality of the manifold.

1: Impute missing values with mean item votes respectively for $\mathcal{X}$ and $\mathcal{Y}$ to get $\mathcal{X}_{norm}, \mathcal{Y}_{norm}$.
2: Calculate adjacency matrices $A_{\mathcal{X}}, A_{\mathcal{Y}}$ for graphs represented by $\mathcal{X}_{norm}, \mathcal{Y}_{norm}$.

$$A_{\mathcal{X}}(i,j) = \begin{cases} 1, & \text{if } i \sim j \\ 0, & \text{otherwise}. \end{cases}$$

3: Compute reconstruction Weights $W_{\mathcal{X}}, W_{\mathcal{Y}}$. For Laplacian Eigenmaps, $W_{\mathcal{X}} = \mathbf{F}(A_{\mathcal{X}})$, while for LLE, a quadratic program has to be solved for every point.
4: Compute the Graph Laplacians $L_{\mathcal{X}}, L_{\mathcal{Y}}$ from the Weight Matrices as defined in equation 2.9. For constrained LLE, use $L^* = (I - W)^T(I - W)$.
5: Compute

$$\mathbf{L}_{\mathcal{X}\mathcal{Y}} = \begin{bmatrix} L_{\mathcal{X}}^{cc} + L_{\mathcal{Y}}^{cc} & L_{\mathcal{X}}^{cs} & L_{\mathcal{Y}}^{cs} \\ L_{\mathcal{X}}^{sc} & L_{\mathcal{X}}^{ss} & 0 \\ L_{\mathcal{Y}}^{sc} & 0 & L_{\mathcal{Y}}^{ss} \end{bmatrix} \tag{3.6}$$

c represents the points in alignment, while s represents the *single* points.
6: Find the low dimensional manifold $\mathbf{H}_M$ for the matrix $\mathbf{L}_{\mathcal{X}\mathcal{Y}}$. $\mathbf{H}_M$ has a dimensionality of $(n_{\mathcal{X}} + n_{\mathcal{Y}} - c) \times d$.

Output: Low dimensional manifold $\mathbf{H}_M$

Using LLE, one selects *exactly* k neighbors, while for Laplacian eigenmap, one does not impose this constraint. As a result, the neighborhood of some points in the Laplacian Eigenmaps method can be much larger than k. This usually shows the importance of a node and is similar to the notion of *authority* nodes in the HITS algorithm (Kleinberg, 1999).

2. **Calculate Affinity Matrix**: After the k nearest neighbors have been identified for every point, an affinity weight with every neighbor has to be computed. Options here include an affine decomposition (like in LLE), an exponential weight (aka the heat kernel used in Laplacian Eigenmaps) based either on euclidean distance or on a similarity measure like Pearson's correlation. Finally, the Laplacians $\mathbf{L}_{\mathcal{X}}, \mathbf{L}_{\mathcal{Y}}$ of the graphs characterized by affinity matrices for $\mathcal{X}$ and $\mathcal{Y}$ are computed.

$$\begin{aligned} &W_{ij} = \exp\left[-\beta \|\mathbf{x}_i - \mathbf{x}_j\|^2\right] \text{ or} \\ &W_{ij} = 1 \text{ or} \\ &W_{ij} = \arg\min_{W} |\mathbf{x}_i - \sum_{j \sim i} W_{ij}^* \mathbf{x}_j|^2, \text{s.t.} \sum_j W_{ij}^* = 1 \end{aligned} \tag{3.7}$$

3. **Compute points on manifold**: This is usually done by solving an eigenvalue problem, and finding the eigenvectors of the Laplacian $\mathbf{L}$ (or ${\mathbf{L}^T}_*\mathbf{L}_*$ in case of LLE). For points in alignment, a modified eigenvalue problem has to be solved: A joint graph of the two datasets is formed and the eigenvectors of this Laplacian matrix $\mathbf{L}_{\mathcal{X}\mathcal{Y}}$ are computed (see Eq. (3.6)). The only parameter here is the dimensionality of the manifold (the number of eigenvectors that are chosen).

Algorithm 2 ComputePreimage $(\mathbf{H}_M, c, n_{\mathcal{X}}, n_{\mathcal{Y}}, K, \mathcal{X}_{norm}, \mathcal{Y}_{norm}, n_{\mathcal{X}}, n_{\mathcal{Y}})$

Input: Matrix $\mathbf{H}_M$ of size $(n_{\mathcal{X}} + n_{\mathcal{Y}} - c) \times d$ representing the aligned manifold with c points overlapping between manifolds of $\mathcal{X}$ and $\mathcal{Y}$. K is the number of neighbors. $\mathbf{H}_M$ has the first c points representing the overlapping users. The next $n_{\mathcal{X}} - c$ points represents single points of $\mathcal{X}$ and the last $n_{\mathcal{Y}} - c$ points represent the single points of $\mathcal{Y}$. $\mathbf{H}_M(i)$ denotes the i^{th} d-dimensional point on the manifold.

1: Extract sub-manifold $\mathbf{H}_{\mathcal{Y}}$ by combining the first c and the last $n_{\mathcal{Y}}$ points of $\mathbf{H}_M$.
2: **for** $i = (c+1)$ to $(c + n_{\mathcal{X}})$ **do**
3: $\hat{\mathbf{x}}_i \leftarrow \mathbf{H}_M(i)$
4: Compute the K nearest neighbors $\hat{\mathbf{y}}_r$ of $\hat{\mathbf{x}}_i$ on the sub-manifold $\mathbf{H}_{\mathcal{Y}}$. Let $\mathbf{y}_r$ represent the preimage of $\hat{\mathbf{y}}_r$ in $\mathcal{Y}_{norm}$.
5: Compute affine weights $\mathbf{W}^* = (w_r)_{r=1}^K$ for the neighborhood.
6: Compute Preimage prediction $F(\hat{\mathbf{x}}_i) = \sum_r w_r \mathbf{y}_r$.
7: $\hat{\mathcal{X}}_s(i-c) = F(\hat{\mathbf{x}}_i)$
8: **end for**
9: Repeat above procedure by exchanging $\mathcal{X}$ and $\mathcal{Y}$ to calculate preimages for single points of $\mathcal{Y}$.

Output: Preimages $\hat{\mathcal{X}}_s, \hat{\mathcal{Y}}_s$

4. **Compute preimages for points not in correspondence**: In this step, neighborhoods for points not in correspondence are formed in a manner similar to the first step. The normal method to follow here is to find the nearest neighbors (based on Euclidean distance) and compute a weight distribution over this neighborhood. We do this in the following manner: For a point $\mathbf{x}_i \in \mathcal{X}$ with $i > c$ and manifold coordinates $\hat{\mathbf{x}}_i$ we first identify a set of k nearest neighbors $\hat{\mathbf{y}}_r$ on the manifold among the points that are images of points in $\mathcal{Y}$, resulting in some set of image/pre-image pairs $\{(\mathbf{y}_r, \hat{\mathbf{y}}_r)\}$. We then compute the optimal affine combination weights w_r that optimally reconstruct $\hat{\mathbf{x}}_i \approx \sum_r w_r \hat{\mathbf{y}}_r$. Then the pre-image prediction is given by $F(\mathbf{x}_i) = \sum_r w_r \mathbf{y}_r$. Similarly, we can compute an inverse map by exchanging the role of the $\mathbf{x}_i$ and $\mathbf{y}_j$. Notice that one can also generalize this for arbitrary new samples $\mathbf{x} \in \mathbb{R}^n$ by generalizing the manifold mapping $\mathbf{x} \mapsto \hat{\mathbf{x}}$ to new points, which can be done along the lines presented in (Bengio et al., 2003).

Computation Complexity The Laplacian Eigenmap method clearly offers computational advantages over the LLE method. The LLE method has 3 basic steps: a) find nearest neighbors, b) compute reconstruction weights, and c) find eigenvalues and eigenvectors. For two datasets of sizes $m_{\mathcal{X}} \times n_{\mathcal{X}}$ and $m_{\mathcal{X}} \times n_{\mathcal{X}}$ with c common points, the size of the common graph is $n_{\mathcal{X}} + n_{\mathcal{Y}} - c$ nodes. The complexity of the LLE method for a matrix with n points each of dimensionality m is thus $O(mn^2) + O(mnk^3) + O(kn^2) \equiv O(nm(n + k^3))$. The Laplacian Eigenmap method essentially skips the second step, and hence has a complexity of $O(dn^2)$. Therefore the overall complexity of *Algorithm 1* (without the reconstruction of user profiles) is $O(mn(n + k^3))$ where $n = n_{\mathcal{X}} + n_{\mathcal{Y}} - c$. For our experiments, k typically had a value between $24 - 48$, while n was around 1000. In this range, k^3 was 1-2 orders of magnitude higher than n, thus explaining the difference between the running times of LLE and LapE based NLDR. Note however that the entire alignment computation can be performed off line. For a new user, out-of-sample extensions for LLE and Laplacian Eigenmaps (Bengio et al., 2003) can be used. These typically have a computational complexity of $O(nm) + O(mk^3)$. Importantly, the neighborhood

formation step can be reused in the second part of the algorithm where user profiles have to be reconstructed.
The reconstruction of a user profile (*Algorithm 2*) involves (a) neighborhood formation (b) finding reconstruction weights, and (c) combining neighbor votes. The complexity reconstructing the profile for one user therefore is $O(mn)+O(mk^3)+O(mk)$. The significant term here depends on the values of the parameters: for a large neighborhood, the second term dominates. However if the number of items is very large (say a million), then the last term is the most significant one.

Privacy One important aspect of cross system personalization is privacy. With our method, the important thing is to discover the underlying social similarity between people and not their exact buying/rating patterns. A less accurate, but more secure approach could start with a dimensionally reduced user database from say 1 million items to 1000 dimensions. Also the complete user database does not need to be known: a random selection of a sufficient number of users might be sufficient to learn the mapping from one system to another.

Scaling to a n-system scenario The manifold alignment algorithm needs only a minor modification in case some users are common to all n systems. This modification in the step where a joint graph $\mathbf{G}$ is formed. The low dimensional embedding of this graph will have all the sub-manifolds aligned. More fine-tuned modifications are required in case the set of overlapping users is different between different users. Manifold alignment in $n-$system scenario is successful only if atleast a small fraction of users cross from one system to another. In order to test this scenario, larger datasets are needed.

3.3.2 Cross System Personalization as a matrix completion problem

Two basic assumptions help us in casting the CSP task as a missing value problem: *first*, that users have their profiles for multiple systems available to them, and *second*, that users are willing to provide their multiple profiles for computing a mapping between the profile formats of these systems. We assume also for now that the user profile stored by multiple systems is numerical and of a fixed length (i.eȧ *vector*). Note that this assumption holds trivially for collaborative filtering.

In a two-system scenario, we have two sites A and B, containing user profiles for their users represented as vectors. A user i has a profile $\mathbf{x}_i^A \in \mathbb{R}^m$ at site A, and a profile $\mathbf{x}_i^B \in \mathbb{R}^p$ at site B. We assume that c users are common to both sites and that the data matrices can be partitioned as

$$\mathbf{X}^A = \begin{bmatrix}\mathbf{X}_c^A & \mathbf{X}_s^A\end{bmatrix}, \qquad \mathbf{X}^B = \begin{bmatrix}\mathbf{X}_c^B & \mathbf{X}_s^B\end{bmatrix}, \tag{3.8}$$

where $\mathbf{X}_c^A$ and $\mathbf{X}_c^B$ represent the sub-matrices of $\mathbf{X}^A$ and $\mathbf{X}^B$ corresponding to the common users and $\mathbf{X}_s^A$ and $\mathbf{X}_s^B$ the sub-matrices for users that are unique to A and B.

One way of looking at the CSP problem is to relate the profiles in both (or multiple) systems by assuming that the user profiles are likely to be consistent in terms of the basic factors, i.e˙ that they can be explained by latent factors common to both systems. This is similar to the *manifold alignment* idea of (Ham et al., 2003).

A simple manner of enforcing this constraint is to construct a new combined random vector $\mathbf{x} = [\mathbf{x}^A\ \mathbf{x}^B]$ and to perform a joint factor analysis over the combined profile space of system A

and B. This means we effectively generate a data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A & ? \\ \mathbf{X}_c^B & ? & \mathbf{X}_s^B \end{bmatrix}, \tag{3.9}$$

where '?' denotes matrices of appropriate size with unobserved values. Note that the other sub-matrices of $\mathbf{X}$ may also contain (many) missing entries.

It is interesting to make a further simplification by restricting the data matrix to users that are known to both systems

$$\mathbf{X}_c = \begin{bmatrix} \mathbf{X}_c^A \\ \mathbf{X}_c^B \end{bmatrix}, \tag{3.10}$$

and to ignore the data concerning users only known to one system. Obviously, this will accelerate the model fitting compared to working with the full matrix $\mathbf{X}$. Also, this setting is more realistic, since in a real world scenario, only the restricted portion of crossing users might be made available by individual systems. This situation corresponds to a *supervised learning* setting where labeled output data is available for all training samples. However, it is likely to be less accurate than the *semi-supervised learning* setting where $\mathbf{X}$ is used, since the unlabeled samples will potentially improve the estimation of the parameters.

A third scenario is one, where each site gets profiles of some of its users from another system, but wants to make use of all of its locally available user profiles. In this case, missing value problem has to be solved for the following data matrix:

$$\mathbf{X}_1 = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A \\ \mathbf{X}_c^B & ? \end{bmatrix} \text{ and } \mathbf{X}_2 = \begin{bmatrix} \mathbf{X}_c^A & ? \\ \mathbf{X}_c^B & \mathbf{X}_s^B \end{bmatrix}, \tag{3.11}$$

will be performed at system A and B, respectively. We have explored this model in our experiments for the case where users with an existing profile at one site bring along their profile from the second system.

3.3.3 Sparse Factor Analysis

In a two system scenario, we have two sites A and B, containing user profiles for their users represented as vectors. A user i has a profile $\mathbf{x}_i^A \in \mathbb{R}^m$ at site A, and a profile $\mathbf{x}_i^B \in \mathbb{R}^p$ at site B. We assume that c users are common to both site and that the data matrices can be partitioned as

$$\mathbf{X}^A = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A \end{bmatrix}, \qquad \mathbf{X}^B = \begin{bmatrix} \mathbf{X}_c^B & \mathbf{X}_s^B \end{bmatrix}, \tag{3.12}$$

where $\mathbf{X}_c^A$ and $\mathbf{X}_c^B$ represent the sub-matrices of $\mathbf{X}^A$ and $\mathbf{X}^B$ corresponding to the common users and $\mathbf{X}_s^A$ and $\mathbf{X}_s^B$ the sub-matrices for users that are unique to A and B.

One way of looking at the CSP problem in the context of factor analysis is to relate the profiles in both (or multiple) systems by assuming that the user profiles are likely to be consistent in terms of the basic factors, i.e˙ that they can be explained by latent factors common to both systems. This is similar to the *manifold alignment* idea of (Ham et al., 2003) and effectively couples the factor analysis between the different systems.

A simple manner of enforcing this constraint is to construct a new combined random vector $\mathbf{x} = [\mathbf{x}^A\ \mathbf{x}^B]$ and to perform a joint factor analysis over the combined profile space of system A

and B. This means we effectively generate a data matrix

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A & ? \\ \mathbf{X}_c^B & ? & \mathbf{X}_s^B \end{bmatrix}, \tag{3.13}$$

where '?' denotes matrices of appropriate size with unobserved values. Again we assume that the columns of $\mathbf{X}$ are independent realizations of $\mathbf{x}$ in a factor analysis model. Note that the other sub-matrices of $\mathbf{X}$ may also contain (many) missing entries.

As mentioned earlier, there are also other related missing value problems which can be solved for the CSP task. The reader can refer back to Sec. 3.3.2 for more details.

EM for Factor Analysis A standard approach for performing maximum likelihood estimation in a factor analysis model is the expectation maximization (EM) algorithm. For completeness, we state here the expectation maximization recurrence relations (for simplicity restricted to the $\mu = 0$ case). In the EM approach, maximum likelihood estimation is performed by maximizing the expected complete data log-likelihood with respect to the parameters of the model, i.e˙ one needs to perform the maximization

$$(\hat{\Lambda}, \hat{\Psi}) = \underset{\Lambda,\Psi}{\operatorname{argmax}} \sum_{i=1}^{n} \mathbf{E}_{\mathbf{z}} \left[\log p(\mathbf{x}_i, \mathbf{z}; \Lambda, \Psi) \right], \tag{3.14}$$

where the expectation for $\mathbf{z}$ is computed with respect to the posterior distribution of $\mathbf{z}$ given a particular profile $\mathbf{x}_i$. Note that the latter will also depend on the parameters Λ and Ψ, so that both steps, the computation of the posteriors (E-step) and the re-estimation of the parameters (M-step) need to be alternated until (guaranteed) convergence. The posterior distribution of the $\mathbf{z}$ is a multivariate normal for which the mean vector and co-variance matrix can be calculated as

$$\begin{aligned} &\mathbf{E}[\mathbf{z}|\mathbf{x}] = \langle \beta, \mathbf{x} \rangle, \;\; \mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}\right] = \mathbf{I} - \beta\Lambda + \beta\mathbf{x}\mathbf{x}'\beta' \;\; \text{where} \;\; \beta = \Lambda'(\Psi + \Lambda\Lambda')^{-1} \\ &\Lambda = \left(\sum_{i=1}^{n} \mathbf{x}_i \mathbf{E}(\mathbf{z}|\mathbf{x}_i)' \right) \left(\sum_{i=1}^{n} \mathbf{E}(\mathbf{z}\mathbf{z}'|\mathbf{x}_i) \right)^{-1} \\ &\Psi = \frac{1}{n} \operatorname{diag} \left[\sum_{i=1}^{n} \mathbf{x}_i \mathbf{x}_i' - \Lambda \mathbf{E}(\mathbf{z}|\mathbf{x}_i) \mathbf{x}_i' \right]. \end{aligned} \tag{3.15}$$

Learning from Incomplete data in Factor Analysis

Canny's approach (Canny, 2002b) called *Sparse Factor Analysis*(SFA) also uses an Expectation Maximization recurrence to solve the factor analysis model, while paying attention to the case of incomplete data (Ghahramani & Jordan, 1994). An additional advantage of his approach is that the model building can be distributed among participating users in a manner that preserves privacy. In Canny's approach, each user can contribute to the model building by computing locally terms that contribute to the overall computation. Since each user has rated only a subset of items, only the available rating data is used to compute the value of the latent variables, effectively removing missing variables from the inference process. Defining an $m \times m$ trimming diagonal matrix $\mathbf{T_i}$ for the i^{th} user which has $\mathbf{T_i}(j,j) = 1$ where ever the user i has voted for item j, the factor analysis E-step equations are modified as follows:

$$\begin{aligned} &\mathbf{E}[\mathbf{z}|\mathbf{x}_i] = \langle \beta_i, \mathbf{x}_i \rangle, \beta_i = \Lambda_i'(\Psi + \Lambda\Lambda_i')^{-1}, \Lambda_i = \Lambda\mathbf{T}_i \\ &\mathbf{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right] = I - \beta_i\Lambda_i + \beta_i\mathbf{x}_i\mathbf{x}_i'\beta_i'. \end{aligned} \tag{3.16}$$

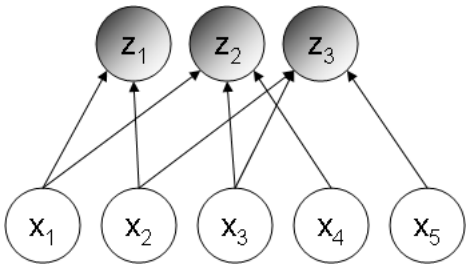

Figure 3.6: Factor analysis using incomplete data. Only observed variable (x_1, x_2, x_3) are used to predict the latent variables (z_1, z_2, z_3).

Similarly, the M-step equations can be generalized to the missing data case to yield:

$$\begin{aligned} \Lambda^{[t]} &= \left(\sum_{i=0}^{n} \mathbf{T}_i \mathrm{E}\left[\mathbf{z}\mathbf{z}'|\mathbf{x}_i\right]'\right)^{-1} \left(\sum_{i=0}^{n} \mathbf{x}_i \mathrm{E}[\mathbf{z}|\mathbf{x}_i]\right) \\ \Psi^{[t]} &= \left(\sum_{i=0}^{n} \mathbf{T}_i\right)^{-1} \mathrm{diag}\left(\sum_{i=0}^{n} \mathbf{x}_i\mathbf{x}_i' - \mathbf{T}_i\Lambda^{[t]}\mathrm{E}[\mathbf{z}|\mathbf{x}_i]\mathbf{x}_i'\right) \end{aligned} \tag{3.17}$$

A detailed derivation can be found in (Traupman & Wilensky, 2004). The model requires an initial value for Λ and ψ. A random matrix with a Gaussian distribution is used for Λ. This random matrices are then used by the linear regression model to generate an estimate. A linear regression model assumes no noise ($\psi = 0$), and can be obtained from the Eq. (3.16) by setting $\psi = 0$. The linear regression uses the following update scheme:

$$\mathbf{z} = (\Lambda'\Lambda)^{-1}\Lambda'\mathbf{z}, \qquad \Lambda^{[t]} = \mathbf{x}\mathbf{z}'(\mathbf{z}\mathbf{z}')^{-1}$$

Here the matrix $\mathbf{x}$ is a $n \times m$ user rating matrix, were missing values are replaced by some mean values. An over-all average has been used by Canny. A few iterations of this recurrence gives a reasonable starting value of Λ which can be used by the factor analysis model. Please note that Canny's approach assumes a probabilistic distribution in the latent space with a zero-mean. While this is not true for user ratings, simple transformations like subtracting per-user or per-item means from every known value can create data with approximately zero mean.
With the framework established, we now explain how it can be applied. At first, the linear regression model is used to calculate an initial estimate of Λ, using a low dimensional latent space (say $d = 5 - 10$). After 10 iterations of linear regression, the factor analysis model is initialized with this estimate of Λ. The EM recurrence for factor analysis (EM-FA) converges reliably in $15 - 25$ iterations. After this, a new user with a partial rating vector $\mathbf{x}_j$ can be used to calculate $\mathbf{z}_j$ using Eq. (3.16). Given $\mathbf{z}_j$, $\Lambda\mathbf{z}_j$ provides the complete rating vector $\hat{\mathbf{x}}_j$.

Time and Space complexity The time complexity of a single iteration of Sparse FA is $O(nmk^2)$, which is linear in the size of the user database. The prediction time is $O(mk^2)$ which is linear in the number of items. The space complexity is low $\sim O(mk)$, as is characteristic of algorithms for model-based collaborative filtering.

Extension to n-system Scenario One of the important factors on which the success of a method for CSP depends, is to be able to effectively use data spread across multiple systems. The 2-system scenario is clearly simplistic: a real world scenario has multiple systems with users

having profiles in one or more of these systems. Besides the obvious case of a new user entering a system, where CSP has benefits, it is also desirable that the recommendation quality for an existing user can be improved by leveraging his/her profiles in other systems. While the n-system scenario can be dealt with by treating the entire setup like one huge collaborative system, with items and user ratings distributed across multiple sites, subtle issues make the difference crucial. These issues concern correspondence (user profiles on only a subset of systems), and item overlap. Therefore, a generalized approach should take these issues into consideration.

We start by assuming, as earlier, that each site can be modeled by a linear factor analysis model. Further, corresponding profiles have the same representation in the latent space. Thus, if random variables $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n$ represent the user profiles at the n sites respectively, then we form a new random variable $\mathbf{x}$ by concatenating the n vectors, i.e:

$$\mathbf{x} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad ... \quad \mathbf{x}_n] \tag{3.18}$$

Let the matrix $\mathbf{X}$ encode all the user profiles: as before, each user now represents a column in this matrix. Note that this simple setup can allow for correspondence for every user, since the combined user profile is a concatenation of user profiles at individual sites. In case a site does not have a profile for the given user, a vector (of appropriate length) with missing values '?' is chosen. The resulting model can then be used in a fashion similar to the 2-system scenario: All known ratings for all systems are combined appropriately (as dictated by Eq. (3.18)) to yield a vector of length m (where m is the sum of the number of items in all n -systems). This vector is then first projected to get the lower dimension embedding $\mathbf{z}$ in the latest space, and then $\Lambda\mathbf{z}$ gives the desired complete vector with all predicted values filled in. This approach also does not need explicit item-to-item mapping in case the same items are available in more than one system. Instead, the learning approach can figure this out by looking at only principal components in the data.

CSP for Existing Users As mentioned before, one of the objectives of CSP is to leverage data about the user available at other sites. While we posit that new users can unmistakably gain from CSP, users with existing profiles should also be able to gain from their profiles at other systems in a similar way. The advantage offered by CSP in this context is a systematic and principled integration of data and evidence across all systems. By taking user ratings at other sites into account in addition to locally gathered data, one can expect to get more accurate ratings. In our experimental results, we have tested this hypothesis by performing *All-but-1*, *All-But-5*, and *Only-n* tests at site A, while providing information about the ratings at site B as side-information.

Privacy The framework we adopt here is based on Privacy Enhanced Collaborative filtering (cf. (Canny, 2002b)), where user profile data is not known to anyone except the system and the user. The computation of the factor model parameters can be decentralized and users can provide only *their* profile's contribution to a central totaler. Importantly, this contribution can be encrypted, and properties of public-key/private-key encryption allow the computation to proceed with these encrypted values without loss of precision. The key idea of the distributed computation is the use of encryption homomorphism which also products of vectors to be calculated from their encrypted version. The interested reader can find more details of the encryption framework and its use in (Canny, 2002b). We will discuss a modification of this scheme for the purpose of CSP in the next section on Distributed PLSA.

3.3.4 Distributed Probabilistic Latent Semantic Analysis

PLSA is a probabilistic variant of Latent Semantic Analysis (LSA), which is an approach to identify hidden semantic associations from co-occurrence data. The core of PLSA is a latent variable model (also known as the aspect model) for general co-occurrence data which associates a hidden variable $\mathbf{z} \in Z = \{z_1, z_2, ..., z_K\}$ with each observation. In the context of collaborative filtering, each observation corresponds to a vote by a user to a item. The space of observations is normally represented as an $M \times N$ co-occurrence matrix (in our case, of M items $\mathcal{Y} = \{y_1, y_2, .., y_M\}$ and N users $\mathcal{X} = \{x_1, x_2, .., x_N\}$. The aspect model can be described as a generative model:

- select a data item y from $\mathcal{Y}$ with probability $P(y)$,
- pick a latent factor $\mathbf{z}$ with probability $P(z|y)$,
- generate a data item x from $\mathcal{X}$ with probability $P(x|z)$.

As a result we obtain an observed pair (x, y), while the latent factor variable z is discarded. Since in collaborative filtering we are usually interested in predicting the vote for an item for a given user, we are interested in the following conditional model:

$$P(y|x) = \sum_z P(y|z)P(z|x) \tag{3.19}$$

Further details have been provided in Sec. 2.4.2.

Learning a Gaussian PLSA model

PLSA requires a single parameter to be set (the number of communities), given data as a list of triplets (u, y, v), where u denotes a user, y an item and v a numerical vote or rating. The centralized version of PLSA requires several iterations of the EM update equations, which is run over all data. At every step, the following equations are run:

E-Step: (requires knowing μ and σ)

$$p(z|u, y, v) = \frac{p(z|u)p(v|z, y)}{\sum_{z'} p(z|u)p(v|z, y)} \tag{3.20}$$

M-Step 1:

$$p(z|u) = \frac{\sum_{(u',y,v):u=u'} p(z|u, y, v)}{\sum_{(u',y,v):u=u'} 1}. \tag{3.21}$$

M-Step 2: μ and σ are updated

$$\mu_{z,y} = \frac{\sum_{(u,y',v):y'=y} p(z|u, y, v)\, v}{\sum_{(u,y',v):y'=y} p(z|u, y, v)} \tag{3.22}$$

$$\sigma^2_{z,y} = \frac{\sum_{(u,y',v):y'=y} p(z|u, y, v)\, (v - \mu_{y,z})^2}{\sum_{(u,y',v):y'=y} p(z|u, y, v)} \tag{3.23}$$

Privacy Preserving Distributed PLSA

We assume that each user accesses the recommendation system via a client which can communicate with other clients. We also assume a completely distributed setting where each client can interact with every other client like in a peer to peer environment. The main goal

behind distributed PLSA is that private user data should not be shared with other users or with a central server. However the PLSA model is known to everyone, and can be used to by a user's client to compute recommendations for the user. Therefore, a new user only needs to know the probability distribution over the user communities z and the values of PLSA parameters μ and σ. The probability distribution can be computed by Eq. (3.21), given the model parameters.

Initially, given the first n users, the initial model has to be constructed. For Gaussian PLSA, this requires the repeated iteration of the EM equations (3.20)-(3.23). To maintain our goals of privacy, the EM equations have to be computed in a distributed fashion, with contributions from each user made available in the encrypted format.

Our communication protocol between the clients has two phases: in the first phase, the model parameters are computed by iterating the EM equations; the second phase is the normal recommendation phase where a trained model is available and is available to everyone for computing their own recommendations. Similar protocols based on shared Elgamal encryption (Elgamal, 1985) have also been used by Canny (cf. (Canny, 2002a)).

Phase 1: Training the dPLSA model

In the first version of the protocol, we assume all users to be honest. We assume that the set of items $\mathbf{y}$ and the set of users $\mathbf{u}$ remains fixed during the entire protocol, and have a size of M items and N users. Note that we refer to the combined user profile in this protocol ($\mathbf{x} = [\mathbf{x}^A\ \mathbf{x}^B]$, with a combined dimensionality of M), and build a model for the matrix $\mathbf{X}$. The protocol proceeds in the following fashion:

1. At first, the number of communities is fixed, and this parameter K is communicated to every client.
2. The first set of model parameters are initiated as $\mu_{z,y} = \{\mathbf{0}\}^{K\times M}$ and $\sigma_{z,y} = \{\mathbf{1}\}^{K\times M}$. Further, each client initiates the probability distribution of belonging to a user community to a random distribution.

$$\mathbf{P}_{z|u} = [p(z|u)]^{K\times N},\ \text{such that}\ \sum_z p(z|u=i) = 1\,,\ \forall i$$

3. Each client receives the unencrypted values of μ and σ.
4. Each client computes the prior probabilities using given values for μ and σ

$$p(v|z,y) = e^{-\frac{(\mu_{z,y}-v_{u,y})^2}{2\sigma_{z,y}}}/\sigma_{z,y} \tag{3.24}$$

5. Using $p(z|u)$ and $p(v|z,y)$ calculated in the previous step, each client computes the posterior probabilities of each of its votes:

$$p(z|u=i,y,v) = \frac{p(z|u=i)p(v|z,y)}{\sum_{z'} p(z|u=1)p(v|z,y)} \tag{3.25}$$

6. Each user also updates their probability distribution over the user communities.

$$p(z|u=i) = \frac{\sum_{(u,y,v):u=i} p(z|u=i,y,v)}{\sum_{(u',y,v):u=i} 1}. \tag{3.26}$$

7. Each client computes two matrices of fixed point[5] numbers

$$\mathbf{F}_i^{k\times m}, \text{ where } \mathbf{F}_i(z,y) = \sum_{(u=i,y',v):y'=y} p(z|u=i,y,v)v \tag{3.27}$$

$$\mathbf{G}_i^{k\times m}, \text{ where } \mathbf{G}_i(z,y) = \sum_{(u=i,y',v):y'=y} p(z|u=i,y,v) \tag{3.28}$$

Notice that the overall mean $\mu_{z,y}$ can be written as

$$\mu^{k\times m}, \text{ s.t. } \mu_{z,y} = \frac{\mathbf{F}_1(z,y)+\mathbf{F}_2(z,y)+...+\mathbf{F}_n(z,y)}{\mathbf{G}_1(z,y)+\mathbf{G}_2(z,y)+...+\mathbf{G}_n(z,y)} \tag{3.29}$$

where $\mathbf{F}_i$ and $\mathbf{G}_i$ are contributions from user i.

8. Vector addition can be be done in an encrypted format using the scheme discussed in (Canny, 2002b) where an El-Gamal public key (Elgamal, 1985) is known to everyone, and the private key is shared by some d users. The key generation protocol of Pederson (Pedersen, 1991) does exactly this: it enables each user to have a share s_i of the private key s, which can be reconstructed from given sufficient number of shares. The advantage of the El-Gamal encryption process is that multiplicative homomorphism is preserved.

$$E(M_1+M_2) = E(M_1)E(M_2)$$

Thus an addition of two numbers can be performed even if only their encrypted values are available. Using this property, addition of vectors and matrices can be simulated by doing piecewise encryption of each matrix value. Each client therefore uses the public key to encrypt each value of their matrix $\mathbf{F}$, and create a vector of the encrypted values $\mathbf{\Theta_i}^{1\times km}$, such that $\mathbf{\Theta_i}(l) = Enc(\mathbf{F}_1(div(l,m), mod(l,m)+1))$ (concatenating rows to make one large row). Here $Enc()$ is an encryption function. Similarly, another vector $\mathbf{\Omega_i}^{1\times km}$ is created from the encryption of the matrix G.

9. Each client sends its encrypted values Γ and $\mathbf{\Omega}$ to all the *tallier* nodes. Tallier nodes are a subset of the user population which are trusted to perform the vector additions. On receiving the contributions of each user, the talliers compute the addition of the $\mathbf{F}$ and $\mathbf{G}$ matrices.

10. Since homomorphic properties for division do not exist, one needs to decrypt the totals $\prod_i Enc(\mathbf{F}_i)$, and $\prod_i Enc(\mathbf{G}_i)$. To decrypt, the encrypted sums are broadcast to every client which then decrypt these totals using their portions of the keys. The decrypted values are then sent back to the talliers, who them perform an element-wise division of $\sum_i \mathbf{F}_i$ and $\sum_i \mathbf{G}_i$ to compute $\mu_{z,y}$

$$\mu_{z,y} = \frac{\sum_i \mathbf{F}_i(z,y)}{\sum_i \mathbf{G}_i(z,y)} \tag{3.30}$$

When we apply the previously mentioned multiplicative homomorphic property, $\mu_{z,y}$ can be computed as follows:

$$\mu^{k\times m}, \text{ s.t. } \mu_{z,y} = \frac{Decrypt(E(\mathbf{F}_1(z,y))\cdot E(\mathbf{F}_2(z,y))+...\cdot E(\mathbf{F}_n(z,y)))}{Decrypt(E(\mathbf{G}_1(z,y))\cdot E(\mathbf{G}_2(z,y))+...\cdot E(\mathbf{G}_n(z,y)))}$$

[5] An assumption of the key-sharing mechanism is that the input is an integer. Fixed points numbers can be easily used to simulate real number with fixed precision.

11. The newly computed $\mu_{z,y}$ is broadcast to all clients, which is then used to calculate a new matrix $\mathbf{S}$

$$\mathbf{S}_i^{k\times m},\ \text{where}\ \ \mathbf{S}_i(z,y) = \sum_{(u=i,y',v):y'=y} p(z|u=i,y,v)\,(v-\mu_{y,z})^2 \tag{3.31}$$

This matrix is encrypted and converted to a vector $\mathbf{\Lambda}$ which is send to the talliers. There, an encrypted sum is calculated, which is then sent back to clients for encryption (see the two previous steps). Finally, a new value of sigma is computed using the following element-wise division.

$$\sigma_{z,y} = \frac{\sum_i \mathbf{S}_i(z,y)}{\sum_i \mathbf{G}_i(z,y)} \tag{3.32}$$

12. Repeat from step 3, till the values of μ and σ converge. 30-100 iterations may be required. To simulate hold out data, talliers may decide to hold back their own data, and compute their predicted values from the model. By judging the performance of the model in these values, a tallier can make a recommendation to perform another iteration, so to stop. If the majority of the talliers recommend stopping the EM updates, the training phase is over, otherwise the protocol is repeated from step 3.

Phase 2: Recommendation mode

For a new user, using the precomputed model is enough to compute recommendations.

1. Each client initiates the probability distribution $p(z|u)$ over all user communities; for simplicity, we use a random distribution. (See Step 2 of Training phase)

$$\{\mathbf{P}_{z|u}\}^{k\times 1}(l) = p(z=l|u=i)\,,\ \text{such that}$$
$$\sum_z p(z|u=i) = 1\,,\ \text{for all users } i$$

2. Repeat steps 3-6 twice.
3. Compute predicted votes using the equations: $p(v|u,y) = \sum_z p(z|u)p(v|z,y)$. Note that the original profile will renumber the item order due to concatenation of profiles form multiple systems.

Update and Synchronize

When a new item is added to one of the systems (say $\mathbf{A}$), the profile representing the user on that system changes. After the profiles for this system has been updated for all users, the model over A and B also has to be updated. We do this by adding one more dimension to $\mu_{z,y}$ and $\sigma_{z,y}$ each for every user, and initializing it to zero. After that, 2-3 iterations from step 3 onwards of the training phase can be run to update the values of σ and μ.

To update the model using data from new users, a similar procedure has to be followed. Note that in this case, the size of matrices σ and μ, remains the same. Therefore, the model simply has to be trained in a manner similar to using held-out data. To do it in our distributed setting, a new client should simply broadcast its availability, and participate from step 2 onwards. This protocol adjustment however opens the door for malicious users to insert arbitrary data to manipulate the system, which has to be dealt with in the algorithm itself. Robust collaborative filtering extensions are required to take this into account. Section 3.4 provides more details on how to robustify collaborative filtering.

3.3.5 Discussion and Conclusion

We have described three techniques for solving the CSP problem using machine learning techniques. These techniques have been published as the following: (Mehta & Hofmann, 2006a, 2006b, 2006c; Mehta, 2007a) and presented to the research community. The methodology has been accepted as sound and there is an accepted belief that cross system personalization is one of the important issues for recommender system research. The experimental evaluation of these methods are presented in the next chapter and provide empirical proof that CSP offers a significant advantage. There is also a recent focus on *Decentralized User Modeling*, where the above techniques are quite relevant.

3.4 Spam detection in Collaborative Filtering

Collaborative filtering technology is being widely used on the web as an approach to information filtering and recommendation by commercial service providers like Amazon and Yahoo!. For multimedia data like music and video where pure content based recommendations perform poorly, collaborative filtering is the most viable and effective solution, and is heavily used by providers like YouTube and Yahoo! Launchcast. For malicious individuals, or a group interested in popularizing their product, there is an incentive in biasing the collaborative filtering technology to their advantage. Such activity is similar in nature to spam observed widely on the web, e. g. link farms for search engine manipulation.

A lot of electronic systems, especially web-enabled ones provide free access to users via simple registration processes. This can be exploited by attackers to create multiple identities for the *same* system and insert ratings in a manner that manipulates the system. *Profile injection attacks* add a few profiles (say 1-3% of the total profiles) which need to be identified and protected against. Such attacks have also been refered to as *shilling* attacks (Lam & Riedl, 2004), and the added profiles are called *shilling profiles*. Since shilling profiles look very similar to that of an authetic user, it is a difficult task to correctly identify such profiles. Further, profile injection attacks can be classified in two basic categories: inserting malicious profiles which rate a particular item highly, are called *push* attacks, while inserting malicious profiles aimed at downgrading the popularity of an item are called *nuke* attacks (M. O'Mahony et al., 2004). In this work, we focus on detecting push attacks: nuke attacks can be detected using the same methodology.

The current techniques in detection are based on reverse engineered heuristics which perform sub-optimally. In particular, by looking only at individual users and not the combined effect of such malicious users, current detection algorithms have low accuracy in detecting shilling profiles. In this work, we provide an in-depth analysis of shilling profiles and describe new approaches to detect malicious shilling profiles. In particular, we provide an unsupervised algorithm which is highly accurate and fast. We also look in depth at properties of shilling profiles, and analyze optimal shilling strategies which use item means. Note that we concentrate on unsupervised methods since they involve much lesser computational effort as compared to supervised approaches, especially if training data has to be generated. Moreover, we concentrate on those methods which can be easily plugged into existing CF framework.

3.4.1 What Is Spam In Collaborative Filtering ?

Spam is generally perceived as biased information which is forcibly sent to a large number of people to influence them: it is defined as the abuse of electronic messaging systems to send

un-solicited bulk messages, which are almost universally undesired. While the most widely recognized form of spam is email spam, the term is applied to similar abuses in other media: instant messaging spam, Usenet newsgroup spam, Web search engine spam, spam in blogs, mobile phone messaging spam and junk fax transmissions. In collaborative filtering, users tend to provide public feedback to a certain set of items in the form of numerical votes. The pattern of user votes is aggregated to find out most popular items, or more generally, use a subset of users similar to each other for generating recommendations. In this scenario, it is not possible to send information to end users, and lists of recommended items is automatically generated. Therefore the only form of spamming in Collaborative Filtering is manipulation of the underlying algorithm to bias the list of recommended items. We consider this form of manipulation as *Spam in Collaborative filtering*. Previous researchers have used the term *shilling* (Lam & Riedl, 2004), which has its basis in 20th century vocabulary[6]. Mobasher at al (Mobasher et al., 2005) have used the phrase *profile injection attacks* to refer to this phenomena. This clearly emphasizes the fact that user data is added to a recommender system in the form of multiple *user profiles* created by a malicious group of users.

Mobasher et al.. also describe various *models* for generating user profiles. We follow the profile models in this work for generating spam to be added to the collaborative filtering system. Attack strategies include (Mobasher et al., 2005):

Random attacks: where a subset of items is rated randomly around the overall mean vote.

Average attacks: where a subset of items is rated randomly around the mean vote of every item

Bandwagon attacks: where a subset of items is rated randomly around the overall mean vote, and some highly popular items are rated with the maximum vote.

Note that Gaussian distributions $\mathcal{N}(\mu, \sigma)$ have been used for generating most of the random votes rather than the uniform random distribution. This implies that attack profiles have votes near, or equal to the mean vote with a very high probability. Also, standard deviation of the complete vote is used for random and bandwagon attacks, while the standard deviation of the each individual item is used for the average attack. Formally, a shilling profile can be said to consist of three parts

Target Item: One item is chosen in the shilling profile and assigned the highest vote for a push attack, or the minimum for a nuke attack. Usually a set of shilling profiles have a common target item for the attack to be effective.

Filler Items: The set of items which are voted for in the shilling profile; the above cited attack models are used to generate these votes. Typically these values are generated by a random gaussian generator $\mathcal{N}(0, 1)$ and then scaled to the voting range, with a mean μ and variance σ. Depending on the information available, the mean and variance can be varied for different items for higher impact, or kept the same for a low knowledge attack. The filler size is measured as a percentage of the item set. Typical values range from 1% to 10%.

Non voted items: The remaining unvoted items form the majority of the profile.

One more technical term used in the literature and in this paper refers to the size of the attack; this is known as *attack size* and is measured as a percentage of the total user population. The higher the attack size, the more effective the attack is. However, attack sizes tend to be small, because creating profiles comes with a (human) cost, and therefore only a few profiles can be

[6] The wikipedia entry on Shilling at *http://en.wikipedia.org/wiki/Shill* provides a brief history of the term

inserted by one set of people. Typical values range from 1% to 10%. Smaller attacks are also very effective, and often more difficult to detect.

Obfuscation of shilling profiles

To make spam less detectable, researchers have proposed strategies to make the signatures of spam less prominent. Three strategies have been proposed in (Williams et al., 2006) which we also refer to in our experiments:

Random noise: Add random noise to each generated vote.

User shift: Add the same random noise to each vote of the same shilling profile.

Target shifting: Instead of using the highest vote for the recommended items, randomly use the next highest vote (e. g. using 4, instead of 5).

3.4.2 Characteristics Of Shilling Profiles

The common property of all shilling detection algorithms (for collaborative filtering) is that they exploit specific properties of profiles injected in order to identify them. After reverse engineering the profile signatures, appropriate heuristics are used to capture information characterizing shilling users. This is similar to the 2-player model where a move is made by maximizing a certain objective. In order to understand why shilling detection algorithms work, or don't work, one needs to understand the goals of shilling users and methods used to achieve them.

The primary objective of shilling is to *maximize* (or *minimize*, in the case of nuke attacks) the predicted value for the chosen item for the largest possible number of users. This can be achieved by constructing profiles which are highly correlated to a fraction of the users and affect them significantly.

In order to achieve these objectives, profiles have to be constructed in a special manner. Most attack strategies involve rating items around the mean vote, which minimizes the deviation from other existing votes, except the item under attack. Usually, only a subset of the item set is voted on by a shilling profile; this is called the *filler size* and is reported as a percentage of the item space. Filler items are usually selected at random.

Various attack models have been studied [7] in the literature on shilling detection. The results of these studies show that the impact of well constructed profiles can be huge. Even a 1% attack (number of shilling profiles) can skew the system and push the attacked item to top of the ratings. Such attacks are especially severe for items without many votes where shilling users can easily become the authoritative users and force higher ratings. The most effective attack is the average attack where small attack sizes can cause large deviations in the targetted item; it is usually also the most difficult attack to detect. We focus on detecting average attacks in this paper.

The specific construction methods of shilling profiles also have interesting properties, some of which are used by detection algorithms:

1. *Low deviation from mean votes value, but high deviation from the mean for the attacked item*: RDMA (Rating deviation from Mean Agreement) and WDA (weighted degree of agreement) are statistical measures which are based on this idea. The reason for this property is that by placing most votes close to the mean, similarity with other users (based on say Pearson's distance) gets increased significantly.

[7] The interested reader is referred to (Mobasher et al., 2005) for a detailed study on how shilling profiles are constructed.

Table 3.1: No. of neighborhoods that each user belongs to

# *Neigh*	0-20	20-40	40-60	60-80	80-100	100-120
Normal	818	253	62	15	19	9
Shilling	0	1	10	13	17	9

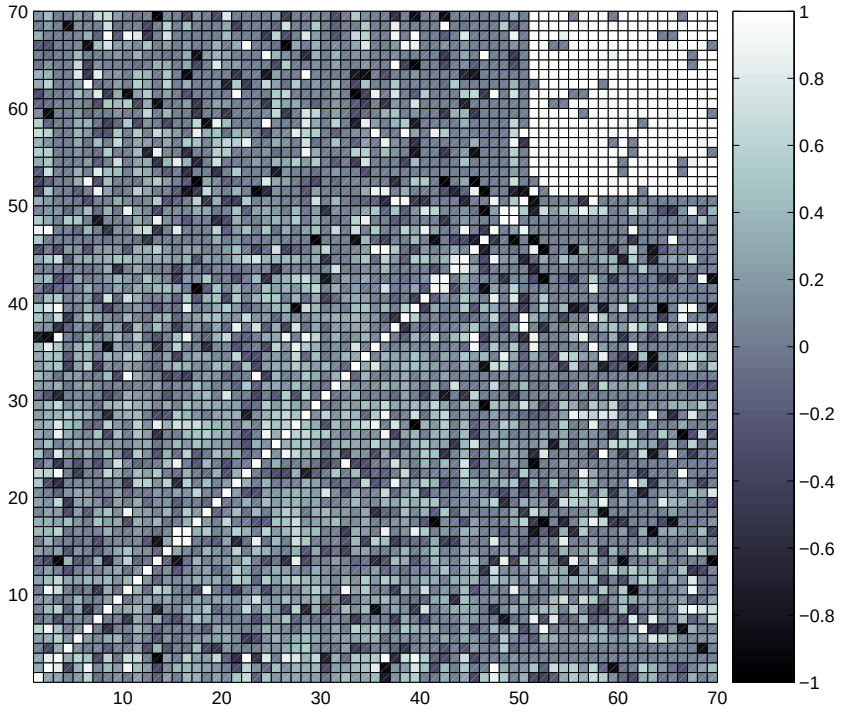

Figure 3.7: Spam users are highly correlated: 50 authentic profiles and 20 shilling profiles are used for calculating the Pearson's Correlation coefficient. Notice how spam users exhibit a noticeably higher degree of correlation

2. *High similarity with large number of users*: Shillers[8] have a high correlation with a significant number of users due to the mean-like votes for most items. A direct result of being highly correlated with a user is that a shiller becomes an authoritative neighbor and figures prominently in the set of k-nearest neighbors. Fig. 3.7 shows the high correlation pattern observed for 20 shillers, compared with 50 normal users.

3. *Shillers work together*: A large fraction of top-20 neighbors for all users are shillers for a well constructed attack. Shillers magnify each other's effect and together push the attacked item to a significantly higher rating. While this is an important characteristic, no algorithms have used this for detection so far. Experiments show that after a shilling attack, the top-20 neighbors of every user are full of shilling users. Table. 3.1 demonstrates these properties for a bandwagon attack[9]. According to our estimates, with a small attack of 3% shilling profiles, approximately 15% (3 out of 20) of every user's closest neighbors

[8] We use the term *shiller* to denote a user which a shilling profile points to.

[9] All data and plots in this section are generated using the MovieLens dataset with 100,000 votes, with 944 users and 1682 movie-items.

are shilling users.

4. *Shillers are highly correlated*: Shillers tend to have very high correlation coefficients (> 0.9) due to the same underlying model which is used to generate them. Average attacks and random attacks have been observed to have this characteristic, and previous work has also used this characteristic to construct a metric which captures the average similarity for top-25 neighbors. Fig. 3.7 also highlights this pattern. Recent work has pointed out some obfuscation strategies that can be used to decrease the average similarity and make the shilling users less noticeable.

3.4.3 Optimal Shilling Strategy

In this section, we discuss what the optimal strategy for a shiller should be while constructing a shilling profile. Assume that the end system **A** has n users and m items. We use the notation $v_{u_i,y}$ for the vote given to an item y by a user u_i, and $\hat{v}_i$ denotes the average vote of a user u_i. $C_{i,j}$ is the Pearson's correlation coefficient between user u_i and u_j.

We assume that the system provides recommendations using Pearson's correlation based collaborative filtering. In this scheme, a user's vote on an unknown/unvoted item is calculated based on the votes of other users who are similar to the current user. In a general scheme, it is also possible to use all users, and weight their opinion with their similarity to the current user. Formally, the predicted vote for user u_i for an item y can be expressed as

$$v_{u_i,y} = \hat{v}_i + \frac{\sum_j C_{i,j}(v_{u_j,y} - \hat{v}_j)}{\sum_j |C_{i,j}|} \tag{3.33}$$

The Pearson's correlation coefficient is calculated according to the following equation:

$$C_{i,j} = \frac{\sum_y (v_{u_i,y} - \hat{v}_i)(v_{u_j,y} - \hat{v}_j)}{\sqrt{\sum_y (v_{u_i,y} - \hat{v}_i)^2}\sqrt{\sum_y (v_{u_j,y} - \hat{v}_j)^2}} \tag{3.34}$$

Note that the correlation coefficient is measure only over items that two users have commonly voted on. Let us add a shilling user s to the user set $\mathcal{U}$. This shilling user wishes to cause item y to be recommended more often. The strategy to do this is to change the predicted value of the item y for as many users as possible. An effective attack would make this value as high as possible. *Prediction shift* is a measure used in literature to measure how effective an attack is. It is defined as the difference in the predicted value of an item before and after a shilling attack.

$$\mathcal{P} = \sum_i \grave{v}_{u_i,y} - v_{u_i,y} = \sum_i \mathcal{P}_u, \tag{3.35}$$

where $\grave{v}_{u_i,y}$ denotes the predicted value of item y for user u_i after an attack and $\mathcal{P}_u$ denotes the prediction shift in user u.

Thus the aim of the shilling user s is to maximize the prediction shift $\mathcal{P}$. Clearly, the attacked item is rated as v_{max} (the maximum allowed rating) by the shilling user to have the maximum deviation. Also, the total shift is maximized when each of the respective prediction shifts $\mathcal{P}_u$ are maximized.

$$\mathcal{P}_u = \frac{\sum_j C_{i,j}(v_{u_j,y} - \hat{v}_j) + C_{i,s}(v_{max} - \hat{v}_s)}{\sum_j |C_{i,j}| + |C_{i,s}|} - \text{const}$$

$\mathcal{P}_u$ can be written as a function of the correlation coff $C_{i,s}$ (replacing $C_{i,s}$ by x) of the form.

$$\mathcal{P}_u = \frac{\kappa_1 + \kappa_2 x}{\kappa_3 + |x|} - \text{const}$$

Note that the correlation coefficient lies in [-1,1] and $\acute{P}_u$ is positive everywhere in [0,1] making P_u a strictly increasing function; the maximum value of P_u is reached at $x = 1$. Thus the overall prediction shift is maximized if the correlation coefficient of the shilling profile is maximized with all the user profiles. If the neighborhood of every user is also limited to a fixed size, then clearly, the impact of the shilling profile is maximum if the shilling user is a part of these neighborhoods. Since the neighbors are formed based on the Pearson's correlation, maximizing the correlation with maximum users is the primary objective of shillers.

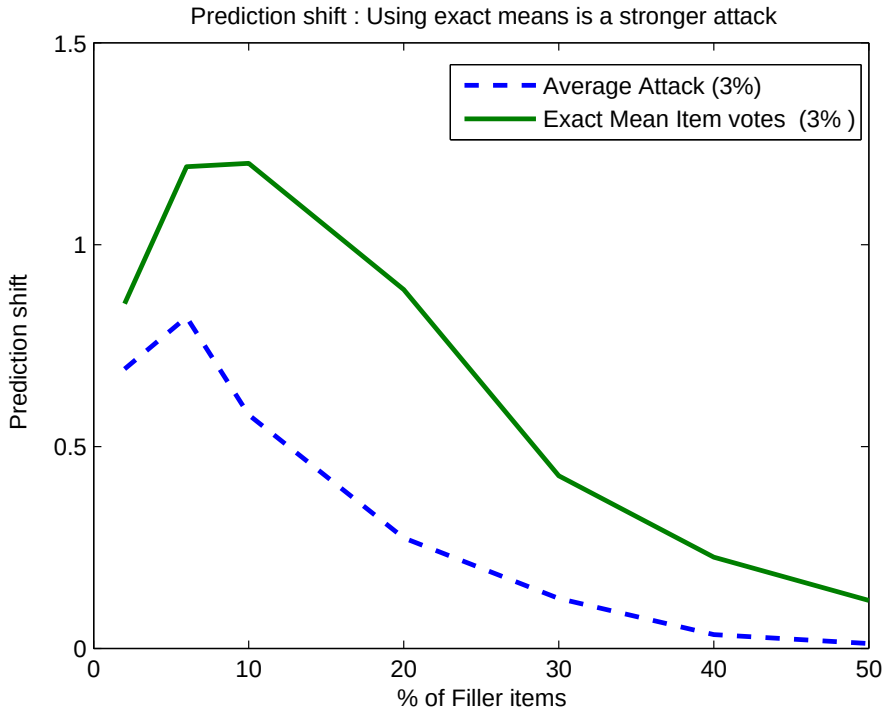

Figure 3.8: Prediction shift as a function of filler size measured over the complete user population(excluding shilling users). Notice how using exact means has a much stronger prediction shift as compared to an average attack.

Maximizing correlation with maximum users

The above analysis shows that a shilling profile must be constructed to maximize correlation with the maximum number of users. Here, we try to motivate the use of mean item votes for maximizing the correlation coefficient. We use concepts used in Canonical Correlation Analysis to analyze the optimal strategy: Canonical correlation analysis seeks vectors a and b such that the random variables $a'X$ and $b'S$ maximize the correlation $\rho = cor(a'X, b'S)$.

Let us construct a random variable $X = (X_1, \ldots, X_n)'$ where X_i represents the i^{th} data (user profile i). Let S represent the shiller's profile. We would like to maximize the correlation between Y and X with the additional constraints that all users are given equal weight. We further constrain $\sum_i a_i = 1, \sum_i b_i = 1$ to avoid scaling. This leads us to use $a = (\frac{1}{n}, \ldots, \frac{1}{n})$. Trivially, $b = 1$. Note

that $a'X$ leads to the average of the user profiles (we represent the mean vote of an item y_i by $\hat{y_i}$).

$$a'X = \sum_i \frac{1}{n} X_i \equiv \hat{X} \sim (\hat{y_1}, \ldots, \hat{y_m}) \tag{3.36}$$

The expression to maximize now is

$$\rho = \frac{cov(\hat{X}, S)}{\sqrt{var(\hat{X}) var(S)}} = \frac{\sum_i (\hat{y_i} - \hat{u})(s_i - \hat{s})}{\sqrt{\sum_i (\hat{y_i} - \hat{u})^2} \sqrt{\sum_i (s_i - \hat{s})^2}}$$

where $\hat{y_i}$ represents the average vote for an item i. and $\hat{u}$ denotes the overall average. It is easy to see that placing $s_i = \hat{y_i}$ maximizes the above expression to make $\rho = 1$. This implies that the optimal strategy for maximizing correlation with all users is to use mean votes for individual items. Attack generation models discussed in (Mobasher et al., 2005) also use this idea for filler votes with the addition of gaussian noise to make the profiles more varied. Note that attacking an item y requires placing the maximum vote for this item; however this does not significantly effect the correlation with other users, since the other votes are still based around the item mean.

Note: This work has been described in (Mehta, 2007b; Mehta & Nejdl, 2007).

Proposed Approaches For Detecting Spam Attacks

Current feature based algorithms tend to pick users with the maximum impact in terms of the measures/features used. However, authentic users who are authoritative and different from many other users can also show significant impact and be falsely classified. Importantly, by working in groups, the effect of shilling users is large only in groups, and individual shilling profiles can be undetectable especially when in small numbers; we call this the *group effect*. Thus it makes sense to eliminate clusters of shilling users, rather than individual shilling users. Below, we outline two algorithms based on this intuition: PLSA is a mixture model which computes a probabilistic distribution over communities (clusters of users) based on latent factors and has been reported to be robust to shilling (Mobasher et al., 2005); PCA is a linear dimensionality reduction model which can be used to select dimensions which are very different, or as in this work, very similar to other dimensions.

3.4.4 Using PCA for Spam Detection

Various algorithms exist for collaborative recommendation with the most successful ones being model based system like PLSA and Sparse Factor Analysis and k-NN algorithms based on Pearson's correlation. All these algorithms consider users similar to a given user (for whom a recommendation has to be generated) and make predictions based on the weights assigned to these similar users. A robust algorithm must modify the weight assigned to malicious user, since a highly weighted malicious users can potentially affect a large portion of the user community. While the most accurate way to robustify an algorithm would be to use a probabilistic model of the trustworthiness of a user, this requires *a priori* knowledge of the distribution of the possible attack, which can be hard to model. To use our idea as a general approach, we categorize users as either trusted, or untrusted. This implies we either completely believe the user or completely ignore the ratings provided by this user.

Current feature-based algorithms tend to pick users with the maximum impact in terms of the measures/features used. However, authentic users who are authoritative and different from

many other users can also show significant impact and be falsely classified. Importantly, by working in groups, the effect of shillers is large only in groups, and individual shilling profiles can be undetectable especially when in small numbers. Thus it makes sense to eliminate clusters of shillers, rather than individual shillers.

Clustering in such a domain is not without problems. The pattern exhibited by normal users is non-uniform and we can at best hope to discover one cluster of highly correlated shillers. Conventional techniques which assume multiple distributions in underlying data cannot successfully deal with such data. Spectral clustering which are graph-based techniques, can deal with such data by constructing a graphical model of data taking pairwise distances into account. However, current approaches deal optimally only with 2-way clustering with clusters of almost equal size, and thresholding to a given size is cumbersome.

Similar problems have arisen in the area of *Gene Clustering* where it is important to identify highly correlated genes from experimental data of patients. The *Gene Shaving* technique (Hastie et al., 2000) is such a technique in use for clustering highly correlated genes from a set of observed data, which includes missing values. The technique is very simple and essentially involves performing PCA and shaving off the top few genes (say 10%) which have the lowest dot product with the principal component. Since the first principal component captures the maximum variance in the high dimensional data, variables with the lowest variance have typically low values along the first few principal components. Therefore this simple idea should be able to exploit the characteristic of the shillers to have low variance, even if the correlation is not extremely high (e. g. due to obfuscation of attack profiles).

A straight forward application of this method is not very successful. While many shillers are correctly detected, many false positives are also detected. Out of the top 100 users detected by gene shaving, only 25% are correctly identified as spam, while others are false hits. We attribute this to the fact that the shilling profiles do not form a clear cluster: shillers are highly correlated to each other *as well as* with normal users. What we want to do is to exploit the highly inter-correlation structure to find a group of shillers which vary least within the group. Collaborative filtering dataset typically have high dimensionality i.eėach observation is multivariate (having multiple variables), where some variables may be unobserved. If a dataset has variables which are very similar and highly correlated, then these variables would uninteresting for a PCA since very low information is added by these highly correlated dimensions. A dimensionality reduction method which identifies the most important and representative dimensions would thus discard these dimensions. Variables which are highly correlated to most other variables would be one of the first to be discarded.

We observe that collaborative filtering data corresponding to shillers is very similar. If we interpret users as variables (i.ėthe dimensions of the data are the users, and the observations are the items), we have data where a number of dimensions are very similar. Thus dimensionality reduction would discard these dimensions since their covariance will be low. A closer look at our data shows us that the covariance between shilling profiles is much lower than normal users. This low covariance is observed not only in between shillers, but also between shillers and normal users. Covariance between normal users is observed to be much higher. This means that PCA of this dataset will compute principal components which are oriented more towards real users who exhibit the maximum variance of the data. We therefore need to select those users (which are viewed as dimensions, by transposing the data) which show the least covariance with all other users. This amounts to selecting some variables from the original data using PCA, which is known in literature as *Variable selection using PCA*.

Variable Selection using PCA

Variable selection using PCA (Jolliffe, 2002) is a much studied area; many researchers have provided algorithms for reducing dimensionality of data by selecting a subset which contains virtually all the information. In our case however, we are interested in selecting dimensions which contain the least information (in the sense of variance). Many effective variable selection algorithms require associating a variable with every principal component, and selecting variables associated with the first m-principal components. Our aim is to achieve the reverse: we seek to detect those users (dimensions) which add the least amount of noise(variance) to data. We apply a simple algorithm to achieve this purpose:

Algorithm 3 PCASelectUsers ($\mathbf{D}$, Cutoff parameter $\mathbf{r}$)

$\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$)
$COV \leftarrow \mathbf{D}\mathbf{D}^T$ {Covariance of $\mathbf{D}^T$}
$\mathbf{U}\lambda\mathbf{U}^T =$ Eigen-value-Decomposition(COV)
$PCA_1 \leftarrow \mathbf{U}(:,1)$ {First Eigenvector of COV }
$PCA_2 \leftarrow \mathbf{U}(:,2)$ {Second Eigenvector of COV }
for all columnid $user$ in $\mathbf{D}$ **do**
  $Distance(user) \leftarrow PCA_1(user)^2 + PCA_2(user)^2$
end for
Sort $Distance$

Output: Return $\mathbf{r}$ users with smallest $Distance$ values

1. Transform the rating matrix $\mathbf{D}$ to a mean centered matrix $\mathbf{D}_0$. One way of doing this is to reduce the ratings to z-scores. We recommend taking raw z-scores without filling missing values with means. *z-scores* can be computed for a user u for a item y, where the user has voted $v_{u,y}$, by using the following equation:

$$z_{u,y} = \frac{v_{u,y} - \hat{v}_u}{\sigma_u}, \tag{3.37}$$

   where $\hat{v}_u$ is the avg vote of the selected user u over observed votes, and σ_u is the standard deviation. Note that the covariance matrix of z-scores is the correlation matrix of the original data.
2. Transpose $\mathbf{D}_0$ to make each user a variable and every item as an observation.
3. Compute the first Principal Component. This requires solving for eigenvalues of the covariance matrix of $\mathbf{D}_0$.
4. Sort variables according to their contribution to the PC in ascending order (higher coefficients are ranked lower). A modified version of this uses the top 3 principal components, and sorts variables in (ascending order) of their total magnitude of contribution to these 3 principal directions.
5. Select the *top*-m variables in this list.

The advantage of using PCA is that it looks at the contribution of shillers as a whole instead of individually. Since shillers exhibit low covariance, the use of PCA is very effective as it looks for principal directions in data which

Cluster	Maha. dist	real users	shillers	Cluster	Maha. dist	real users	shiller
1	707.63	222	1	11	426.36	295	0
2	568.06	250	0	12	575.37	237	0
3	372.31	302	0	13	1195.50	169	0
4	391.40	302	0	14	751.19	212	0
5	273.52	352	0	15	576.48	243	0
6	**225.16**	**240**	**199**	16	441.41	275	0
7	894.96	195	0	17	490.81	265	0
8	348.50	309	0	18	619.65	235	0
9	1156.85	173	0	19	535.85	254	0
10	713.30	217	0	20	539.10	253	0

Table 3.2: A run of PLSA based shilling detection for 200 shillers on a dataset with 5000 users. '*real users*' represents the number of real users in the cluster, and '*shillers*' represents the number of shillers.

3.4.5 Soft clustering using PLSA

Probabilistic Latent Semantics Analysis (PLSA) is a well known approach for text analysis and indexing used to discover hidden relationships between data. It is a highly successful approach for indexing documents and has been well research. Extensions to handle Collaborative filtering are also extremely popular; PLSA enables the learning of a compact probabilistic model which captures the hidden dependencies amongst users and items. It is a graphical model where latent variables are used to render users and items conditionally independent. The hidden variables can be interpreted as a probability distribution over communities of users or clusters; each user is allowed to be a part of multiple clusters, with a certain probability. The patterns in data along with the model fitting algorithm ensure that the learnt distribution minimizes the log-likelihood of the data.

While accuracy has been a well known advantage of PLSA, recent studies have also concluded that PLSA is a very robust CF algorithm, and is highly stable in the face of shilling attacks. (Mobasher, Burke, & Sandvig, 2006) indicates that the prediction shift for PLSA is much lower than similarity based approaches. However, a clear explanation for this has not been provided so far. We investigated the reasons for PLSA's robustness over many experiments and observed the model to understand the mechanisms. The intuition is that PLSA leads to clusters of users (and items) which are used to compute predictions, rather than directly computing neighbors. However this intuition is challenged by experimental results using a k-means clustering algorithm in the same work. Clearly, shilling profiles deceive clustering algorithms due to their high similarity with normal users.

PLSA is a mixture model where each data vector has its own distribution. Membership to a distribution is however not constrained; a data point can belong (probabilistically) to many distributions, with combination weights chosen so that the observed ratings are explained best. This results in *soft clustering* where a data point can lie in multiple clusters. We posit that this is also the reason why shilling is less effective against PLSA: shillers are close to many users, but often dominant in one cluster due to their extraordinary similarity. Since user ratings are more noisy than shilling profiles, the likelihood of user ratings being explained by shilling profiles is limited, though not minuscule. This explanation has also been verified experimentally: we learn a model of an EachMovie data-subset with 5000 users to which 200 shilling profiles are added. On learning a PLSA model with *40* communities, we select the dominant community for each

user; the dominant community for a user is defined as follows:

$$\mathrm{Comm}_u = \operatorname*{argmax}_z P(z|u)\,, \tag{3.38}$$

On experimental analysis, we notice that all the shillers are clustered in either 1 or 2 communities. By correctly identifying this community, we can isolate the shillers and remove them. Table 3.2 shows that this line of reasoning is correct and experimentally verified.

Algorithm 4 PLSASelectUsers ($\mathbf{D}$)

1: $\mathbf{D} \leftarrow$ z-scores($\mathbf{D}$)
2: Train a PLSA model for $\mathbf{D}$.
3: **for all** Users $u \in \mathbf{D}$ **do**
4: $\quad \mathrm{Comm}_u = k$ where $P(z_k|u)$ is maximum
5: **end for**
6: **for all** Community k **do**
7: $\quad U_k \leftarrow$ The set of users u with $\mathrm{Comm}_u = k$
8: $\quad \mathit{Distance}(k) \leftarrow \frac{1}{n}\sum_{u\in U_k}(u-\overline{U_k})^2$
9: **end for**
10: Return r users with smallest $\mathit{Distance}$ values

Output: Return U_k with smallest $\mathit{Distance}$ value

Identifying the community to be removed is vital: noticing how the profiles are close to each other, we have to identify a measure which examines how closely knit a community is: one possibility is to use Mahalanobis distance, which is traditionally used to identify outliers in multivariate data. We suggest using the average Mahalanobis distance of a community as follows: for each community $\mathcal{C}$ which is a set of users, we find the Mahalanobis distance d_u of each user u as

$$d_u = \sqrt{(u-\overline{u})\mathbf{C}_0^{-1}(u-\overline{u})^T}\,, \tag{3.39}$$

where the matrix $\mathbf{C}_0$ is the covariance matrix of the community $\mathcal{C}$, and $\overline{u}$ is the mean profile over the same. Notice how $d_u > 0$ since $\mathbf{C}_0$ is positive semi-definite. We measure the 'closeness' of the community $\mathcal{C}$ by the average Mahalanobis distance over the user-set of $\mathcal{C}$. The intuition is that the cluster containing shilling profiles will be tighter, leading to lower average distances from the centroid of the cluster.

Initial implementation showed that computing Mahalanobis distances is very time consuming due to the inversion of large covariance matrices. To get around this, we observe that a fixed Mahalanobis distance defines a hyper-ellipsoid which is scaled by the variance in observed data in a direction. If variances are assumed to be one, *Mahalanobis* distance reduces to *Euclidean* distance. Based in this observation, we use z-scores (see Eq. (3.37)) instead of actual discrete votes to find the closeness of a cluster, and thus use the simpler Euclidean distance measure:

$$d_u = \sqrt{(u-\overline{u})(u-\overline{u})^T}\,, \tag{3.40}$$

Experimental results (see Table 3.2) have showed that these two measures correlate very well if z-scores are used.

Note: The above approach was published as (Mehta, 2007b). A more detailed analysis of these two approaches was done in (Mehta & Nejdl, 2007).

3.5 Robustness in Collaborative Filtering

The popularity of Recommender Systems has attracted users with malicious intent to bias recommendations in their favor. Other users provide low quality ratings which deviate from the statistical assumptions made by various collaborative filtering algorithms. As a result, there is a danger of producing low quality or faulty outputs from recommender systems which may result in users loosing faith in the system. Recent research has revealed the vulnerability of similarity-based collaborative filtering. While recent algorithms (Mehta, Hofmann, & Fankhauser, 2007; Mehta, 2007b; Mobasher et al., 2005) are successful in identifying spam in collaborative filtering, it is desirable to develop algorithms which are robust to spam from the ground up. A robust collaborative filtering algorithm would provide protection from insertion of random noise as well as attack profiles injected into the system without any explicit detection. Robust statistical methods like M-estimators (Huber, 2004) that have been used successfully in statistics provide an alternative approach when the data has abnormal entries, e.gdue to outliers.

In this work, we propose a matrix factorization algorithm based on robust M-estimators and compare it with various other algorithms. The resulting algorithm provides more stability against spam than previous approaches, but is outperformed by newer versions of SVD in robustness. However, the predictive performance of our proposed algorithm is better than other robust approaches like PLSA and the newly invented SVD based on Hebbian learning.

3.5.1 SVD and Its Variations

SVD stands for Singular Value Decomposition; it is a method of factorizing a matrix into two orthonormal matrices and a diagonal matrix. It stems from the Spectral Theorem (Jolliffe, 2002) which states that a square normal[10] matrix can be decomposed into as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\mathsf{T}}, \tag{3.41}$$

where $\mathbf{U}$ is an Unitary normal Matrix and $\mathbf{\Lambda}$ is a diagonal matrix containing eigenvalues of $\mathbf{A}$. SVD is a more general decomposition than Spectral decomposition since it is applicable to rectangular matrices as well. SVD factorizes a rectangular $n \times m$ matrix $\mathbf{D}$ as follows

$$\mathbf{D} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}, \tag{3.42}$$

where $\mathbf{U}, \mathbf{V}$ are unitary normal matrices and $\mathbf{\Sigma}$ is a diagonal matrix of size $\mathtt{rank}(\mathbf{D}) \leq \min(m, n)$, where $\mathtt{rank}(\mathbf{D})$ is the rank of the matrix $\mathbf{D}$. Moreover, the entries on the diagonal of $\mathbf{\Sigma}$ are in non-increasing order such that $\sigma_i \geq \sigma_j$ for all $i < j$. Note that we may chose to set all singular values $\sigma_i = 0$, $i > k$ for some $k \leq \mathtt{rank}(D)$ (say $k = 10$), leading to an optimal low rank approximation $\mathbf{D}_k$ of the matrix $\mathbf{D}$.

$$\mathbf{D}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^{\mathsf{T}}, \tag{3.43}$$

where $\mathbf{U}$, $\mathbf{\Sigma}$, $\mathbf{V}$ are now $n \times k$, $k \times k$ and $m \times k$ dimensional matrices, respectively. It can be shown that $\mathbf{D}_k$ is the minimizer of $\|\mathbf{D} - \hat{\mathbf{D}}\|_2$ for all matrices $\hat{\mathbf{D}}$ of rank less or equal to k. (Azar, Fiat, Karlin, McSherry, & Saia, 2001) provides more details on properties of SVD. SVD is interesting in the context of many data analysis applications, since real-world data can often be approximated well by a few independent dimensions.

Applications of SVD to Collaborative Filtering assume the representation of user-item ratings by such an $n \times m$ matrix $\mathbf{D}$. Here each of the n users corresponds to a row in the matrix, whereas

[10] A matrix $\mathbf{A}$ is normal if $\mathbf{A}^{\mathsf{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathsf{T}}$, and unitary normal if $\mathbf{A}^{\mathsf{T}}\mathbf{A} = \mathbf{A}\mathbf{A}^{\mathsf{T}} = \mathbf{I}$

the m items are represented as columns, with D_{ij} representing the vote of user i on item j. The application of SVD to $\mathbf{D}$ leads to a low rank estimate $\hat{\mathbf{D}}$, which generalizes the observed data, since it may result in non-zero values $\hat{D}_{il}$, even for user-item pairs (i,l) that are unrated (often set to zero in $\mathbf{D}$, i.e. $D_{il}=0$).

Typically, user-item matrices are very sparse ($\leq 5\%$ non-zero entries) and the presence of a large number of zeros can make the computation of SVD very biased towards unobserved values. Initial applications of SVD to CF such as (B. Sarwar et al., 2000) tried to compensate for that by using the overall means for missing values. This approach, though more successful than previous approaches is highly biased towards the used means. In the last decade, there has been significant research on computation of SVD for large and sparse matrices. Significant work has been done in the design of *PROPACK*[11] and *SVDPACK*[12]. However, these approaches do not treat missing values in a principled fashion, an issue which is discussed in (Ghahramani & Jordan, 1994). (Zhang, Wang, Ford, Makedon, & Pearlman, 2005) discusses the use of the Expectation Maximization (Dempster et al., 1977) procedure to approximate SVD optimally in the log-likelihood sense. However, their approach requires an SVD to be performed at each EM iteration, which cannot be scaled to large matrices, since it is improbable that any method which needs more than a few hundred iterations over the entire data can be scaled to large matrices with millions of rows.

A recent algorithm by Gorrell (Gorrell, 2006) proposed a new approach to computing SVD for virtually unbounded matrices. This method is based on the Generalized Hebbian Algorithm (Sanger, 1989) and calculates SVD by iterating through only observed values. The method has come into the limelight following its use in the *Netflix contest*[13] by a top-10 contestant named Brandyn Webb (who operates using the team name "*Simon Funk*") (Webb, 2006). The advantage of this approach is that it uses a simple Hebbian learning rule which is easily expressed in "*two lines of code*" (Webb, 2006). The method has been found to be highly accurate for CF and scales easily to a matrix with 8.4 billion potential values. Below we describe this approach in detail.

SVD using Hebbian learning

Gorrell (Gorrell, 2006) extends an existing method for Eigenvalue decomposition to multiple eigenvalues with this simple observation: The second eigenvalue of a matrix can be calculated by removing the projection of the previous eigenvalue. This means that if $\mathbf{u}_1$ and $\mathbf{v}_1$ are the first singular vectors corresponding to the largest eigenvalue σ_1, then a matrix $\mathbf{D}_{rem}$ can be defined as follows

$$\mathbf{D}_{rem} = \mathbf{D} - \mathbf{u}_1\sigma_1\mathbf{v}_1^T, \tag{3.44}$$

The eigen-decomposition of $\mathbf{D}_{rem}$ leads to the *second* eigenvalue of $\mathbf{D}$. This observation can be generalized to compute the first k eigenvectors/eigenvalues of a large sparse matrix.

Mathematically the Hebbian learning rule can be expressed as follows: suppose $\mathbf{u}$ and $\mathbf{v}$ are the first eigenvectors being trained for Matrix $\mathbf{D}$, and $D_{ij}=x$. Further, suppose the eigenvalue σ is absorbed into the singular vectors $\mathbf{u}$ and $\mathbf{v}$ to yield $\hat{\mathbf{u}}$ and $\hat{\mathbf{v}}$. The estimate for x would then be

$$x_{est} = \hat{u}_i \cdot \hat{v}_j. \tag{3.45}$$

[11] http://soi.stanford.edu/ rmunk/PROPACK/
[12] http://www.netlib.org/svdpack/
[13] www.netflixprize.com

Since this estimate might have an error, lets suppose further that the residual is represented by $r(x)$.

$$r(x) = x - x_{est} = x - \hat{u}_i \cdot \hat{v}_j \,, \tag{3.46}$$

To get a better estimate of the modified eigenvectors, the Hebbian learning rule updates the value based on the error.

$$\triangle \hat{u}_i = \lambda \cdot \hat{v}_j \cdot r(x) \,, \qquad \triangle \hat{v}_j = \lambda \cdot \hat{u}_i \cdot r(x) \,, \tag{3.47}$$

where λ is the learning rate. It can be shown that with a suitable choice of decaying learning rates, the repeated iteration of the above equations converges to the required eigenvectors. After the first pair of singular vectors has been learnt, their projection can be removed ($x \leftarrow x - u_1 \cdot v_1$) and the next pair can be learnt.

Webb (Webb, 2006) modified this basic algorithm by introducing a weight decay regularization factor:

$$\triangle \hat{u}_i = \lambda(\hat{v}_j \cdot r(x) - \kappa \cdot \hat{u}_i) \,, \qquad \triangle \hat{v}_j = \lambda(\hat{u}_i \cdot r(x) - \kappa \cdot \hat{v}_j) \,, \tag{3.48}$$

where κ denotes the regularization strength. To ensure fewer iterations, he suggests the use of a base estimate using item averages (represented by $\bar{j}$ for item j) and the average user offset. This gives a good estimate and reduces the number of iterations by a factor of 2.

$$x_{base}(i,j) = \bar{j} + \underset{k:x_{i,k}\neq 0}{\text{AVG}} \left\{ x_k - \bar{k} \right\} , \tag{3.49}$$

Further modifications include clipping the estimated value to the permissible range of values. Clipping makes this SVD approach particular to CF, where data values are discrete value bounded by a minimum and maximum rate (say 1-5).

$$\mathbf{D}^{sf}_{ij} = x_{base}(i,j) + \sum_k Clip(\hat{u}_{ik} \cdot \hat{v}_{jk}) \tag{3.50}$$

Where $Clip()$ clips the value to $[1,5]$. Other modifications have also been proposed but have been found to have minor benefits. For the Netflix dataset, $k = 25-40$ has been found optimal. The performance of this simple algorithm is surprisingly good: it performs up to 6% better on the Netflix dataset than the baseline. We have also experienced similar benefits in performance when running this algorithm on other datasets.

3.5.2 Robust Matrix Factorization

Matrix Factorization aims at learning a low rank approximation of a Matrix $\mathbf{D}$ under certain constraints. This technique is often applied in unsupervised learning from incomplete matrices, and is related to SVD. Formally, the problem is stated as follows: Given a non negative matrix $\mathbf{D}$, find matrix factors $\mathbf{G}$ and $\mathbf{H}$ such that

$$\mathbf{D} \approx \mathbf{GH} \tag{3.51}$$

In general, MF can be applied to an $n \times m$ matrix to recover $\mathbf{G}_{n\times d}, \mathbf{H}_{d\times m}$, where $d \ll m,n$. Thus MF is a low rank approximation of $\mathbf{D}$ under some cost function. One such cost function is the Euclidean distance or Frobenius norm

$$\|\mathbf{A} - \mathbf{B}\|_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2} \tag{3.52}$$

Under this cost function, MF reduces to

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} \|\mathbf{D} - \mathbf{GH}\|_F, \tag{3.53}$$

which is a formulation that is equivalent to the SVD, if singular values are absorbed appropriately into the left and right singular vectors.

$$\mathbf{D} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$
$$\mathbf{D} = \mathbf{GH}, \text{s.t. } \mathbf{G} = \mathbf{U}\boldsymbol{\Sigma}^{1/2}, \mathbf{H} = \boldsymbol{\Sigma}^{1/2}\mathbf{V}^T$$

Under other cost functions, MF takes a slightly different form. Assume GH_{ij} is the (i,j)-th element of the matrix $\mathbf{GH}$. Then, for a real valued even function ρ, the MF problem is restated as

$$\underset{\mathbf{G},\mathbf{H}}{\operatorname{argmin}} \sum_{ij} \rho(D_{ij} - GH_{ij}) \tag{3.54}$$

The formulation $r_{ij} = D_{ij} - GH_{ij}$ has also been used in the literature. r_{ij} is known as the *residual* of the fit. Clearly, if $\rho(0) = 0$, the above minimization has a lower bound of 0, when $\mathbf{D} = \mathbf{GH}$. The *least square* formulation corresponds to $\rho(x) = x^2/2$.

Robust approximation using M-estimators

In many real world scenarios, the observed matrix $\mathbf{D}$ is prone to erroneous values. In addition to some small noise, some values may be out of range, or unexpectedly different from the rest of the observations. Such values are typically called outliers; note that we are assuming outliers at a cell level, meaning individual observations D_{ij} might be faulty, with completely arbitrary values and random distribution of cells. Least squares estimates have been shown to be highly error prone to outliers: even 1-2 erroneous values can completely disrupt the approximation. Fig. 3.9 shows the effect of one outlier on a linear least square estimator. A lot of research has been done in the last 35 years on the topic of robust regression. The theory suggests that minimizing the squared residual is not stable: instead a function of the residual should be minimized. This is done by the use of *M-estimators* which are based on bounded real valued functions $\rho(r_i)$

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \sum_{ij} \rho(r_{ij}) \tag{3.55}$$

where θ representing the model fitting parameters, with θ^* being the optimal value. Let us assume that ρ is a differentiable function, and its derivative is represented by ψ. The minimization of the above function w.r.t the model parameters θ occurs when the derivative of the above equation is zero, i.e;

$$\sum_i \psi(r_i) \frac{\partial r_i}{\partial \theta} = 0\,, \tag{3.56}$$

$\psi(x)$ is called the *influence function* of the M-estimator ρ and models the influence of a residual on the model fitting. It is postulated that robustness requires a bounded influence function. Clearly, ordinary least squares, which has an unbounded influence function ($\psi_{LS}(x) = x$) is non-robust following this criteria. To further simplify, let us define a weight function $w(x) = \psi(x)/x$. Then Eq. 3.56 becomes:

$$\sum_i w(r_i) r_i \frac{\partial r_i}{\partial \theta} = 0, \tag{3.57}$$

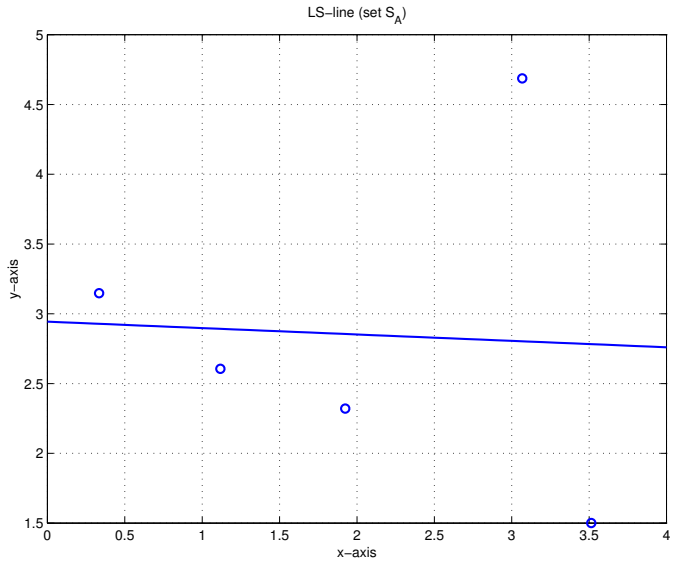

Figure 3.9: The effect of a single outlier on the least squares estimate.

which is exactly the same condition required for solving the following iterative reweighted least square problem:

$$\underset{\theta}{\operatorname{argmin}} \sum_{i} w(r_i^{k-1}) {r_i}^2 \tag{3.58}$$

The final issue remaining is the choice of an M-estimator: various M-estimators have been described in literature, with Huber, Andrew and Tukey estimators being more popular. Huber's M-estimator (Huber, 2004) is recommended for general purposes and is characterized by the following weight function:

$$w(r) = \begin{cases} r \leq k & 1, \\ r > k & \frac{k}{|r|} \end{cases} \tag{3.59}$$

In Eq. (3.59), k is an appropriately chosen constant. For our application, we choose $k = 1.345$, a value reported to work well for normally distributed data with $\sigma = 1$; note that standard deviation in our dataset is 1.118 (~ 1). The influence function of the Huber M-estimator is bounded by $|k|$. The Huber weight function also has distinct computation advantages over other M-estimators; its application results in a dampened effect of large errors, providing more robustness. In case of spam meant to cause large deviations in one item's predicted value, we expect robust regression to discourage large shifts and provide a moderate estimate.

Robust Matrix Factorization using M-estimators

Robust regression problems have been studied in a linear setting where observables Y and inputs X are known and Y is assumed to be noisy. Previous work shows that Matrix fitting problems can be performed in a similar manner using an Alternating fitting scheme. Assume we want to find the rank over factors $\mathbf{G}_1, \mathbf{H}_1$ as defined in Eq. 3.54, with the Huber M-estimator; higher rank estimates can be easily computed in a similar manner to SVD (see Sec. 3.5.1). For a rank-1

solution where $\mathbf{G}, \mathbf{H}$ are both vectors, the broad outline is as follows: first, we initialize $\mathbf{G}_1, \mathbf{H}_1$. Then we fix $\mathbf{G}_1$ and minimize the reweighed least square problem:

$$\underset{H_k}{\operatorname{argmin}} \sum_{ij} w(D_{ij} - G_k H_k) \cdot (D_{ij} - G_k H_k)^2 \tag{3.60}$$

This can be achieved by a fixed rate gradient decents algorithm, where updates are performed as follows:

$$G_i^{k+1} = G_i^k + \eta \cdot r_{ij}^k \cdot w(r_{ij}^k) \cdot H_j \, , \forall D_{ij} > 0 \tag{3.61}$$

Note that we use the function r_ij to denote the residual at $D_{i,j}$. After a few iterations, $\mathbf{G}_1$ converges to a minimum. At this point, we switch $\mathbf{G}_1$ and $\mathbf{H}_1$, and minimize for $\mathbf{G}_1$. The above scheme is known as *Iteratively Reweighted Least Squares* and was proved to converge to the rank-1 least squares estimate of the matrix (Gabriel & Zamir, 1979). For higher rank matrices, the above routine is repeated for the matrix $\mathbf{D}^k = \mathbf{D} - \mathbf{G}^k\mathbf{H}^k$, for $k = 1, \cdots, d$, to get a k-rank estimate. Algorithms 5 and 6 summarize the above procedure.
Note: This algorithm is presented in (Mehta, Hofmann, & Nejdl, 2007)

Algorithm 5 Rank-1-estimate $(\mathbf{D}_{n\times n})$

1: Initialize $\mathbf{G}^0, \mathbf{H}^0, k \leftarrow 1$.
2: Define $r_{ij} = D_{ij} - (G^k H^{k-1})_{ij}$.
3: Solve for $\underset{\mathbf{G}^k}{\operatorname{argmin}} \sum_{ij} w(r_{ij})(r_{ij})^2$
4: Solve for $\underset{\mathbf{H}^k}{\operatorname{argmin}} \sum_{ij} w(r_{ij})(r_{ij})^2$
5: $k \leftarrow k + 1$
6: Iterate steps 2, 3 and 4 till convergence.
7: $\mathbf{G}_1 = \mathbf{G}^k, \mathbf{H}_1 = \mathbf{H}^k$

Output: Matrices $\mathbf{G}_1, \mathbf{H}_1$

Algorithm 6 Rank-K-estimate $(\mathbf{D}_{n\times m}, K)$

1: Initialize $\mathbf{G} \leftarrow \mathbf{0}_{n\times k}, \mathbf{H} \leftarrow \mathbf{0}_{k\times m}, k \leftarrow 1$.
2: Define $\mathbf{D}_{rem}^k = \mathbf{D}$
3: **while** $k \leq K$ **do**
4: $\quad \mathbf{g}, \mathbf{h} \leftarrow$ Rank-1-estimate$(\mathbf{D}_{rem}^k)$
5: $\quad \mathbf{G}(:, k) \leftarrow \mathbf{g}$, $\mathbf{H}(k, :) \leftarrow \mathbf{h}$
6: $\quad \mathbf{D}_{rem}^{k+1} \leftarrow \mathbf{D}_{rem}^k - \mathbf{G}(:, k)\mathbf{H}(k, :)$.
7: $\quad k \leftarrow k + 1$
8: **end while**

Output: Matrices $\mathbf{G}, \mathbf{H}$, Residual error$= \|\mathbf{D}_{rem}^k\|$

Related Work

Robust statistics have been applied previously to SVD (Liu, Hawkins, Ghosh, & Young, 2003) using L-estimators. Liu et al. uses alternative least squares with an L_1 minimization. RANSAC based methods have also been developed for SVD (Li, Ning, & Xiang, 2005). There is plenty of

work in the application of robust statistics to regression (Huber, 1964; Gabriel & Zamir, 1979) and least square estimates. However, all the above approaches for SVD have been designed with full matrices in mind. Moreover, the objective in the work above to deal with numerically large outliers. In our domain, the erroneous values are still in the permissible range; however their effect to cause large deviations which we want to guard against. The use of M-estimators for Matrix factorization is novel to the best of our knowledge; the RMF approach outline above is also meant to work with large and sparse matrices.

3.5.3 Discussion and Conclusion

Spam in Collaborative Filtering is a recent but important problem, which has been attacked by many researchers. However, all previous approaches looked at single shilling profiles instead of a *group*, an observation which is key to our solution presented in the previous section. Our proposed solutions are simple and work with extremely high accuracy. One key issue though with this approach is that it is an offline approach; addition of new data might require a re-computation of the principal component. This however, might be a direct result of our variable selection procedure; discovering other possible strategies for variable selection (using PCA) is a part of future work.

We have also investigated the effectiveness of robust statistics in protecting against collaborative filtering spam. Experimental results show that application of M-estimators does not add significant stability to collaborative filtering; modified SVD algorithms outperform RMF in robustness (details in Sec. 4.5). In addition, we have explored the effectiveness of vote sampling on stability and performance; removal of 20% of extreme votes leads to a significant increase in robustness for every method. In Chapter 4, we put our algorithm to the test and conclude that robust statistics is not sufficient against spam in collaborative filtering; however robust methods can provide a more accurate recommendation algorithm.

4 Evaluation

> ...it doesn't matter how beautiful your theory is, it doesn't matter how smart you are – if it doesn't agree with experiment, it's wrong. –
>
> (R.P. Feynman)

4.1 Evaluation Plan

The algorithms proposed in this thesis fall in two broad categories: the first category aims at learning mappings between user profiles, and the second category aims at detecting shillers and robustifying recommender systems. For the purposes of evaluation, we limit ourselves to collaborative filtering based recommender systems: the reason for this is the availability of standard datasets like *EachMovie*[1] and MovieLens[2] which provide user ratings over a set of movies. These standard datasets have been used by several researchers which make it possible to compare the results with others. We use both these datasets for evaluating robustness as well, albeit measuring different performance metrics.

The setup of the experiments conducted to evaluate our proposed approach is very specific and differs greatly from the collaborative filtering literature. Specifically for the first set of algorithms, the setup is very unique and has not been performed before. For shilling detection algorithms, we use the same setup as in recent literature (Chirita et al., 2005; Mobasher et al., 2005; M. P. O'Mahony et al., 2006); this is also different from the standard collaborative filtering setup. In this chapter, we first briefly describe the relevant metrics; we also provide details about the experimental setup in the beginning of each section.

4.2 Evaluation of Learning methods for CSP

The evaluation of CSP methods is aimed at testing how accurately we can predict user profile information in a multi-system setup. Clearly, it is imperative to have such data where the *same* users occur in multiple systems. To our best knowledge, such data is unavailable, and the only option would be to collect data. Given that the effectiveness of CSP would be visible only if a large number of users (likely, *thousands* of users) cross from one system to another, the magnitude of such a data collection process is out of scope of this thesis.

To avoid this challenge, but still being able to perform relevant evaluation, we chose to use existing datasets and to *simulate* multiple systems. The principle behind the simulation performed here is to simply divide the data and assume that two (in the simplest case) systems instead of one contain this data. In case of movie data, this is akin to having two data collections: *one* about genres like comedy, romance, and action, and *another* about genres like independent,

[1] EachMovie is not publicly distributed anymore
[2] http://www.grouplens.org/

horror, drama, thriller etc. Note that these two *systems* can now be setup to have common users; for users unique to only one system, only a part of the original data set is used. The aim is clearly to predict the withheld data and compare it with data available in the original dataset. To give a concrete example, let us assume we want to simulate a two system scenario using the Movielens dataset with 944 users and 1682 movies; Fig. 4.1 depicts this setup. The following steps are then performed:

- Randomly divide the item set of 1682 movies into two roughly equal parts (say 840 & 842) and call them $\mathbf{A}$ and $\mathbf{B}$.
- Randomly select 5% of the items set and make sure that both $\mathbf{A}$ & $\mathbf{B}$ have these items in common. Assume this results in $\mathbf{A}$ having 870 items and $\mathbf{B}$ with 892 items. Note that it is not necessary to have an equal number of items.
- We now assume that a fraction of the user population c is common to both systems. For the rest, the data is removed; the goal would be to predict this missing data.

We now have the situation described in Eq. 3.8, which is reproduced here for convenience:

$$\mathbf{X}^A = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A \end{bmatrix}, \qquad \mathbf{X}^B = \begin{bmatrix} \mathbf{X}_c^B & \mathbf{X}_s^B \end{bmatrix},$$

which can be combined as a missing value problem as:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_c^A & \mathbf{X}_s^A & ? \\ \mathbf{X}_c^B & ? & \mathbf{X}_s^B \end{bmatrix}$$

4.2.1 Experimental Setup

We concentrate on the two system setup described above: for evaluating the effectiveness of a learning method, we start from a situation of minimal user overlap between two systems and increase this gradually, while measuring how well the predicted user profile vectors compare to the observed data. To measure the goodness of fit, we rely on the following metrics:

1. *Mean Average Error* $= \frac{1}{m}\sum_v |p_v - a_v|$, where p_v is the predicted vote and a_v is the actual vote, and m is the number of observed votes over which the MAE is measured. The average is taken only over known values (assume the active user has provided m votes).

2. *Ranking score of top-*N *items.* $R_{score} = \frac{100 * \sum R}{\sum R_{max}}$. Ranking scores were introduced in (Breese et al., 1998) and have been discussed previously in Sec. 2.4.3. We choose the only the top 20 items instead of the entire set of rated items. The choice of 20 items is also based on the fact that we have picked users with at least 20 votes, and would like to evaluate this metric over observed votes only.

To get a sense of how well our method is performing, we define a baseline approach of the *popularity voting*. This is defined simply as predicting the vote on an item as its average vote in available data. Based on this, one can select the most highly rated items on an average and recommend these to a user. Often for a new user, a collaborative recommender system has no other option. It has been observed in literature that this naive approach performs well in general; in fact the best collaborative filtering algorithms cannot provide more than 10-12% improvement[3]. CSP has to perform better than popularity votes to be a feasible approach.

[3]This problem is so grave that Netflix, an online movie rental site, has started a contest to extend improvement to 20% over baseline.

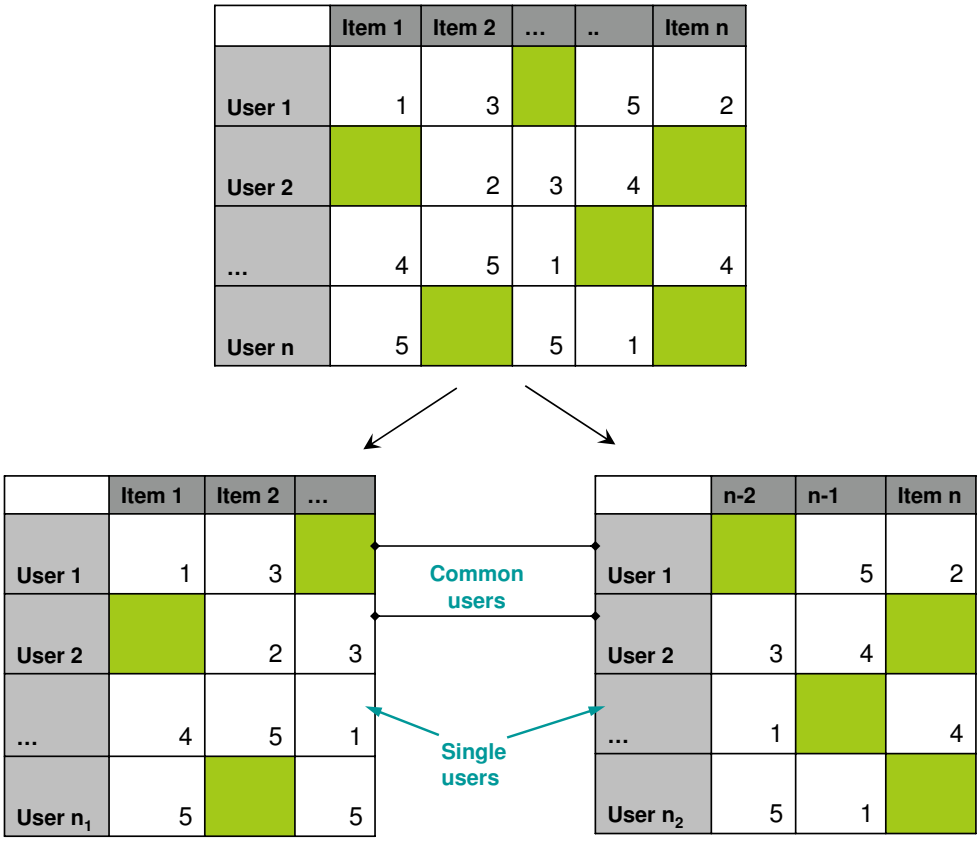

Figure 4.1: Division of data matrix into two parts simulating a 2-system setup. Notice how some users are kept in correspondence, while others are kept as single users.

Often, we also define a *gold standard* which is a more sophisticated approach and offers an advantage over the method being evaluated. Often the gold standard has more favorable conditions, e. g. more data, relaxed assumptions etc, and serves as a measure of what is the best possible performance. In our case, we could assume that the best prediction possible might be if all the data lies with one system; we could then predict the missing votes using a standard CF algorithm like k-NN using Pearson's correlation (see Sec. 2.4.2).

4.3 Evaluation Results for CSP

Below, we evaluate three algorithms for CSP outlined in Chapter 3: Manifold Alignment, Sparse Factor Analysis, and Distributed PLSA. The setup in evaluation of all three methods is similar; we have evaluated two additional scenarios in the later sections where 3 systems are assumed and CSP is also evaluated for existing users. The evaluation showed that SFA and PLSA are both effective and scalable approaches, which perform better than Manifold alignment. While the idea behind manifold alignment is novel, the method does not scale very well and is best suited for smaller datasets. We therefore use different datasets for Manifold Alignment as compared to the other two methods.

4.3.1 Manifold Alignment

Hypothesis to test: We aim to prove that Manifold alignment can outperform the baseline recommendation of popularity votes when users from two systems are put in alignment with one another.

We chose the *MovieLens* dataset with 100,000 votes for the purposes of our evaluation. This data set consists of votes on 1682 items by 944 different users. This data is quite sparse ($\sim 6\%$) as is typical for user ratings. We split the data into two subsets $\mathbf{A}$ and $\mathbf{B}$ by spliting the movie ratings for all users (e. g. two matrices 840×944 and 842×944). In principle, the overlap between datasets can be varied ranging from no overlap to all items overlapping. While in the earlier case, the movie ratings are effectivly split into half, the complete data is available to both systems in the second case. However, in real world scenarios, item overlaps are very small. Therefore we chose a random 5% from the itemset as an overlap. The other free parameter is the number of users set to be in correspondence, which we vary from 0 to 800. The last 144 users form the test set for our evaluations. We randomly choose the test set and the item set for every run of the algorithm. Individual NLDR methods ($i.e.$ LLE and Laplacian Eigenmaps) have other parameters which need to be varied in order to judge their effect. These parameters are (a) the dimensionality of the manifold, (b) the size of the neighborhood for the adjacency matrix, and (c) the size of the neighborhood for the user profile reconstruction. Additionally, the Laplacian Eigenmap method has a free parameter β which can take any real value. In our experiments, we have varied the parameter and present the results for the optimized values. Further increases in neighborhood sizes offers some advantage, but at a much increased computational cost. We have chosen these values: the number of nearest neighbors $k = 36$, the dimensionality of the manifold $d = 6$, and size of neighborhood on the manifold $k_1 = 55$. In addition, we choose different values of heat kernel parameter β, namely $0, 0.4$ and 4. The results of the experiment are shown in Fig. 4.2.

Discussion

The results of the evaluation are encouraging. A simple NLDR to a manifold even with any explicit alignment of user profiles performs better than popular voting. Expectedly, the predicted votes become more accurate as more users cross over and their profiles are aligned. While the predictions are not as good as the gold standard even in the case of complete overlap according to the MAE, the algorithm provides a $4-5\%$ improvement over the baseline after $\sim 35\%$ user profiles have been aligned. For collaborative filtering, this is not an insignificant improvement: the gold standard is only 12.6% better than the baseline. Experimental results also show that the top-N recommendation using manifold alignment is a significantly higher quality than the baseline. In case of complete overlap, Laplacian Eigenmap based manifold aligment can provide a $\text{top} - 20$ ranked list which is more relevant than the gold standard, thus suggesting the strength of Manifold Alignment as a stand-alone collaborative filtering algorithm. The results presented here are obtained after a $10-$fold validation; in some cases, the algorithm was able to outperform the gold standard for MAE as well. One possible reason for the lower performance is the small size of data which is very sparse. Due to the sparsity of data, the majority of the normalized user database consists of mean. Therefore, the reconstructed values are heavily weighted towards the mean votes, especially for items that are note frequently rated. Previous research (Ghahramani & Jordan, 1994) has shown that learning from incomplete data offers significant advantage over strategies like mean imputation. Given that our approach works better than popularity votes even with a heavy bias towards mean values, algorithmic enhancements

which offer a probabilistic interpretation to manifold alignment are likely to be more accurate.

Implementation and Performance

The manifold algorithm outlined in Sec. 3.3.1 has been implemented using Matlab R14 on a Pentium 4 based Desktop PC. Standard Matlab routines have been used and sparse matrices are used wherever possible. For the smaller MovieLens data with 100,000 votes, the algorithm uses around 100 MB of RAM. It performs reasonably w.r.t. to time as well. Each run of the *Algorithm* 1 followed by *Algorithm* 2 runs in approximately 5 seconds using Laplacian Eigenmaps. The LLE algorithm runs slower (70 seconds) since a quadratic program has to be solved for every point. The memory requirements of the LLE algorithm are also higher.

4.3.2 Sparse Factor Analysis

Hypothesis to test: We aim at testing 3 hypotheses with Sparse FA:

1. CSP offers an advantage over *popular item voting* for a large number of first time users,
2. CSP offers an advantage for existing users in a standard collaborative filtering setting, and
3. CSP offers an advantage in a n-system scenario.

We choose the EachMovie[4] data with ratings from 72,916 users for 1,682 movies. Ratings are given on a numeric six point scale (0.0, 0.2, 0.4, 0.6, 0.8, 1.0). The entire dataset consists of around 2.8 million votes, however around 2.1 million of these votes are by the first 20,000 users. We chose this dense subset of 20,000 users and 1682 movies and scaled ratings to integers between 1 and 6. We split this data set into two parts to form two datasets by splitting the item set of the entire data. This way we get two datasets with the same number of users, but with ratings over different items. To mimic a real life setting, we allow a random 5% of items to overlap between the data sets. The overlap is not explicitly maintained nor is the correspondence information made available to the learning algorithm. Moreover, we choose 10,000 test users, since this model is useful only if it works for a large number of users, with only a few correspondences known. In our test runs, we build an FA model using the matrix $\mathbf{X}$ (see Eq. (3.8)) varying c from 500 users to 10,000 users. For the users not in correspondence, we randomly rearrange their order. In our setting, it is vital that we can build an effective predictive model with as few users crossing over from one system to another which works effectively for a large number of new users. We randomly choose the test set and the item set for every run of the algorithm. In addition, we also performed the model building step using only the users common to both systems using $\mathbf{X}_c$ (see Eq. (3.9)).

Results and Discussion

In order for CSP to be useful, we require CSP to provide a perceptible advantage over status quo. Clearly, a new user who has profiles with other systems should be able to use his/her previous ratings to get a meaningful recommendation. This advantage should increase when the number of systems in the CSP setting is higher (i. e. many users have profiles in n systems and provide this information when moving to a new system), or when the user already has a profile at the current system.

Fig. 4.3. provides experimental evidence for the first hypothesis. While popular voting has been reported to provide a good recommendation quality, here it performs very poorly. The

[4] http://research.compaq.com/SRC/eachmovie

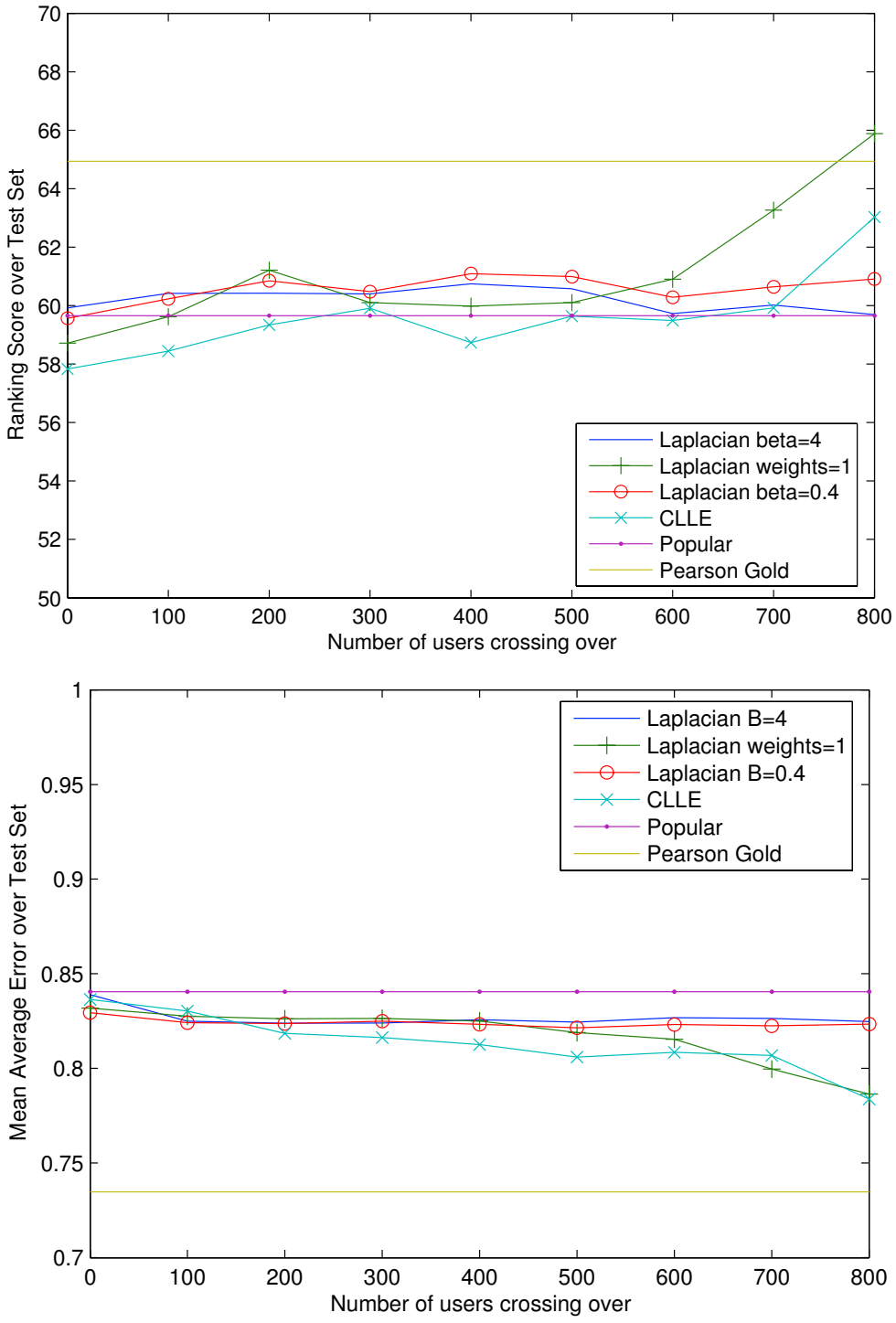

Figure 4.2: Precision and MAE for the different NLDR methods within the manifold alignment framework. Numbers plotted are after 10-fold validation and averaging

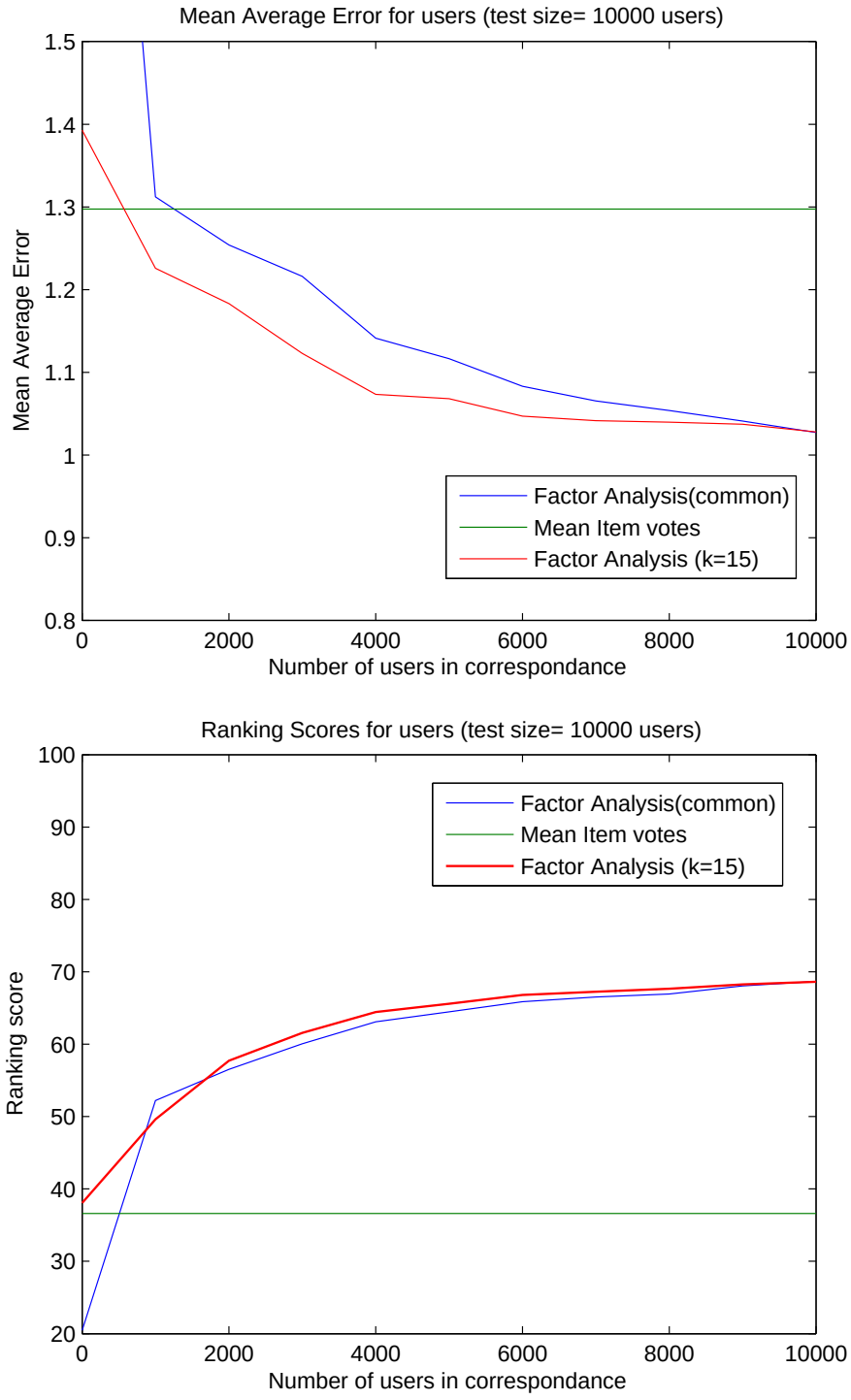

Figure 4.3: MAE and Ranking scores for 10,000 test users (with 10-fold validation) for SFA. "common" refers to the use of only common users (Eq. 3.9) for training the model.

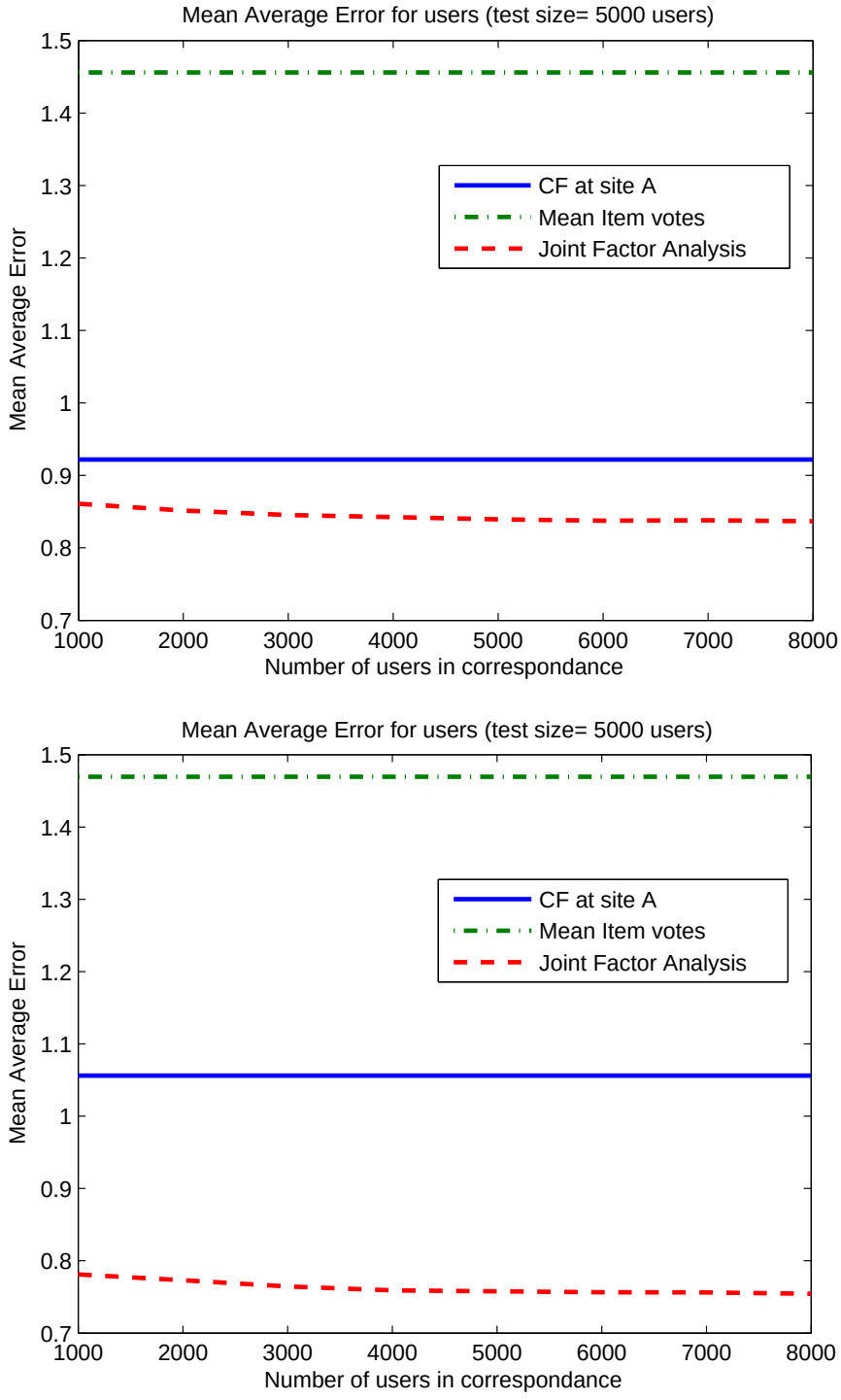

Figure 4.4: MAE and Ranking scores for 5000 test users (with 10-fold validation)

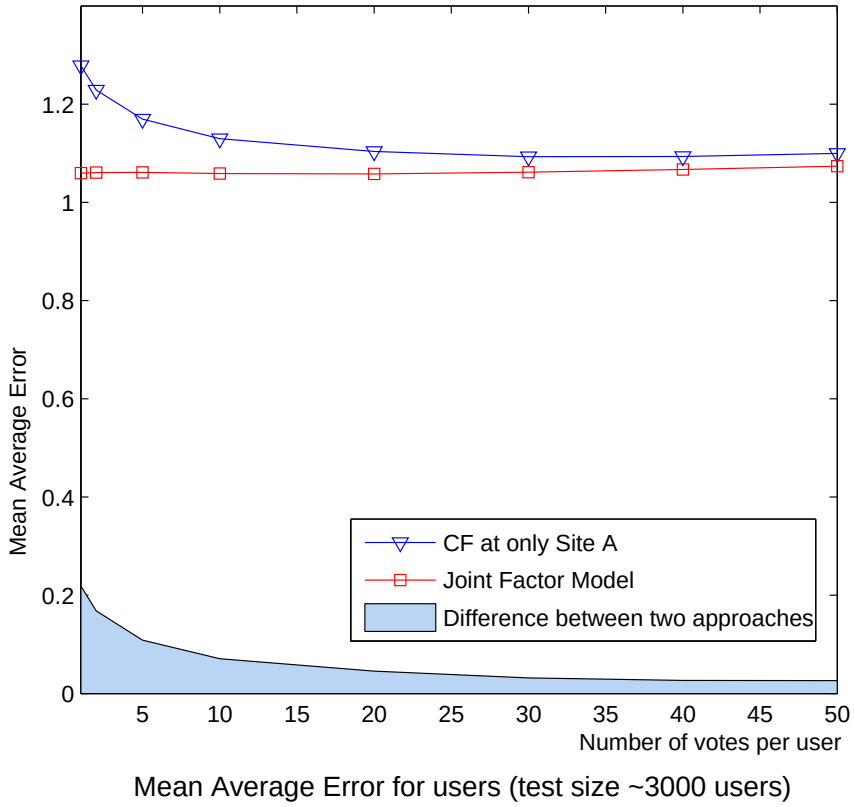

Figure 4.5: MAE and Ranking scores for $\sim 3,000$ test users (with 10-fold validation) in an *Only*-n scenario. n is varied from 1 to 50.

simple reason for this is our large number of test users; clearly the same rating and ranking can not appeal to a large variety of users. In contrast, the CSP approach offers significantly better results, performing 12% better than the baseline when 4,000 users have crossed over. When the complete set of training users are set in correspondence, this advantage jumps to 22%. Note that these numbers are not insignificant for a collaborative filtering setting, where the best performing k-Nearest Neighbor algorithms perform only 12-16% better than popularity voting.

Not surprisingly, the *semi-supervised* approach to CSP performs better than the *supervised* approach. In the supervised approach, only the users with known correspondences are used for training the model. While this approach is clearly more efficient, the early advantage gained by semi-supervised learning here is to the tune of 6% when 4,000 users have crossed. Subsequently, both approaches perform similarly, but this is because the input data to both starts to look much similar and is actually the same when all the training users have been set in correspondence. Clearly, more available data helps to make the model learn faster.

The advantage offered by CSP is even more evident in the Ranking score metric, which depicts the utility of the ranked recommendation list returned to the user. This metric was originally proposed in (Breese et al., 1998) and bases itself on the observation that the utility of an item in the ranked list decays exponentially, with a half life of, say 5 items. We have measure this metric only over the observed votes. As Fig. 4.3 shows, the popularity voting offers a very low ranking score in comparison with the CSP approach. The advantage is more than 70% when around 4000 users have crossed over, and continues to grow a further 15% till all the training set has been used.

To test the second hypothesis, the following setup is used: the entire set is split into 2 parts as earlier. At site **A**, 10,000 users are used to train a model of just site **A**. This model is then used to generate recommendations for 5,000 test users, and then evaluated using the *All-But-1*, *All-but-5*, and *Only*-n protocols (see (Breese et al., 1998) for details on these protocols). In addition, a second model is trained with 8,000 users (total), out of which some users have profiles at Site **B**. The number of such profiles in correspondence is increased from 1000 to 8000. Now, this second model is used to generate recommendation for 5,000 test users *for site* **A**. Notice that the test users have their complete profiles from site **B** available, in addition to some ratings for site **A**. Fig. 4.4 provides evidence for this hypothesis, where the MAE for *All-but-1* and *All-but-5* show an advantage for CSP even for existing users. Clearly, this advantage is higher when a lower number of votes for a user are available at site **A**. Fig. 4.5 shows the results of an experiment where the number of votes available at site **A** (per user) was varied from 1 to 50. As test users, we chose only those users who had cast at least 50 votes in our test set of 5000 users. Over various runs, this number of test users was noted to be around 3000. It is clear that CSP has a big early advantage over single-system collaborative filtering to the tune of 20%. This advantage decreases to 5% when 20 votes per user are available. At the 50-votes per user point, the advantage is less than 3%.

To test the third hypothesis, we constructed a setup with 3 systems in the following way: for 4,000 users, corresponding profiles from all 3 systems are available, for 2,000 users, profiles for systems 1 and 2 are available, and for the next 2,000 users, profiles from system 1 and 3 are available. Now for 5,000 test users, we tested the 3 following scenarios for new users at system 1:

1. No profile is available
2. Profile from system 2 is available
3. Profile from system 2 and 3 is available.

	Popular vote	**Only from A**	**From A,B**
MAE	2.6052	1.1725	1.1142
RS	14.5585	61.8124	66.4900

Table 4.1: MAE and Ranking Score for 3-system scenario. Each system had an non-overlapping item set of size 500.

With this setup, we constructed a joint factor model and evaluated the predictions of the model in the cases stated above. Table 4.1 shows the MAE and Ranking Scores for the n-system scenario. The results clearly show that in an n-system environment, CSP offers a definite advantage. When a new user brings along a profile from another system, there is a dramatic improvement in performance. The numbers in this experiment for MAE are higher than the 2-system cases due to the decreased size of the item sets for each system. Finally, a new user who brings profiles from two systems has a bigger advantage than a user make available his/her profile from only from one system.

Implementation

The factor analysis algorithm has been implemented on a standard Pentium IV based PC running Matlab R14. Canny's original Matlab code has been made available publicly[5], and this served as a starting point for our purposes. The core code is very quick and scales very well to a large data set like EachMovie. Running times for the model building phase with 10,000 users is around 40 seconds, and less than 10 seconds when only common users are used. Prediction times are much faster: recommendations of all 10,000 test users are generated in 6.5 seconds. We have used $k = 14 - 20$, and have found that there is negligible difference for values higher than 14: hence we have reported results for $k = 14$.

4.3.3 Distributed PLSA

Hypothesis to test: We aim to prove the following hypotheses:

1. CSP using *PLSA* offers an advantage over mean item voting for a large number of first time users,
2. CSP using *PLSA* offers an advantage over existing methods like SFA.

We choose the EachMovie data with ratings from 72,916 users for 1,682 movies. Like with SFA, we chose the dense subset of 21835 users and 1682 movies and we scale these ratings on a scale of 5. To simulate two systems $\mathbf{A}$ and $\mathbf{B}$, we divide this data set into two parts by splitting the item set of the entire data. In our experiments, we have used 15,000 users for both A and B, with 8,000 users being common between the two systems. To mimic a real life setting, we allow a random 5% of items to overlap between the datasets. The overlap is not explicitly maintained. In our test runs, we build a PLSA model using the matrix $\mathbf{X}$ (see Eq. (3.8)) varying c from 1000 users to 8000 users. For the users not in correspondance, we randomly rearrange their order. We refer to this case as the *full* data case. In our setting, it is vital that we can build an effective predictive model with as few users crossing over from one system to another which works effectively for a large number of new users. We use 5000 users as test (randomly from the 7000 users not common to the systems). In addition, we also performed the model building step using only the users common to both systems using $\mathbf{X}_c$ (see Eq. (3.9)). We refer to this case as the *common* data case.

Results

The experimental bench described above sets the scene: PLSA models and SFA models are trained over identical datasets, and MAE and Ranking Scores are measured. Results are then averaged over 5 runs and plotted in Fig. 4.6. For the SFA model training, we use an improved implementation (w.r.t. (Mehta & Hofmann, 2006c)) which is optimized w.r.t. model parameters and reports better results than previously. SFA remains a fast and effective model; however, we expect PLSA to outperform SFA.

Fig. 4.6 provides experimental evidence: PLSA has a distinct advantage with smaller training data and provides highly accurate recommendations for 5000 test users *even* when only 1000 users have crossed over. While SFA also outperforms the baseline *most popular*[6] method, it catches up with PLSA only after more than 7000 users have crossed over: even then PLSA maintains a slight lead. The results in the ranking score experiment show an advantage for

[5] http://guir.berkeley.edu/projects/mender/

[6] The most popular strategy recommends the most highly rated(on an average) items.

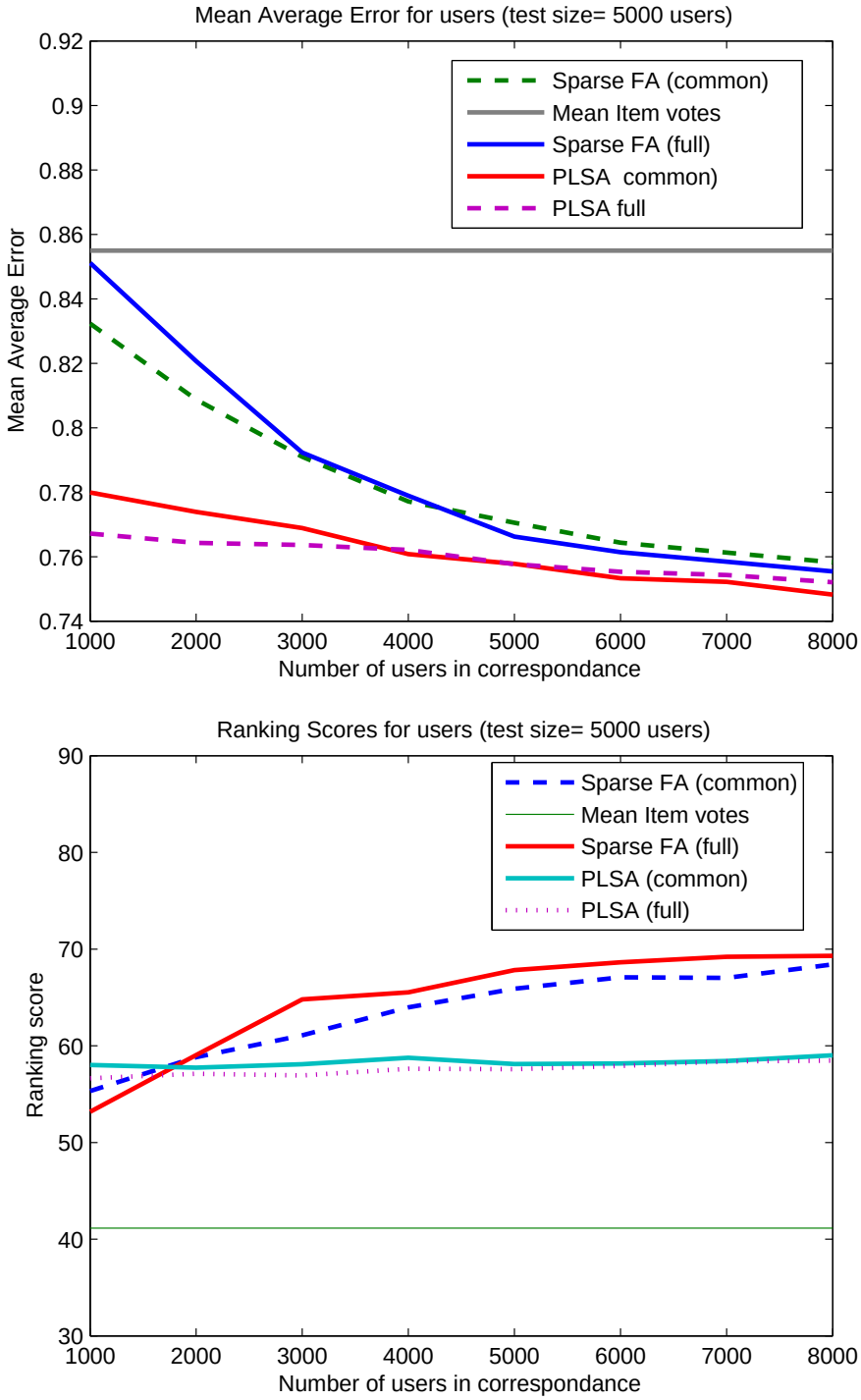

Figure 4.6: MAE and Ranking scores for 5000 test users (with 5 fold validation) with Distributed PLSA. "*common*" refers to the use of only common users (Eq. 3.9) for training the model.

Sparse FA over PLSA: this means that while PLSA is an overall more accurate method, Sparse FA is able to pick the top 20 relevant items and rank them better than PLSA. A lower *Mean Average Error* for PLSA shows that the complete profile predicted by PLSA is closer to the original profile than the one predicted by SFA. One more important observation is that the models trained with only common data (supervised) outperform the models trained with full data (semi-supervised). However, this trend is observable only when a small number of users are common to both systems. Once around 4000 users have crossed over, the semi supervised methods have a small lead. In a practical situation, we might use only the common users, since the overhead of training this model is much smaller than the full data.

Implementation

The Sparse Factor Analysis algorithm has been implemented on a standard Pentium IV based PC running MATLAB. A highly optimized version of PLSA has been implemented in Java 5 using optimized sparse Matrix libraries. The core code is very quick and scales very well to a large data set like EachMovie. Running times for the model building phase with 10,000 users is around 80 seconds, and less than 30 seconds when only common users are used. Prediction times are much faster: recommendations of all 10,000 test users are generated in 1.5 seconds. We have used $k = 40 - 80$, and have found that there is negligible difference for values higher than 80: hence we have reported results for $k = 80$.

4.3.4 Conclusions

The experimental results provide empirical evidence that CSP offers a significant advantage over the state of the art. All three proposed methods for CSP perform better than baseline, with SFA and PLSA performing significantly better. Additionally, all three methods provide some support for privacy preservation, and SFA and PLSA also handle sparse data in a principled sense. However, only our distributed PLSA method supports update and synchronization in case of new items or users being added.

The performance of the proposed methods is good enough for performance not to be a challenge for the future. When cast as a missing value problem, other model based methods for CF can also be adapted to the CSP task. The difficult part however is overcoming the challenges mentioned in Sec. 3.2.1 namely privacy, robustness and synchronizing with multiple systems. This thesis provides *a* solution to CSP which tackles all the above challenges. However, there is scope for improvement for in terms of a solution, specifically one, which can combine semantic information and user provided rules in addition to collaborative filtering user profiles.

4.4 Evaluation of Shilling detection

Hypothesis to test: We aim to experimentally show that PCA and PLSA based user selection methods described in Algorithm 3 & 4 can outperform existing methods (Mobasher et al., 2005).

4.4.1 Experimental Setup

To evaluate the performance of our proposed algorithms, we use the Movielens dataset which consists of 100,034 votes by 944 users over 1682 movies and has been used previously for evaluating spam detection. To this data, shilling profiles are added which all target the same item which is selected at random. Shilling profiles are generated using the well studied models

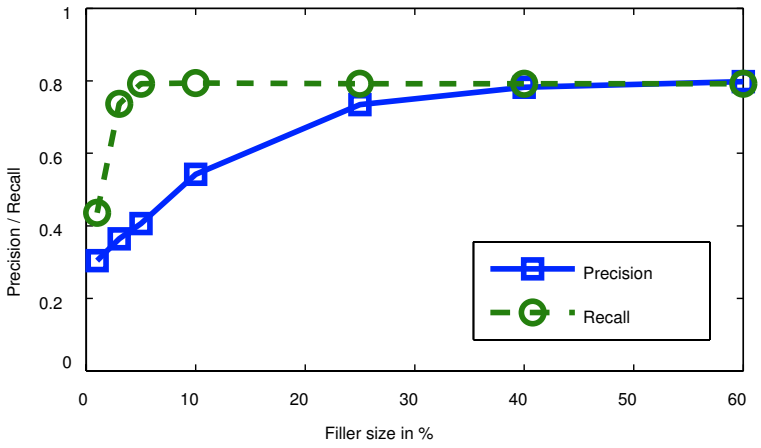

Figure 4.7: Detection Recall and Precision for Average Attacks of varying filler sizes using PLSA based detection. 10% shilling profiles were added to 944 normal users.

of Average, Random and Bandwagon attacks. We use the generative models explained in (Mobasher et al., 2005) which add Gaussian noise to item or overall averages. The parameters of the profile injection attacks are the *attack size* and *filler size* (Sec. 3.4.1 describes these terms). The main task in these experiments is to detect as many of the inserted spam profiles as possible. The results of the detection process are measured by the standard metrics of *precision* and *recall* which are defined as follows:

$$\text{precision} = \frac{|\{\text{relevant profiles}\} \cap \{\text{detected profiles}\}|}{|\{\text{detected profiles}\}|}$$

$$\text{recall} = \frac{|\{\text{relevant profiles}\} \cap \{\text{detected profiles}\}|}{|\{\text{relevant profiles}\}|}$$

Experimental results have been found to hold on larger datasets like EachMovie: we present results on the 100k MovieLens dataset to be directly comparable with other reported results.

Evaluation Results

4.4.2 PLSA based spam detection

Experimental results for PLSA based detection show that spam profiles are indeed clustered together: in most experiments, all shillers end up in one cluster. Moreover, using the *closeness* measure also works well in most cases. For medium and large sized attacks (see Fig. 4.7), more than 70% attackers are correctly identified. However the precision is low as many normal users are also misclassified. We find 20 communities to be ideal for the selected dataset, which makes each cluster between 2-10%. For very small filler sizes (% of rated items) and attack sizes (no. of profiles inserted), low recall and precision are observed. Also in 20% of the cases (2 out of 10 trials), the wrong cluster is selected, leading to maximum 80% recall and precision on an average. This experiment also explains the robustness of PLSA against shilling: the effect of spam is large only in the cluster where most spam users are. For all other clusters, the prediction shift is much lesser as the effect is weighted by the cluster weight of spam users, which is usually a small fraction. However, for large attack sizes, we note that large clusters are formed with a majority of the users in the same cluster as spam users, hence explaining the large prediction shift reported in (Mobasher et al., 2006).

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	96.0	96.0	100.0	94.0	96.0	98.0	92.0
2%	96.0	98.0	99.0	98.0	97.0	99.0	99.0
5%	97.6	97.6	98.0	97.6	98.0	98.4	97.6
10%	97.4	98.4	98.6	98.8	98.6	98.6	98.6

Table 4.2: Detection precision for Random Attacks of varying sizes

Drawbacks of PLSA based spam detection: PLSA based spam detection works well against strong attacks like the average attack and exploits the extraordinary similarity between spam users; however, for weaker attacks spam users tends to be distributed across different communities and hence are impossible to isolate. Similar trends are noted for very small attacks, where 1-3% of the user population is added. Moreover, for weaker attacks, very low precision and recall are observed, meaning that throwing out a cluster can lead to the loss of many real users. The reason for this is that PLSA is a parameterized algorithm, where the number of communities (clusters) is taken as input (usually between 20-40). The expected size of a cluster is around 50 for our dataset. Clearly, smaller attacks would lead to clustering where spam users are a part of a larger group of users, thus the characteristics of normal users would dominate in that group. Thus, PLSA based detection works well in certain conditions, like large attacks, and fails for small and weak attacks. We still think this is a strong algorithm because using PLSA for recommendation can lead us to inspecting suspicious clusters where extreme similarity is observed, and this is done at low additional cost.

4.4.3 PCA based spam detection

To evaluate the performance of our PCA-Variable selection based algorithm, we use the Movielens dataset as earlier. To this data, spam profiles are added which all target the same item which is selected at random. Spam profiles are generated using the well studied models of Average, Random and Bandwagon attacks.

The results of applying this algorithm are very positive: PCA coefficients shows clear clusters of data, which are even more evident if visualized in 2D (1st and 2nd PC), as shown in Fig. 4.9. While the clusters are very clear for unsophisticated attacks (see Fig. 4.9), more sophisticated attacks do not significantly alter this property. With 5% attack profiles, the top 5% eliminated users are spam users with more than 90% accuracy. Similar numbers are observed for a variety of attacks (*random, average, bandwagon*) in variety of attack sizes, and filler sizes. Further accuracy is gained by considering the first three Principal components, and sorting the variables *together*. To do this, assume the three coefficents of each variable represents a point in $\mathbb{R}^3$ and sort the points in order of their distance from origin. Our experiments show an improved performance as compared to using single PCs. Using further PCs doesnt add to the performance. All numbers we report below use the first three principal components to do variable selection.

An additional test was performed to evaluate the effectiveness of the PCA-VarSelect Algorithm when faced with an uncoordinated attack. In this setting, we introduce attack profiles which attack on random, possibly different items, and also are produced from different attack models. A sample of 100 such profiles may contain 45 average attack profiles, 30 random attack profiles and 25 Bandwagon attack profiles. The attacked item may be different for different attack profiles and each profile may be performing either a push or a nuke attack. In the real world, many different items maybe simultaneously be attacked, by different attacks and a spam detection

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	90.0	92.0	94.0	96.0	90.0	80.0	68.0
2%	95.0	96.0	95.0	93.0	89.0	86.0	80.0
5%	97.6	98.0	96.8	96.8	94.8	92.8	89.2
10%	97.6	97.8	97.6	97.0	96.8	95.6	92.4

Table 4.3: Detection precision for Average Attacks of varying sizes

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	78.0	88.0	94.0	94.0	96.0	98.0	100.0
2%	82.0	88.0	90.0	97.0	95.0	95.0	98.0
5%	88.0	93.60	94.40	96.80	96.0	98.0	98.0
10%	87.0	94.20	96.40	97.60	98.40	98.20	98.40

Table 4.4: Detection precision for Bandwagon Attacks of varying sizes

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	74.0	84.0	82.0	80.0	78.0	68.0	64.0
2%	76.0	87.0	90.0	89.0	87.0	83.0	76.0
5%	87.2	92.0	92.4	93.6	92.8	91.2	84.4
10%	85.8	93.2	96.0	95.8	95.0	95.2	89.4

Table 4.5: Detection precision for Average+Bandwagon Attacks of varying sizes

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	94.0	100.0	90.0	100.0	94.0	98.0	96.0
2%	99.0	97.0	97.0	97.0	96.0	96.0	96.0
5%	97.6	98.0	98.0	98.0	97.2	98.8	97.6
10%	97.8	98.6	98.6	98.4	98.4	98.2	98.6

Table 4.6: Detection precision for Obfuscated Random Attacks of varying sizes. Three kinds of Obfuscation strategies have been used: random noise, user shift and target shifting

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	96.0	94.0	90.0	90.0	84.0	74.0	60.0
2%	96.0	95.0	95.0	93.0	89.0	81.0	76.0
5%	96.8	95.6	96.4	95.6	92.4	91.2	82.0
10%	97.4	96.4	97.4	96.8	96.2	92.6	86.6

Table 4.7: Detection precision for Obfuscated Average Attacks of varying sizes. Three kinds of Obfuscation strategies have been used: random noise, user shift and target shifting

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	78.0	80.0	84.0	96.0	100.0	94.0	100.0
2%	79.0	87.0	92.0	94.0	97.0	96.0	95.0
5%	85.6	92.8	93.2	95.6	96.8	98.0	97.6
10%	85.8	94.0	96.2	97.0	98.0	98.2	98.2

Table 4.8: Detection precision for Obfuscated Bandwagon Attacks of varying sizes.

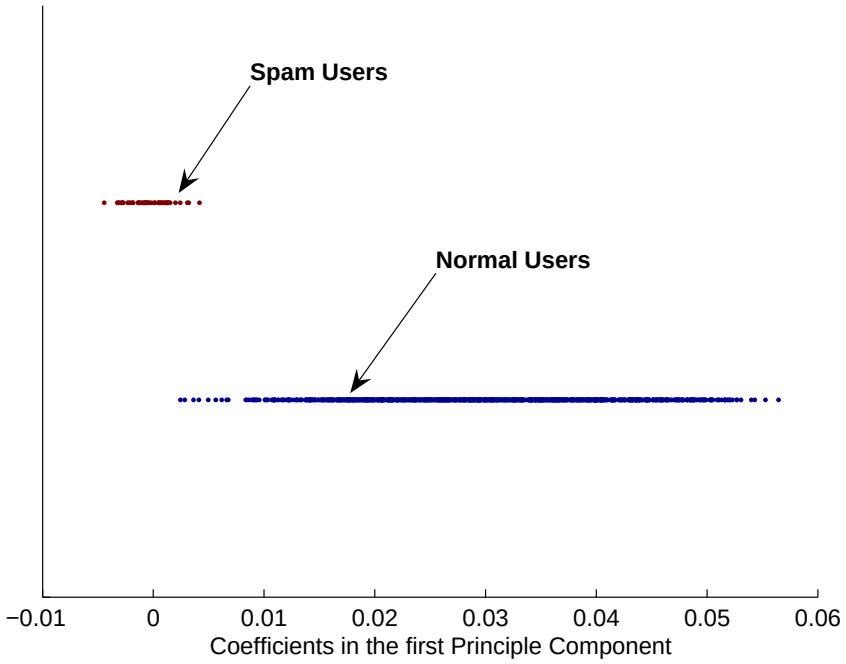

Figure 4.8: Cluster in the first PC space for all users. The coordinates here are the coefficients of each variable (user) in the 1st principal component

Table 4.9: Detection precision for Obfuscated Bandwagon+Average Attacks of varying sizes. Three kinds of Obfuscation strategies have been used: random noise, user shift and target shifting

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	68.0	72.0	80.0	78.0	72.0	64.0	56.0
2%	76.0	81.0	83.0	87.0	83.0	79.0	63.0
5%	84.0	85.2	88.4	90.4	87.6	84.4	79.6
10%	82.6	89.8	91.8	92.8	92.4	89.2	80.8

algorithm should still perform well against such attacks.

Our evaluation shows that PCA-varselect can still successfully detect such attacks, although at a lower precision than with coordinated attacks. This experiment shows that as long as profiles are constructed to have low deviation from mean (which is necessary to increase similarity with a large number of users), PCA-VarSelect can exploit the low covariance and high correlation between such users to detect them effectively.

Discussion

The results of the evaluation clearly show the effectiveness of PCA-VarSelect in identifying densely correlated users. As one can see from Fig. 4.10., the f-measure of the selection procedure is near ideal, with the maximum f-value observed when all spam users have been correctly identified. Fig. 4.11 shows the f-measure for a hard-to-detect attack (an obfuscated bandwagon + average attack), where the highest f-measure is significantly lower than shown in Fig. 4.10. Not surprisingly, the impact of obfuscated attacks is lower than the standard attack

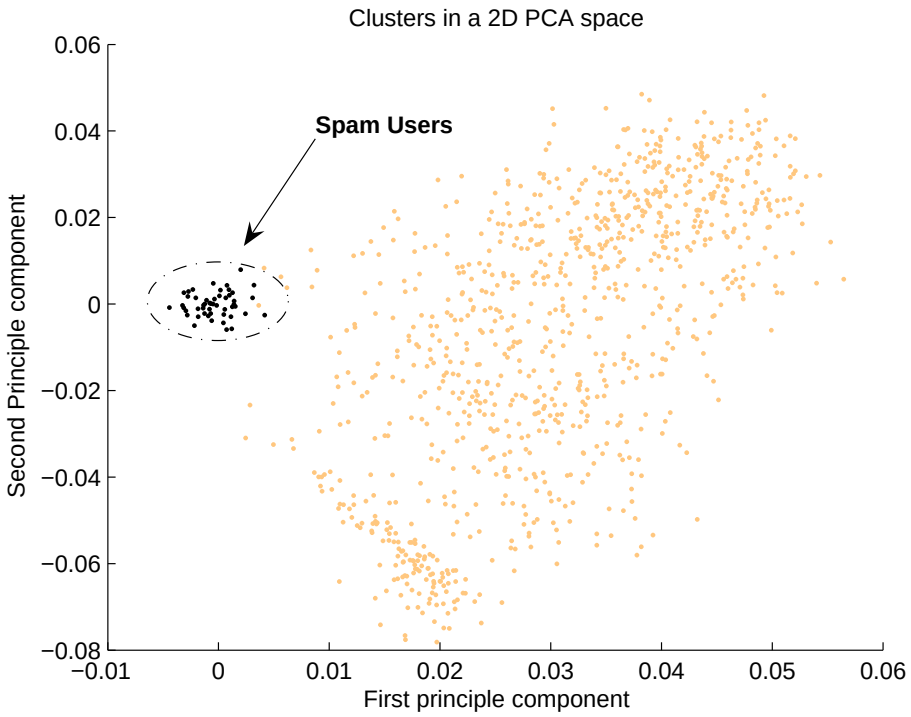

Figure 4.9: Cluster in 2D space for all users. The coordinates here are the coefficients of each variable (user) in the 1st and 2nd principal component. Notice that the spam users are centered around the origin.

attack size	1% filler	3% filler	5% filler	10% filler	25% filler	40% filler	60%filler
1%	74.0	86.0	82.0	86.0	80.0	78.0	72.0
2%	78.0	85.0	85.0	88.0	83.0	83.0	78.0
5%	82.8	89.2	92.8	92.4	92.8	88.0	83.6
10%	85.8	92.0	94.2	94.2	94.8	92.8	89.8

Table 4.10: Detection precision for a mixture of uncoordinated Attacks of varying sizes and types.

models (Williams et al., 2006). Clearly, stealth comes at the price of lower impact.

A comparison with other reported algorithms shows a clear advantage for PCA-based selection. While the Chirita et al. (Chirita et al., 2005) algorithm works well for large filler sizes, it fails in more realistic settings like small attack sizes and small filler sizes(see (Mobasher et al., 2005) for a comparison of the *Chirita et al.* algorithm and the *Burke et al.* algorithm). The Burke et al. (Mobasher et al., 2005) approach is based on a large set of features which exploit the characteristic properties of spam users. However, the detection procedure results in a large number of false positives. Table 4.11 compares the reported performance of (Mobasher et al., 2005) with PCA-Varselect. One advantage of these approaches over PCA-VarSelect is that they are not thresholded. PCA-VarSelect needs a parameter which specifies how many spam users need to be detected. At higher attack sizes, the effectiveness of PCA-VarSelect may be lower; however, such scenarios are unlikely in collaborative filtering. Outlier detection works in general when the number of outliers is significantly less than authentic data. In this scenario, outlier

Precision	*Average Push Attack*		*Random Push Attack*	
	Burke et al.	**PCA**	Burke et al.	**PCA**
1% filler	22	90	26	96
1% obfuscated	22	92		94
5% filler	23	92	28	100
10% filler	29	96	29	94
20% filler	32	90	33	96
40% filler	39	80	40	98
60% filler	42	68	47	92

Table 4.11: Detection precision for Push Attacks of size 1% at different filler sizes compared with other algorithms. Numbers for the *Burke et al.* algorithm have been reported from (Mobasher et al., 2005)

detection can be still applied since scenarios of more than 25% spam profiles being inserted is unrealistic. The reason for this is that there is an intrinsic cost asociated with inserting a spam profile (registration, voting systematially, avoiding multiple identity detection etc) which makes it difficult to insert arbitrary amounts of spam, unlike email spam where costs of sending spam email are not very high.

Another drawback of PCA-Varselect is that it fails to work well when spam profiles are not highly correlated. In this case, the spam profiles also have limited effect since the impact of a spam profile is high only when it is similar to a number of users. Therefore, low-quality spam may not be very well detected by this method.

4.4.4 Conclusions

Based on our experimental findings, PCA based user selection performs better than PLSA based detection. A comparison with other reported algorithms shows a clear advantage for PCA-based selection. The Burke et al. (Mobasher et al., 2005) approach is based on a large set of features which exploit the characteristic properties of spam users. However, the detection procedure results in a large number of false positives. Table 4.11 compares the reported performance of PCA vs the *Burke et al.* approach. However, drawbacks of both approaches do exist: our PLSA based approach identifies the correct cluster only 4 out of 5 times, and has low recall and precision against smaller attacks. When 50 shilling profiles were injected, the recall and precision were both around 25% lower than the reported numbers for detecting 100 profiles. Adding 1-3% profiles only results in zero recall. Clearly, smaller attacks are harder to detect. PCA based detection is more stable against attack size, but does not perform as well when attack profiles are not highly correlated. In this case, the attacks also have limited effect since the impact of a shilling profile is high only when it is similar to a number of users. Therefore, low-quality shilling data may not be very well detected by this method.

4.5 Evaluation of Robustness in Collaborative Filtering

Hypothesis to test: The aim of our experiments is to test whether robust statistical methods can be used to robustify collaborative filtering. The RMF method outlined in Sec. 3.5.2 should withstand profile injection attacks in order to be useful.

To test this hypothesis, we apply RMF to CF data and compare the performance with the

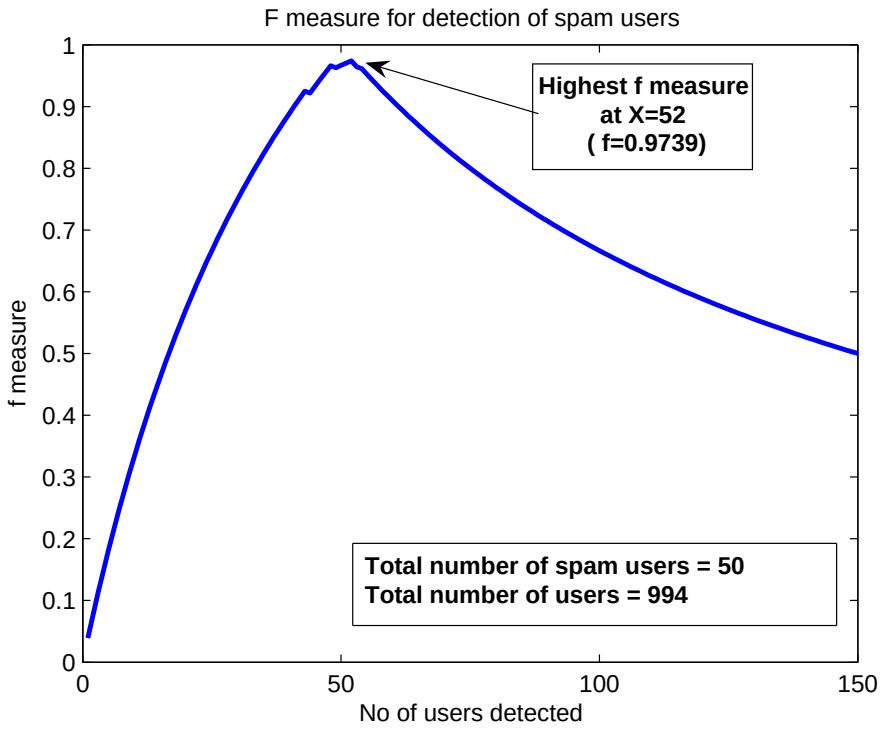

Figure 4.10: f-measure for detection algorithm run on an Average Attack of 5% size with 3% filler size

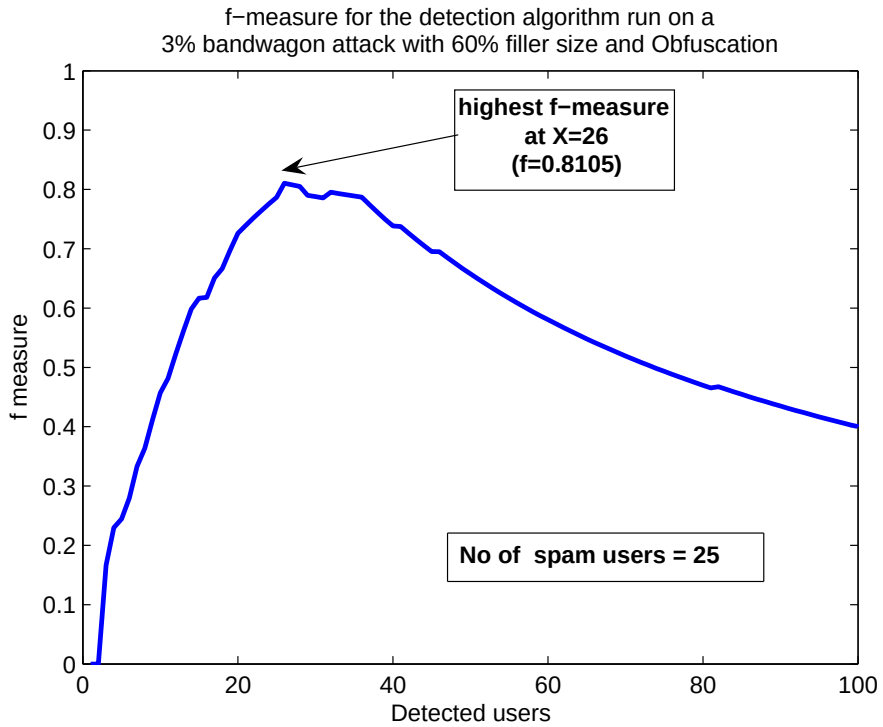

Figure 4.11: f-measure for detection algorithm run on an 3% Average + Bandwagon Attack of 3% size with 60% filler size

prediction accuracy after insertion of attack profiles. To insert attack profiles, we use the average attack model (Mobasher et al., 2006) and generate a certain percentage of profiles. These profiles collate to attack a single item, which is decided earlier. To choose items to attack, we use the following filter: an item which has not been voted by more than 5% of the user propulation and has an average vote of less than 3 (since our data set has votes between $1-5$). We then vary the number of profiles inserted and the number of items voted by the spam user (filler size). All measurements of error are made on 10% of the original data which has not been used for training/prediction; this is called the *test set*. This methodology is standard and been used to measure the effectiveness of spam attacks previously (Mobasher et al., 2006, 2005; Mehta, Hofmann, & Fankhauser, 2007; Mehta, 2007b). We apply the same procedure to PLSA, SVD, and k-NN for comparison.

In addition, we try a simple heuristic where we remove some of the user votes. Since extreme votes are the ones responsible for maximum deviation in case of attacks, we remove 10% of the highest and lowest votes of each person. We expect this heuristic to remove a large fraction of the votes on an attacked item from spam profiles, leading to a reduced prediction shift. Obviously, we expect the overall prediction accuracy to decrease for CF methods: however, it is possible that better methods can generalize well even from lesser data and not lose accuracy significantly.

4.5.1 Experimental Setup

To evaluate the performance of our proposed algorithms algorithm, we use the 1 million Movielens dataset which consists of 1,000,209 votes by 6040 users over 3952 movies and has been used previously for evaluating shilling detection. To this data, shilling profiles are added which all target the same item which is selected at random. shilling profiles are generated using the well studied models of Average, Random attacks, as well as Gaussian and uniform noise. Since average attacks tend to be the strongest, we present results only for them We use the generative models explained in (Mobasher et al., 2005) to generate these shilling profiles. A random 10% votes are removed from the dataset to create the test set; the training set then contains 900,209 votes to which spam votes are added. We add attack profiles with filler sizes of 3%, 5%, 7%, 10%, and 25%: the number of attack profiles ranges from 1% to 10%. Since adding a user profiles has a high human cost, we find addition of more profiles improbable in real-world systems.

4.5.2 Metrics Used

The task of evaluating predictions in collaborative filtering is easily described as the measurement of the *deviation* from observed values. Given that the user database can be compactly represented as a Matrix $\mathbf{X}$, with a user u_i forming a row with m items, the objective is to predict missing values in this matrix. Since only a small percentage of the matrix is observed, a portion of the observed data is artificially removed, and predicted using the remaining values. To measure the success of the prediction task, metrics which capture deviation from actual values are used. These include the *mean* and *root mean error*. An additional metric called the *ranking score* rates the ranking generated by the predicted user votes.

1. *Mean Average Error* $= \frac{1}{m}\sum_v |p_v - a_v|$, where p_v is the predicted vote, a_v is the actual vote, and MAE is measured over m votes.

2. *Root Mean Average Error* $= \sqrt{\frac{1}{m}\sum_v |p_v - a_v|^2}$, where p_v is the predicted vote, a_v is the actual vote, and MAE is measured over m votes.

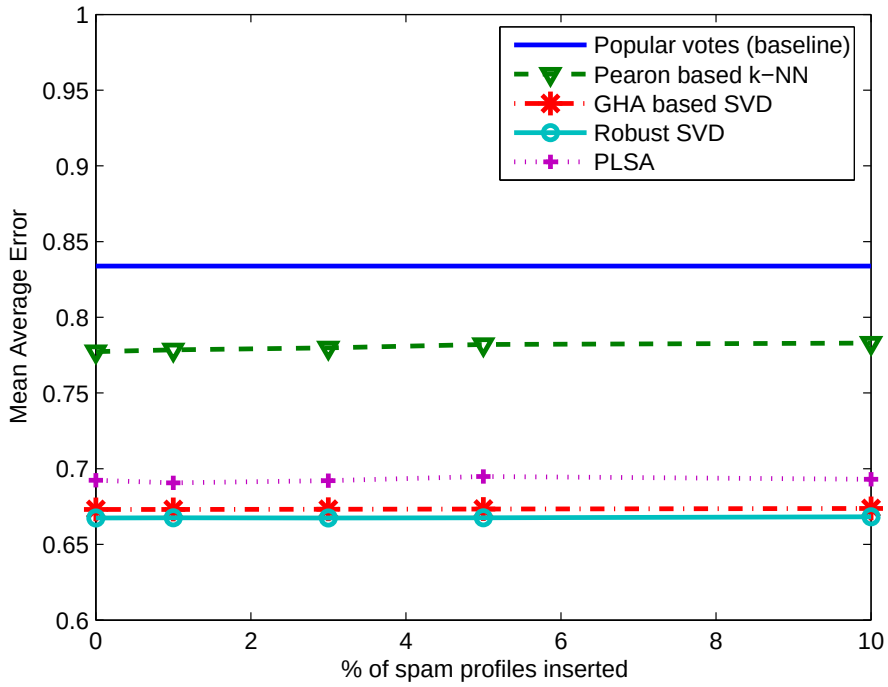

Figure 4.12: MAE of various CF algorithms compared to RMF measured over the testset: Attack profiles are inserted into the data and MAE is measured over the same testset. Interestingly, insertion of gaussian spam does not have a significant effect on the overall MAE

3. *MAE on attacked item* To measure the effect of the attack on prediction stability, we compute the mean average error of predicted votes of the attacked item in the test set. This is usually a small number of votes (say 40-100), and indicates the *real* shift in prediction. We prefer this over prediction shift, as it is difficult to compare multiple methods using prediction shift: a common baseline cannot be established, as the base performance of every method (before attack) is different. We measure the MAE after attack over multiple runs and present average results.

4.5.3 Experimental results

Our experiments show that the effect of targeted spam on the performance of various CF algorithms ranges from moderate to strong. The most robust algorithm turns out to be Simon Funk's SVD, followed by RMF and PLSA (see Fig. 4.13 & 4.14). The k-NN is easily influenced even when we set the neighborhood size to be 5% of the entire user population (300 neighbors). This is due to two reasons: Spam users generated using the average attack can penetrate user neighborhoods very effectively; secondly, the attacked items chosen by us are voted on by very few users ($< 5\%$), therefore the votes of the spam users become highly authoritative. SVD on the other hand does not get influenced so easily since the factors representing a user and an item are learnt from the overall pattern. Since a significant portion of the user community seems to have a below-average opinion of the attacked item, the effect of spam is lesser than for k-NN. (Mehta, 2007b) discusses the impact of spam on PLSA and concludes that the stability of PLSA against spam is due to the soft-clustering nature. This applies to SVD as well since it is similar

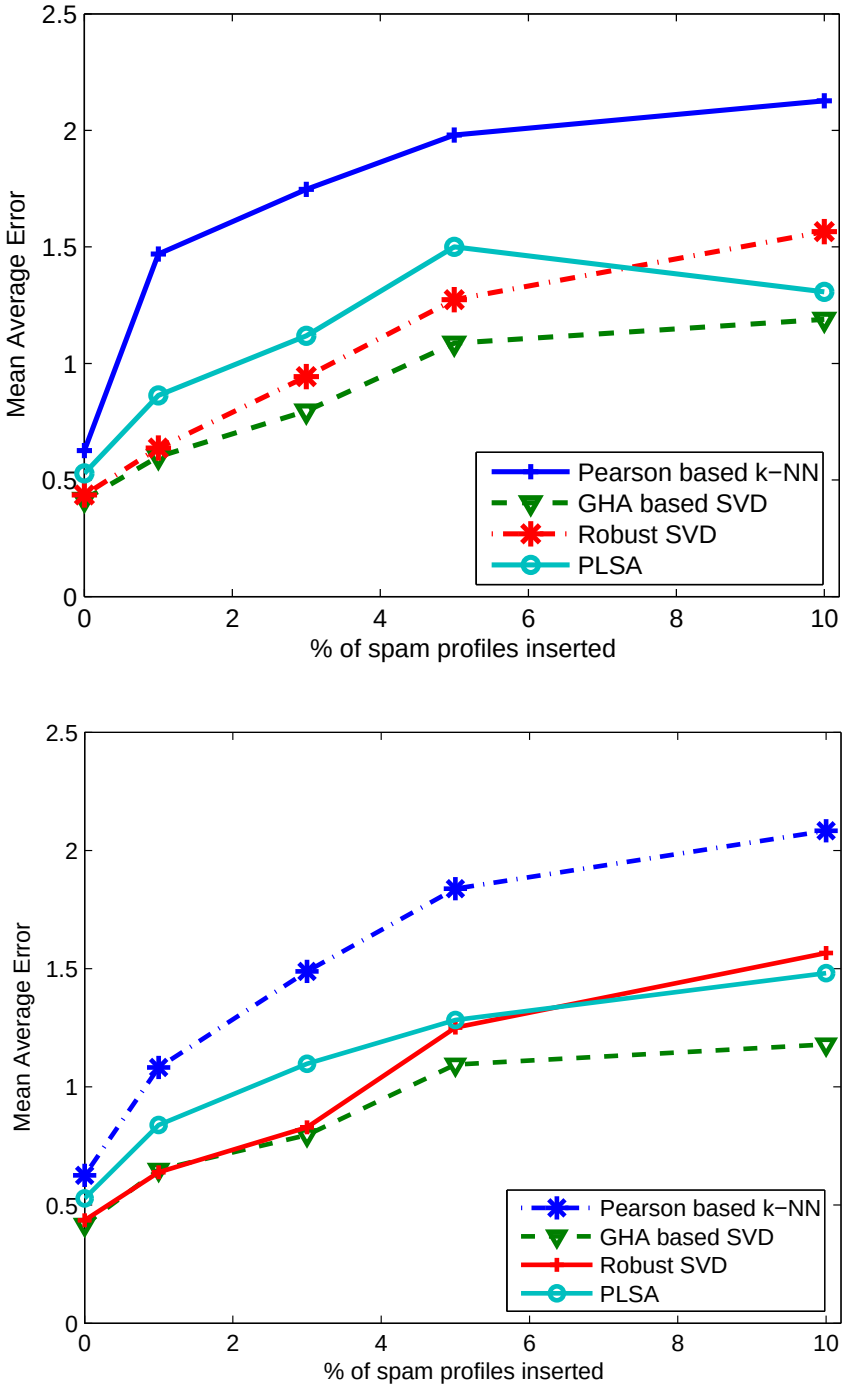

Figure 4.13: MAE of various CF algorithms on votes in the test set on the attack item a) with filler size=3%, b) with filler size=5%

	All data		80% data	
	MAE	Attacked item	MAE	Attacked item
k-NN (1%)	0.7965	*1.4179*	0.8065 (-1.2%)	*1.1014* (22.3%)
SVD (1%)	0.6731	*0.6669*	0.7018 (-4.2%)	*0.5471* (17.9%)
RMF (1%)	0.6677	*0.6721*	0.6982 (-4.5%)	*0.5836* (13.2%)
PLSA (1%)	0.6938	*1.1717*	0.7246 (-4.4%)	*0.6840* (41.6%)
k-NN (3%)	0.7992	*1.5268*	0.8074 (-1.0%)	*1.2178* (20.2%)
SVD (3%)	0.6733	*0.7625*	0.7013 (-4.2%)	*0.6726* (11.8%)
RMF (3%)	0.6681	*0.8806*	0.6987 (-4.6%)	*0.6523* (25.9%)
PLSA (3%)	0.6943	*1.1683*	0.7295 (-5.1%)	*0.8455* (27.6%)
k-NN (5%)	0.8088	*1.6149*	0.8067 (+0.2%)	*1.5198* (5.9%)
SVD (5%)	0.6737	*1.0882*	0.7004 (-3.9%)	*0.9338* (14.2%)
RMF (5%)	0.6684	*1.2514*	0.6980 (-4.4%)	*0.8759* (30.0%)
PLSA (5%)	0.6946	*1.4995*	0.7271 (-4.7%)	*1.1900* (20.6%)
k-NN (10%)	0.8076	*1.8031*	0.8039 (+0.4%)	*1.4930* (17.2%)
SVD (10%)	0.6736	*1.2659*	0.6998 (-3.9%)	*1.2811* (-1.2%)
RMF (10%)	0.6691	*1.5549*	0.6985 (-4.4%)	*1.2310* (20.8%)
PLSA (10%)	0.6969	*1.2589*	0.7292 (-4.7%)	*1.6346* (-29.8%)

Table 4.12: MAE of various CF algorithms on votes in the test set on the attack item, with filler size=7%. 20% of extreme votes has been removed for every user with more than 15 votes. *Attached item* represents the MAE on the observed votes on the attached item in the test set ($\sim 40-80$ votes)

in nature to PLSA. The use of various CF specific optimizations such as clipping leads to a better fitting model. At large filer sizes, k-NN appears to be more stable since the randomness in attack profiles lead to lower correlation; hence the effect of spammers is reduced. This trend has also been noted by previous research (Mobasher et al., 2006).

Our proposed Robust Matrix factorization algorithm also performs well in the face of moderate spam. Clearly, the effect of spam is low at small attack sizes, as the majority opinion is given more importance. However, once the number of votes by spammers are more than actual users, RMF starts treating the spammer's view as the majority opinion. The numbers also show that RMF is more tolerant to spam and model deviations than SVD and PLSA: the prediction accuracy of RMF is higher than any other method (see Fig. 4.12); this trend continues even in the face of spam attacks. Clearly, using robustness offers protection against minor departures from model assumptions.

Removing votes from data: An interesting trend appears when we remove 20% of the extreme votes from each user[7]: all collaborative filtering algorithms tested show increased stability w.r.t. prediction shift. Table 4.12 shows that the accuracy of all methods over the test set votes of the attached item is increased by more than 10%. This clearly comes with a loss in the overall accuracy; however SVD and RMF do not suffer significant losses. The MAE of the SVD, RMF and PLSA remains close to the value without any vote removal, while gaining significant accuracy on the attached item. Particulatly notable is the performance of RMF which gains more than 25% in MAE, outlining how effective it is in learning trends from less and noisy data.

[7]Only users with more than 15 votes in the training test are selected for vote removal.

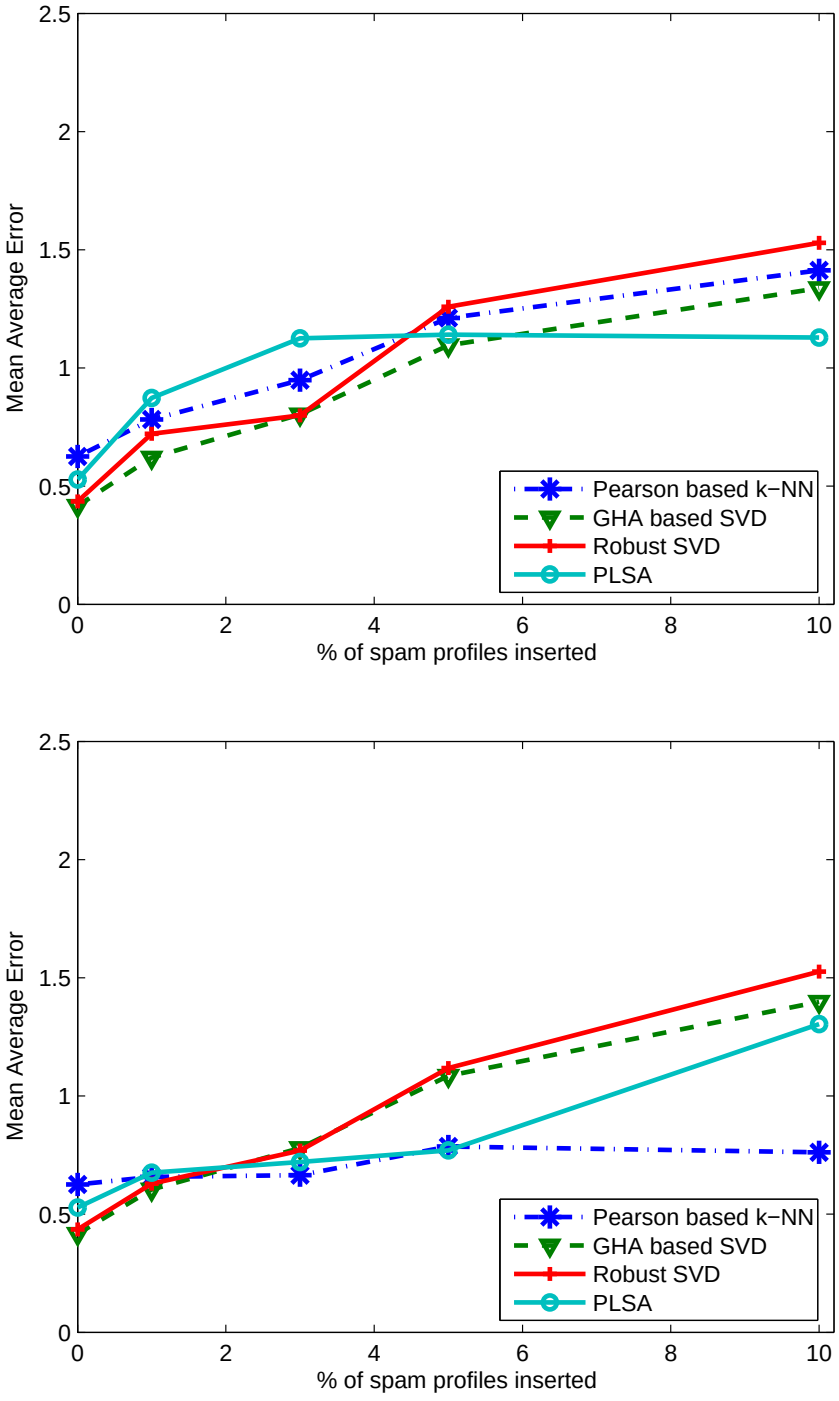

Figure 4.14: MAE of various CF algorithms on votes in the test set on the attack item a) with filler size=10%, b) with filler size=25%

4.5.4 Conclusions

This section investigates the effectiveness of robust statistics in protecting against collaborative filtering spam. We present a new algorithm for Robust Matrix Factorization similar in spirit to SVD which is more stable to noisy data. Experimental results show that application of M-estimators does not add significant stability; modified SVD algorithms outperform RMF in robustness. However, RMF adds significant stability as compared to other CF methods like PLSA and k-NN. The major positive outcome of this work is that RMF outperforms all other algorithms based on latent semantics (PLSA, SVD) in our dataset. However, the addition of robustness comes with a price: the RMF algorithm requires 4 times as much training time as SVD. This is a result of our training procedure which uses a fixed rate gradient descent approach; faster training can be achieved by using methods to accelerate gradient descent.

In addition, we have explored the effectiveness of vote sampling on stability and performance; removal of 20% of extreme votes leads to a significant increase in robustness for every method. While some methods suffer from a significant loss in accuracy due to lesser data, SVD and RMF can generalize well even from reduced data and provide accurate prediction. Future work involves developing faster training procedures for RMF and developing algorithms which provide higher robustness against spam.

5 Conclusions and Future Work

All truths are easy to understand once they are discovered; the point is to discover them.

(Galileo Galilei)

This thesis discusses challenging and relevant problems faced by researchers in the Recommender Systems community: the *cold start problem* and robustness when under shilling attacks. The idea of using a unified user profile which works with multiple systems is a unique one and has gained some popularity in the last few years, especially with the advent of *Web 2.0*. Users today face an overload of choice, in addition to overload of information, and therefore multi-task and use multiple websites for their jobs. Personalization is a successful mechanism to deal with the consequent information overload making it possible for users to directly access the information they need. Slowly however, the overhead of providing personal information in exchange for better service has increased tremendously. Future systems will have to evolve to support their customers making it easy to personalize their system while requiring them to provide lesser information. Approaches described and evaluated in this thesis are confident steps in achieving this perspective. Importantly, we describe not only the benefits of our approach, but also measure *how much* benefit is possible. Evaluation shows a large improvement over baseline approaches to the tune of 20%. Moreover we describe a privacy preserving protocol, where a group of users can collaborate together and learn a model over profile data while not disclosing personal information of any participating user to others. These steps form the crux of cross system personalization, on which a practical framework can be built.

Secondly, we also provide strong algorithms for detection of profile injection attacks (called *shilling* in literature). These algorithms are unsupervised in nature which means that extensive training can be avoided; moreover the performance is extremely good in a variety of situations. These algorithms, specifically the PCA based spam-detection algorithms, are the best performing spam detection algorithms in literature. We also provide a robust collaborative filtering algorithm which is more resistant towards spam than previous approaches. Also, the theoretical discussion of what constitutes an optimum shilling attack is the first of its kind in the collaborative filtering literature.

5.1 Future Work

There is always scope for improvement - goes the famous saying. In our case, there is a scope for improvement in the accuracy of the vector learning methods used for CSP. Moreover, scalability can be further improved by using online learning (Bottou, 1998) in model fitting. Such approaches can significantly accelerate the model learning phase, however fundamental

changes are required to the derived equations to proceed further. This work is currently under progress. Newer machine learning methods can also be applied to this problem after suitable improvements and modifications. An aspect which needs further work is designing a practical framework around this approach: engineering aspects like synchronizing with multiple systems and adding a reasoning back-end would be helpful in providing information in situations where not enough example data is available. Tying in some semantic features of systems and users would help in useful interaction with systems which provide some semantic knowledge about the profiling format used.

Regarding the second focus of this work – robustness – much can be done in developing robust recommendation algorithms. Such approaches should be stable under moderate attack conditions, and show insignificant impact under smaller attacks. The robust matrix factorization algorithm introduced in Chapter 3 is the first step in this direction. However the limited success of this method against strong attack strategies indicates that there is further scope for improvement.

6 References

Artale, A., Franconi, E., Guarino, N., & Pazzi, L. (1996). Part-whole relations in object-centered systems: An overview. *Data & Knowledge Engineering*, *20*(3), 347–383.

Azar, Y., Fiat, A., Karlin, A., McSherry, F., & Saia, J. (2001). Spectral analysis of data. In *STOC '01: Proceedings of the Thirty-third Annual ACM symposium on Theory of computing* (pp. 619–626). New York, NY, USA: ACM.

Bakir, G., Weston, J., & Schölkopf, B. (2004). Learning to Find Pre-Images. In L. S. Thrun S. & B. Schölkopf (Eds.), *Advances in Neural Information Processing Systems* (Vol. 16, pp. 449–456). Cambridge, MA, USA: MIT Press.

Barabási, A., Jeong, H., Néda, Z., Ravasz, E., Schubert, A., & Vicsek, T. (2002). Evolution of the social network of scientific collaborations. *Physica A: Statistical Mechanics and its Applications*, *311*(3-4), 590–614.

Baudisch, P. (2001). *Dynamic Information Filtering.* GMD-Forschungszentrum Informationstechnik.

Belkin, M., Matveeva, I., & Niyogi, P. (2004). Regularization and Semi-supervised Learning on Large Graphs. In *COLT 2004: The Seventeenth Annual Conference on Learning Theory* (pp. 624–638).

Belkin, M., & Niyogi, P. (2003). Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, *15*(6), 1373–1396.

Benerecetti, M., Bouquet, P., & Ghidini, C. (2000). Contextual reasoning distilled. *JETAI: Journal of Experimental and Theoretical Artificial Intelligence*, *12*(3), 279-305. Available from http://citeseer.ist.psu.edu/benerecetti00contextual.html

Bengio, Y., Paiement, J.-F., Vincent, P., Delalleau, O., Roux, N. L., & Ouimet, M. (2003). Out-of-Sample Extensions for LLE, Isomap, MDS, Eigenmaps, and Spectral Clustering. In *NIPS 2003: Advances in Neural Information Processing Systems.* Kaufmann.

Bottou, L. (1998). Online learning and stochastic approximations. *Online Learning in Neural Networks, D. Saad, Ed., Cambridge: Cambridge University Press, 1998*. Available from http://citeseer.ist.psu.edu/bottou98online.html

Breese, J. S., Heckerman, D., & Kadie, C. M. (1998). Empirical Analysis of Predictive Algorithms for Collaborative Filtering. In *UAI '98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence, July 24-26, 1998, University of Wisconsin Business School, Madison, Wisconsin, USA* (p. 43-52). Madison, Wisconsin, USA: Wiley.

Canny, J. (2002a). Collaborative Filtering with Privacy. In *IEEE Symposium on Security and Privacy* (pp. 45–57).

Canny, J. (2002b). Collaborative filtering with privacy via factor analysis. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 238–245). New York, NY, USA: ACM.

Chirita, P.-A., Nejdl, W., & Zamfir, C. (2005). Preventing Shilling Attacks in Online Recommender Systems. In *WIDM '05: Proceedings of the 7th annual ACM*

international workshop on Web information and data management (pp. 67–74). New York, NY, USA: ACM Press.

Colin Smythe, F. T., & Robson, R. (2001). *I.M.S. Learner Information Package Specification.* World Wide Web Publication. Available from http://www.imsproject.org/profiles/lipbpig01.html. (Final Specification)

Cranor, L., Dobbs, G., B. Hogben, Humphrey, J., Langheinrich, M. ., Massimo, M., Presler-Marshall, M., et al. (2004, Feb). *The Platform for Privacy Preferences 1.1 Specification.* http://www.w3.org/TR/P3P/.

Davies, N., Fensel, D., & Van Harmelen, F. (2003). *Towards the Semantic Web: Ontology-Driven Knowledge Management*. John Wiley and Sons.

Dawson, F., & Howes, T. (1998). *vCard MIME Directory Profile.*

Dempster, A., Laird, N., & Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*, *39*(1), 1-38.

Donoho, D., & Grimes, C. (2003). Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, *100*(10), 5591–5596.

Elgamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *Proceedings of CRYPTO 84 on Advances in cryptology*, 10–18.

Everitt, B. S. (1984). *An Introduction to Latent Variable Models*. New York: Chapman and Hall.

Fink, J., & Kobsa, A. (2002). User Modeling for Personalized City Tours. *Artificial Intelligence Review*, *18*(1), 33–74.

Frankowski, D., Cosley, D., Sen, S., Terveen, L., & Riedl, J. (2006). You are what you say: privacy risks of public mentions. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 565–572). New York, NY, USA: ACM.

Furnas, G. W., Deerwester, S., Dumais, S. T., Landauer, T. K., Harshman, R. A., Streeter, L. A., et al. (1988). Information retrieval using a singular value decomposition model of latent semantic structure. In *Sigir '88: Proceedings of the 11th annual international acm sigir conference on research and development in information retrieval* (pp. 465–480). New York, NY, USA: ACM.

Gabriel, K., & Zamir, S. (1979). Lower Rank Approximation of Matrices by Least Squares with Any Choice of Weights. *Technometrics*, *21*(4), 489–498.

Ghahramani, Z., & Hinton, G. (1997, 21). *The EM algorithm for mixtures of factor analyzers* (Tech. Rep. No. CRG-TR-96-1). Available from http://www.citeseer.ist.psu.edu/ghahramani97em.html

Ghahramani, Z., & Hinton, G. E. (1996, April). *The EM Algorithm for Mixtures of Factor Analyzers* (Tech. Rep. No. CRG-TR-96-1). Department of Computer Science. Available from http://citeseer.ist.psu.edu/ghahramani97em.html

Ghahramani, Z., & Jordan, M. I. (1994). *Learning from Incomplete Data* (Tech. Rep. No. AIM-1509). MIT. Available from http://citeseer.ist.psu.edu/ghahramani95learning.html

Gorrell, G. (2006). Generalized Hebbian Algorithm for Incremental Singular Value Decomposition in Natural Language Processing. In *EACL: 11th Conference of the European Association of Computational Linguistics* .

Ham, J., Lee, D., & Saul, L. (2003). Learning High Dimensional Correspondence from Low Dimensional Manifolds. In *ICML Workshop on the Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.*

Ham, J., Lee, D., & Saul, L. (2005). Semisupervised alignment of manifolds. In R. G. Cowell & Z. Ghahramani (Eds.), *AISTATS 2005: Tenth International Workshop on Artificial Intelligence and Statistics* (p. 120-127). Society for Artificial Intelligence and Statistics.

Hastie, T., Tibshirani, R., Eisen, A., Levy, R., Staudt, L., Chan, D., et al. (2000). Gene shaving as a method for identifying distinct sets of genes with similar expression patterns. *Genome Biol*, *1*(2), 1–21.

Hein, M., Audibert, J.-Y., & Luxburg, U. von. (2005). From Graphs to Manifolds - Weak and Strong Pointwise Consistency of Graph Laplacians. In *COLT 2005: The Eighteenth Annual Conference on Learning Theory* (p. 470-485).

Herlocker, J., Konstan, J. A., & Riedl, J. (2002). An Empirical Analysis of Design Choices in Neighborhood-Based Collaborative Filtering Algorithms. *Information Retrieval*, *5*(4), 287–310.

Hofmann, T. (2003). Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (pp. 259–266). New York, NY, USA: ACM.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, *22*(1), 89–115.

Huber, P. (1964). Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, *35*(1), 73–101.

Huber, P. (2004). *Robust Statistics*. Wiley-IEEE.

IEEE. (2000). *Draft Standard for Learning Technology. Public and Private Information (PAPI) for Learners.* (P1484.2/D7)

Joachims, T., Freitag, D., & Mitchell, T. (1997). WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*.

John W. Sammon, J. (1969, May). A non-linear mapping for data structure analysis. In *IEEE Transactions on Computing* (Vol. C18 (5), pp. 401–409).

Jolliffe, I. T. (2002). *Principal Component Analysis (2nd Edition)*. Springer.

Kaplan, C., Fenwick, J., & Chen, J. (1993). Adaptive Hypertext Navigation based on User Goals and Context. *User Modeling and User-Adapted Interaction*, *3*(3), 193–220.

Karypis, G. (2001). Evaluation of Item-Based Top-N Recommendation Algorithms. In *CIKM '01: Proceedings of the Tenth International Conference on Information and Knowledge Management* (pp. 247–254). New York, NY, USA: ACM Press.

Keerthi, S., & Chu, W. (2006). A matching pursuit approach to sparse Gaussian process regression. In Y. Weiss, B. Schölkopf, & J. Platt (Eds.), *Advances in Neural Information Processing Systems 18*. Cambridge, MA: MIT Press.

Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, *46*(5), 604-632.

Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M., et al. (2001). Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies. *W3C Working Draft*, *15*.

Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M. H., et al. (2003, March). *Composite Capabilities / Preferences Profile Working Group.* http://www.w3.org/Mobile/CCPP.

Kobsa, A. (2001). Generic User Modeling Systems. *User Modeling and User-Adapted Interaction Journal*, *11*, 49-63.

Kobsa, A., & Fink, J. (2003). Performance Evaluation of User Modeling Servers Under Real World Workload Conditions. *Proc. of the 9th International Conference on User Modeling, Johnstown, PA*.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). GroupLens: Applying Collaborative Filtering to Usenet News. *Communications of the ACM*, *40*(3), 77-87.

Lam, S. K., & Riedl, J. (2004). Shilling Recommender Systems for Fun and Profit. In *WWW '04: Proceedings of the 13th international conference on World Wide Web* (pp. 393–402). New York, NY, USA: ACM Press.

Li, X., Ning, Z., & Xiang, L. (2005). Robust 3D Reconstruction with Outliers Using RANSAC Based Singular Value Decomposition. *IEICE Transactions on Information and Systems*, *88*(8), 2001.

Liu, L., Hawkins, D., Ghosh, S., & Young, S. (2003). Robust singular value decomposition analysis of microarray data. *Proceedings of the National Academy of Sciences*, *100*(23), 13167–13172.

McDonald, D. (2003). Recommending collaboration with social networks: a comparative evaluation. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 593–600). New York, NY: ACM Press.

McGuinness, D., & Dvan Harmelen, F. (2004, February). *OWL Web Ontology LanguageOverview, W3C Recommendation 10 February 2004.* Available from http://www.w3.org/TR/2004/REC-owl-features-20040210/

Mehta, B. (2007a). Learning From What Others Know: Privacy Preserving Cross System Personalization. In *User Modelling 2007: Proceedings of the 11th International User Modelling Conference.* Corfu, Greece: Springer.

Mehta, B. (2007b). Unsupervised Shilling Detection for Collaborative Filtering. In *AAAI 2007: Proceedings of the 22nd Twenty-Second Conference on Artificial Intelligence.* Vancouver, Canada: AAAI Press.

Mehta, B., & Hofmann, T. (2006a, December). Cross System Personalization and Collaborative filtering by Learning Manifold Alignment. In *NIPS 2006 Workshop on Novel Applications of Dimensionality Reduction.*

Mehta, B., & Hofmann, T. (2006b). Cross System Personalization and Collaborative filtering by Manifold Alignment. In C. Freksa, M. Kohlhase, & K. Schill (Eds.), *KI 2006: Advances in Artificial Intelligence, Proceedings of the 29th German Conference on Artificial Intelligence.* Bremen, Germany: Springer.

Mehta, B., & Hofmann, T. (2006c). Cross System Personalization by Factor Analysis. In B. Mobasher & S. Singh (Eds.), *AAAI Press.* Boston, USA. (ISBN 978-1-57735-292-1)

Mehta, B., Hofmann, T., & Fankhauser, P. (2007). Lies and propaganda: detecting spam users in collaborative filtering. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces* (pp. 14–21). New York, NY, USA: ACM Press.

Mehta, B., Hofmann, T., & Nejdl, W. (2007). Robust Collaborative Filtering. *Proceedings of the 2007 ACM conference on Recommender systems*, 49–56.

Mehta, B., & Nejdl, W. (2007, Feb). An analysis of unsupervised spam detection strategies for Collaborative Filtering. *Document under submission to UMUAI Journal*.

Mehta, B., Niederee, C., & Stewart, A. (2005). Towards Cross-System Personalization. In *International Conference on Universal Access in Human-Computer Interaction.*

Mobasher, B., Burke, R., Williams, C., & Bhaumik, R. (2005). Analysis and Detection of Segment-Focused Attacks Against Collaborative Recommendation. In *Advances in*

Web Mining and Web Usage Analysis, 7th International Workshop on Knowledge Discovery on the Web, WebKDD 2005, Chicago, IL, USA, August 21, 2005. Revised Papers (p. 96-118).

Mobasher, B., Burke, R. D., & Sandvig, J. J. (2006). Model-Based Collaborative Filtering as a Defense against Profile Injection Attacks. In *AAAI 2006: Proceedings of the Twenty-First Conference on Artificial Intelligence.*

Mobasher, B., Cooley, R., & Srivastava, J. (2000). Automatic personalization based on Web usage mining. *Communications of the ACM*, *43*(8), 142–151.

Motta, E. (1999). *Reusable Components for Knowledge Modeling.* IOS Press, Amsterdam.

Nasraoui, O., Frigui, H., Joshi, A., & Krishnapuram, R. (1999). Mining Web access logs using relational competitive fuzzy clustering. *Proceedings of the Eight International Fuzzy Systems Association World Congress, August*.

Neuhold, E. J., Niederée, C., & Stewart, A. (2003). Personalization in Digital Libraries: An Extended View. In *Proceedings of ICADL 2003: 6th International Conference on Asian Digital Libraries* (pp. 1–16).

Newman, M. (2003). Elsevier Science BV (2003). Ego-centered networks and the ripple effect. *Social Networks*, *25*(1), 83–95.

Niederée, C. J., Stewart, A., Mehta, B., & Hemmje, M. (2004). A Multi-Dimensional, Unified User Model for Cross-system Personalization. In *Proceedings of Advanced Visual Interfaces International Working Conference (AVI 2004) – Workshop on Environments for Personalized Information Access, Gallipoli, Italy.*

O'Mahony, M., Hurley, N., Kushmerick, N., & Silvestre, G. (2004). Collaborative recommendation: A robustness analysis. *ACM Trans. Inter. Tech.*, *4*(4), 344—377.

O'Mahony, M. P., Hurley, N. J., & Silvestre. (2006, Jan). Detecting Noise in Recommender System Databases. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI'06), 29th–1st* (pp. 109–115). Sydney, Australia: ACM Press.

Pazzani, M. J. (1999). A Framework for Collaborative, Content-Based and Demographic Filtering. *Artificial Intelligence Review*, *13*(5-6), 393-408. Available from http://citeseer.ist.psu.edu/pazzani99framework.html

Pedersen, T. (1991). A threshold cryptosystem without a trusted party, Advances in Cryptology-EUROCRYPT'91. *Lecture Notes in Computer Science*, *547*, 522–526.

Pohl, W. (1997). LaboUr – Machine learning for user modeling. *Design of Computing Systems: Social and Ergonomic Considerations (Proceedings of the Seventh International Conference on Human-Computer Interaction). Elsevier, Amsterdam*, 27.

Pretschner, A., & Gauch, S. (1999, December). *Personalization on the Web* (Technical Report No. ITTC-FY2000-TR-13591-01). Information and Telecommunication Technology Center (ITTC), The University of Kansas, Lawrence, KS.

Riecken, D. (2000, August). Personalized Views of Personalization. *Communications of the ACM*, *43*(8), 27-28.

Rousseau, B., Browne, P., Malone, P., & ÓFoghlú, M. (2004). User Profiling for Content Personalisation in Information Retrieval. In *ACM Symposium on Applied Computing.*

Sanger, T. D. (1989). Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, *2*(6), 459-473.

Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. (2000). Application of dimensionality reduction in recommender systems–a case study. In *Application of dimensionality reduction in recommender systems–a case study.* Defense Technical Information Center. Available from http://citeseer.ist.psu.edu/sarwar00application.html

Sarwar, B. M., Karypis, G., Konstan, J. A., & Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. In *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001. ACM, ISBN 1-58113-348-0* (p. 285-295).

Saul, L. K., & Roweis, S. T. (2003). Think Globally, Fit Locally: Unsupervised Learning of Low Dimensional Manifold. *Journal of Machine Learning Research*, *4*, 119–155.

Schmidt, A., Beigl, M., & Gellersen, H. (1999). There is more to context than location. *Computers & Graphics*, *23*(6), 893–901.

Schölkopf, B., Smola, A. J., & Müller, K.-R. (1998). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, *10*(5), 1299-1319.

Schreck, J. (2003). *Security and Privacy in User Modeling*. Kluwer Academic Pub.

Schreiber, G., Akkermans, H., Anjewierden, A., De Hoog, R., Shadbolt, N., Velde, W. Van de, et al. (2000). *Knowledge Engineering and Management - The CommonKADS Methodology*. MIT Press, Cambridge, Massachusetts.

Shahabi, C., Zarkesh, A., Adibi, J., & Shah, V. (1997). Knowledge discovery from users Web-page navigation. *RIDE '97: Proceedings of the 7th International Workshop on Research Issues in Data Engineering*, 20-28.

Shapira, B., Shoval, P., & Hanani, U. (1997). Stereotypes in information filtering systems. *Information Processing and Management*, *33*(3), 273–287.

Shardanand, U., & Maes, P. (1995). Social information filtering: Algorithms for automating word of mouth. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 210–217). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Spiliopoulou, M., Pohle, C., & Faulstich, L. (1999). Improving the Effectiveness of a Web Site with Web Usage Mining. *Revised Papers from the International Workshop on Web Usage Analysis and User Profiling*, 142–162.

Srivastava, J., Cooley, R., Deshpande, M., & Tan, P. (2000). Web usage mining: discovery and applications of usage patterns from Web data. *ACM SIGKDD Explorations Newsletter*, *1*(2), 12–23.

Tenenbaum, J. B., Silva, V. de, & Langford, J. C. (2000). A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, *290*, 2319-2323. Available from http://web.mit.edu/cocosci/Papers/sci_reprint.pdf

Tipping, M., & Bishop, C. (1999). Probabilistic Principal Component Analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*(3), 611–622.

Traupman, J., & Wilensky, R. (2004). Collaborative Quality Filtering: Establishing Consensus or Recovering Ground Truth? In *WebKDD: KDD Workshop on Web Mining and Web Usage Analysis, in conjunction with the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2004).*

Tyler, S., & Treu, S. (1989). An interface architecture to provide adaptive task-specific context for the user. *International Journal of Man-Machine Studies*, *30*(3), 303–327.

Vassileva, J. (1994). A practical architecture for user modeling in a hypermedia-based information system. *4th International Conference on User Modeling*, 115–120.

Wasserman, S., & Galaskiewicz, J. (1994). *Advances in Social Network Analysis*. Sage Publications.

Webb, B. (2006). *Netflix Update: Try This at Home* [web page]. World Wide Web electronic publication. Available from http://sifter.org/~simon/journal/20061211.html

Weston, J., Chapelle, O., Elisseeff, A., Schölkopf, B., & Vapnik, V. (2002). Kernel Dependency

Estimation. In *NIPS 2002: Advances in Neural Information Processing Systems* (pp. 873–880).

Williams, C., Mobasher, B., Burke, R., Sandvig, J., & Bhaumik, R. (2006). Detection of Obfuscated Attacks in Collaborative Recommender Systems. In *ECAIŠ06 Workshop on Recommender Systems (ECAI'06)* .

Zhang, S., Wang, W., Ford, J., Makedon, F., & Pearlman, J. (2005). Using Singular Value Decomposition Approximation for Collaborative Filtering. In *Proceedings of CEC '05: the 7th International IEEE Conference on E-Commerce* (pp. 257–264). Washington, DC, USA: IEEE Computer Society.

Appendix A: List of Figures

Appendix B: List of Tables

Appendix C: List of Algorithms

Appendix D: List of Publications

Note: Only relevant publications have been listed.

2007

- Bhaskar Mehta, Thomas Hofmann, and Wolfgang Nejdl. Robust collaborative filtering. In *Recommender Systems:Proceedings of the 1st ACM Conference on Recommender Systems*, ACM Press New York, June 2007.
- Bhaskar Mehta and Wolfgang Nejdl. An analysis of unsupervised spam detection strategies for collaborative filtering, Document under submission to UMUAI Journal, Feb 2007.
- Bhaskar Mehta. Learning from what others know: Privacy preserving cross system personalization. In *User Modeling 2007: Proceedings of the 11th International User Modeling Conference*, Lecture Notes in Artificial Intelligence, Corfu, Greece, 2007. Springer.
- Bhaskar Mehta. Unsupervised shilling detection for collaborative filtering. In *AAAI 2007: Proceedings of the 22nd Twenty-Second Conference on Artificial Intelligence*, AAAI Press, Vancouver, Canada, 2007. AAAI Press.
- Bhaskar Mehta, Thomas Hofmann, and Peter Fankhauser. Lies and Propaganda: detecting spam users in collaborative filtering. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 14–21, New York, NY, USA, 2007. ACM Press.

2006

- Bhaskar Mehta and Thomas Hofmann. Cross system personalization and collaborative filtering by learning manifold alignment. In *NIPS 2006 Workshop on Novel Applications of Dimensionality Reduction*, December 2006.
- Bhaskar Mehta and Thomas Hofmann. Cross system personalization and collaborative filtering by manifold alignment. In Christian Freksa, Michael Kohlhase, and Kerstin Schill, editors, *KI 2006: Advances in Artificial Intelligence, Proceedings of the 29th German Conference on Artificial Intelligence*, Lecture Notes in Artificial Intelligence, 4314, Bremen, Germany, 2006. Springer.
- Bhaskar Mehta and Thomas Hofmann and Peter Fankhauser. Cross system personalization by factor analysis. In Bamshad Mobasher and Sarabjot Singh, editors, *AAAI Press*, AAAI Workshop Series Technical Report, WS-06-10, Boston, USA, 2006.

2005

- Bhaskar Mehta, Claudia Niederee, and Avare Stewart. Towards cross-system personalization. In *International Conference on Universal Access in Human-Computer Interaction*, 2005.
- Bhaskar Mehta, Claudia Niederée, Avare Stewart, Marco Degemmis, Pasquale Lops and Giovanni Semeraro. Ontologically-Enriched Unified User Modeling for Cross-System Personalization. In *International Conference on User Modeling*, 2005, pages 119–123.

2004

- C. J. Niederée, A. Stewart, B. Mehta, and M. Hemmje. A multi-dimensional, unified user model for cross-system personalization. In *Proceedings of Advanced Visual Interfaces International Working Conference (AVI 2004) - Workshop on Environments for Personalized Information Access, Gallipoli, Italy,*, 2004.