Association rule mining for recommender systems

Rick Smetsers
ANR: 349120

Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Arts in Communication and Information Sciences,
Master Track Human Aspects of Information Technology,
at the School of humanities
of Tilburg University

Thesis committee:

Dr. M.M. (Menno) van Zaanen
Dr. S. (Sander) Wubben

Tilburg University
School of Humanities
Department of Communication and Information Sciences
Tilburg center for Cognition and Communication (TiCC)
Tilburg, The Netherlands
August 2013

Abstract

Association rule mining is the task of identifying patterns in basket data – transactions that possibly consist of multiple items. The aim of this thesis is to better understand the applications of association rule mining for recommender systems, by researching how such systems perform compared to state-of-the art collaborative filtering approaches. First, we survey the tasks and purposes of recommender systems and we present an overview of the field of current generation collaborative filtering recommender systems. Then, two novel approaches are proposed for recommending items of low complexity. These items (such as books, movies and music tracks) rely on little or no metadata. Our first systems uses association rules as a complete replacement of collaborative filtering to predict item ratings, and our second system uses association rule mining to preprocess a latent factor model. The proposed systems are compared to a third system that uses a state-of-the art collaborative filtering approach. Our research methodology comprises an offline evaluation of these systems on the MovieLens dataset. We found that our systems did not improve upon the performance of a state-of-the art collaborative filtering approach.

Contents

Preface

> *Now finish the work, so that your eager willingness to do it may be matched by your completion of it, according to your means.*
>
> — 2 Corinthians 8:11

Despite my belief that this section is more important than any of those yet to follow, I will only concisely express my gratitude towards those that have helped me achieve this goal, as plenty of pages already make up for the rest of this thesis.

First and foremost, I would like to thank my parents and sister for supporting me and giving me the opportunity to fully focus on my study this past year. Second, I would like to thank my supervisors. Menno, thank you for your guidance during this thesis and the other projects that I have embarked on during my Masters'. Also, thank you Sander, for taking time out of your busy schedule to read this thesis. Lastly, I would like to thank Nathan and his colleagues from *Bibliotheek Midden Brabant* for giving me the opportunity to deploy our recommender system in a real-world application.

I would like to conclude this preface with a sincere "thank you" to some of the most important people in my life, who made sure I took my mind off my study every now and then.

Thank you Claudette, for being there when I most needed you. Thank you Jos, for the countless cups of coffee and tea that we have shared during our years here. Inez, thank you for making me realize that I am capable of so much more than I thought. Thank you Suzanne, for some the worst jokes I have heard in my life, and thank you Chris, for letting me win the occasional game of *OpenArena*. Lastly, thank you Peter, for keeping my beer mug filled in the weekends.

Chapter 1

Introduction

"Begin at the beginning," the King said gravely,
"and go till you come to the end; then stop."

— Lewis Carroll, *Alice in Wonderland*

As more and more information becomes available digitally, users will need help to determine what content to read, which products to buy, what music to listen to, or which movies to watch. The need for a digital personal assistant was recognized two decades ago (Goldberg, Nichols, Oki, & Terry, 1992; Maes, 1994) and ever since, **recommender systems** have gained a lot of attention from experts in fields such as artificial intelligence, information retrieval, machine learning, and marketing (Adomavicius & Tuzhilin, 2005).

This thesis proposes two novel approaches for recommender systems to identify interesting items. In this chapter, we explain the motivation for our research (Section 1.1), we briefly introduce the relevant concepts (Section 1.2) and we present our research questions and hypotheses (Section 1.3). We conclude this introduction with an outline of the remainder of this thesis and the methodology we use to evaluate our approaches (Section 1.4).

1.1 Use case: a public library

By automatically suggesting interesting content, recommender systems have changed the way how people find interesting information in a collection of books, movies, or music (Ricci, Rokach, Shapira, & Kantor, 2011). Utilizing past behavioral patterns of like-minded people, a recommender system identifies items that are (hopefully) interesting for a user. As such, a recommender system can be a valuable addition to an organization's services.

For the purpose of this thesis we approached *Bibliotheek Midden Brabant*, a public library that does not yet utilize this technology. While we were discussing the features of their application domain and their collection of books,

we discovered an important property of user data that is often overlooked in research on recommender systems: the notion that optimal recommendations may depend not only on the previous books read by a user, but also on the order in which those books were read. It would be much more sensible to recommend *The Two Towers after* one has read *The Fellowship of the Ring*, for example. While this example is trivial, we expect such sequential patterns to be widely evident in usage data.

Sequential decision problems are subject in studies on **language modeling**, a field in which some promising results for recommender systems have been found (Su et al., 2000; Zimdars, Chickering, & Meek, 2001; Shani, Heckerman, & Brafman, 2005). Despite these promising results, language models do not provide us with a proper solution for the problem at hand. The reason for this is that books are often borrowed in (small) batches by library card holders, which makes it impossible to effectively determine the order in which the books were read.

In this thesis, instead of focusing the sequence of items, we focus on these batches – or **baskets** – of items that are borrowed, bought or viewed together by users.[1] Let us introduce the approach that we examine, and describe how this approach relates to traditional recommendation approaches.

1.2 Beer and diapers

The aim of recommender systems is to suggest novel, unexpected items that the user might like – i.e. items for which the predicted rating is highest. Throughout the years different approaches have been proposed to maximize the value that recommendations have for a user. Our interest lies in the group of approaches that are collectively known as **collaborative filtering**, which recommend to a user the items that people with similar tastes have liked in the past. For example, if you ponder on watching *2001: A Space Oddyssey*, you might favor the opinion of a close friend with whom you have enjoyed watching multiple movies together over the opinion of your mother who seems to have an inexplicable fondness for movies starring *Hugh Grant*. Alternatively, you might favor the opinions of all people that liked a similar movie such as *A Hitchhiker's Guide to the Galaxy*, given that you enjoyed that movie as well.

By studying patterns of behavior, collaborative filtering automates this very principle of "word-of-mouth" to suggest novel, unexpected items that users might like. As such, collaborative filtering systems are centered around the use data mining to discover knowledge in the relations between items and users (Ricci et al., 2011). In this thesis we focus on a specific type of data mining knowledge known as association rules. One of the most repeated examples of knowledge discovery through association rules is the story that beer and dia-

[1] In the body of this thesis we will use a dataset that is more widely known in the field of recommender systems, rather than the data from *Bibliotheek Midden Brabant*. However, as the datasets share similar properties, the conclusions that are presented in Chapter 8 apply to our use case in particular.

pers frequently appear together in shopping baskets. "The explanation goes that when young fathers are sent out on an errand to buy diapers, they often purchase a six-pack of their favorite beer as a reward".[2] Regardless of whether this example is fact or fiction, it gives a good indication of the approach we take on in this thesis.

Association rule mining aims to find rules that will predict the occurrence of an item, based on other items in user's shopping baskets (as opposed to their complete transaction history in collaborative filtering). Despite its similarities with collaborative filtering, association rule mining has not become mainstream for recommender systems (Amatriain, Jaimes, Oliver, & Pujol, 2011). Nevertheless, some promising examples of recommender systems based on association rule mining exist (Mobasher, Cooley, & Srivastava, 2000; Davidson et al., 2010). These systems "indicate that association rules still have not had their last word" (Amatriain et al., 2011, p.65).

1.3 Research questions and hypotheses

The aim of this thesis is to better understand the applications of association rule mining for recommender systems, by researching how such systems perform compared to state-of-the art collaborative filtering approaches. Here, we focus on items of low complexity, such as books, movies and music tracks. Our problem definition is twofold. First, we develop a novel approach that can predict a user's utility for an item by using the association rules that are evident in previous interactions with the system. Accordingly, our first research question is:

RQ1 To what extent can association rule mining be used as an alternative to state-of-the-art collaborative filtering recommender systems?

Second, we propose a system that uses association rules in conjunction with current state-of-the-art collaborative filtering approaches. In this approach we use association rule mining to make the input data for a collaborative filtering system less sparse. We formulate the following research question:

RQ2 To what extent can association rule mining be used to improve state-of-the-art collaborative filtering recommender systems?

Based on the promising results by Mobasher et al. (2000) and Davidson et al. (2010) we expect our novel approaches to be suitable alternatives for the current state-of-the-art in collaborative filtering. Consequently, we hypothesize that

H1 Association rule mining can be used as a suitable alternative for current state-of-the-art collaborative filtering approaches,

and

H2 Association rule mining can be used to improve state-of-the-art collaborative filtering approaches.

[2] `http://www.dssresources.com/newsletters/66.php`

1.4 Outline

We start this thesis with an explanation of the relevant concepts of recommender systems in Chapter 2. By formulating the goals and functions of recommender systems, this chapter emphasizes the need for recommender systems and provides the reader with an intuition as to *why* such systems are important in the current digital landscape.

In Chapter 3 we describe in detail *how* existing recommendation approaches work. In this chapter, we limit our review to collaborative filtering, as a comprehensive description of alternative recommendation approaches is outside the scope of this thesis. First, we explain the basic principles that collaborative filtering and association rule mining approaches use to predict ratings in Section 3.1. Then, in Section 3.2 we provide a detailed description of the current state-of-the-art. Previous research on collaborative filtering is also thoroughly explored in this chapter.

In our experiments we evaluate two systems based on association rule mining, which we describe in Chapter 4. The former of these systems uses association rules as a replacement of collaborative filtering, while the latter uses association rule mining in conjunction with collaborative filtering to predict ratings. The performance of these systems is compared to that of state-of-the-art collaborative filtering system, which is described in Chapter 4.

Our research methodology comprises an offline evaluation of these systems. In Chapter 5 we describe the metrics that we use to evaluate the performance of our systems. Here, we also describe the experimental setup and dataset for our experiments. The experimental results are presented in Chapter 6, and our results are discussed in Chapter 7.

Chapter 8 concludes this thesis. In this chapter we formulate the answers that we found to our research questions. Based on these answers we discuss the extent to which association rule mining is applicable for recommending low-complex items. Given that this thesis proposes a range of novel recommendation approaches, we also address some future research directions in our conclusion.

Chapter 2

Recommender systems

People often say motivation doesn't last.
Neither does bathing –
that's why we recommend it daily.

— Hilary Hinton 'Zig' Ziglar

Recommender systems guide users in a personalized way to interesting, surprising, and useful items from a large number of possible options. The results are known as recommendations; options that are hopefully worthy of consideration (Burke, 2007). The motivation for recommender systems comes from the increasing volume of digitally available information. The need for filtering digital information was recognized more than two decades ago (Goldberg et al., 1992; Maes, 1994). These early recommender systems proposed a variety of methods to filter the increasing amounts of e-mail. Nowadays, recommender systems are deployed in a wide variety of applications, such as e-commerce, online music and movies, news, and digital libraries (Montaner, López, & De La Rosa, 2003). Today, recommendations can act as a viable alternative to traditional information retrieval methods, because they do not need keywords to search for content. Large scale commercial systems such as *YouTube*, *Amazon*, and *Netflix*, show that these recommendations are of great importance for the companies' services. According to Linden, Smith, and York (2003, p.76):

> "[*Amazon.com*] click-through and conversion rates – two important measures of Web-based and e-mail advertising effectiveness – vastly exceed those of untargeted content such as banner advertisements and top-seller lists."

In this chapter we discuss the need for recommender systems in the current digital landscape. We begin this chapter with a formal definition of recommendations (Section 2.1). Then, the various tasks for recommender systems (Section 2.2), and their functions (Section 2.3) are explained. We conclude this

chapter with an overview of the various evaluation methods for recommender systems (Section 2.4).

2.1 Recommendations defined

The purpose of a recommender system is to estimate ratings for items that have not yet been rated by a user (Adomavicius & Tuzhilin, 2005). While the recommendation task has been described formally more than a decade ago (Shardanand & Maes, 1995; Hill, Stead, Rosenstein, & Furnas, 1995), we follow the more recent formal representations proposed by Ricci et al. (2011). In their textbook the authors present an overview of the field of recommender systems and they describe the current generation of recommendation approaches. Their formal representations are appropriate for our thesis, because they are consistent across the different approaches that we present in Chapter 3 and Chapter 4.

Formally, the recommendation problem can be split into two subproblems: the **prediction problem** and the **presentation problem**. The prediction problem is about the estimation of an item's utility for a user. Let $\mathcal{U}$ be the set of all users and let $\mathcal{I}$ be the set of all possible items that can be recommended. Moreover, let $\mathcal{R}$ be the set of all possible ratings that the system can record, and let $\mathcal{S}$ be the range of possible values for a rating. A rating $r \in \mathcal{S}$ is a score by a user $u \in \mathcal{U}$ for an item $i \in \mathcal{I}$. It is defined as a tuple $\langle u, i, r \rangle$ that is denoted as $r_{u,i}$. We denote with $\mathcal{K}$ the subset of ratings $r_{u,i}$ in $\mathcal{R}$ that the system has recorded.

The prediction problem arises because ratings are usually not defined for the whole user-item space (i.e. $\mathcal{K} \subset \mathcal{R}$). Our goal is to come up with a utility function f that predicts unknown ratings. We denote such predicted ratings with $\hat{r}_{u,i}$. Formally, this means $\mathcal{K}$ needs to be extrapolated to the entire $\mathcal{U} \times \mathcal{I} \times \mathcal{S}$ space: $\mathcal{R}$ (see Equation 2.1).

$$\begin{aligned} f : \mathcal{U} \times \mathcal{I} \times \mathcal{S} &\rightarrow \mathcal{R} \\ \mathcal{K} &\rightarrow \mathcal{R} \end{aligned} \tag{2.1}$$

Once we have defined a utility function, the recommendation problem is reduced to a problem of presentation. Let us denote with $\mathcal{I}_u$ the set of items that are known to user u. Similarly, let us denote with $\mathcal{U}_i$ the set of users that know item i. For the active user u we aim to return the top-n previously unknown items $i \in \mathcal{I} \backslash \mathcal{I}_u$ for which the predicted rating $\hat{r}_{u,i}$ is highest (see Equation 2.2). A system that presents the top-n most relevant items to the user is known as a top-n recommender.

$$\arg\max_{i \in \mathcal{I} \backslash \mathcal{I}_u} \hat{r}_{u,i} \tag{2.2}$$

Recommender systems use various kinds of data to predict unknown ratings. Independent of the type of approach being used, data used by recommender

systems consists of three kinds of objects: **users**, **items**, and **interactions** – the latter being most commonly referred to as ratings (Ricci et al., 2011).

Users interact with recommender systems to discover new items. In order to personalize recommendations, recommender systems exploit a range of information sources about the users. Accurate modeling of their interests is required for providing good personalized recommendations (Berkovsky, Kuflik, & Ricci, 2008). The resulting models, known as user profiles, are subject in studies on user modeling (Fischer, 2001). In a sense, recommender systems can be viewed as tools that generate recommendations by building and exploiting user models. In personalized recommendations, the user model will always play a central role (Berkovsky et al., 2008). Depending on the recommendation approach, users might be modeled as a list of the ratings they provided for some items. Other approaches might use socio-demographic attributes such as age, gender, profession, and education to construct these user models (Ricci et al., 2011).

Items are the objects that are being recommended. In their extensive taxonomy, Montaner et al. (2003) provide an overview of item domains for which recommender systems have been developed. These include news articles (Sakagami & Kamba, 1997), web pages (Chen, Meng, Zhu, & Fowler, 2000), e-mail (Goldberg et al., 1992), movies (Hill et al., 1995), music (Shardanand & Maes, 1995), and e-commerce products (Cunningham, Bergmann, & Schmitt, 2001). The complexity of an item is of importance when selecting the most appropriate recommendation approach. Montaner et al. (2003) argue that recommendations for more complex items typically rely on more (meta)data. The authors consider the most complex items in their taxonomy to be financial investments, insurance policies, jobs, and travels. Consumer electronics, such as PCs, mobile phones, and digital cameras are considered to be of medium complexity. News items, web pages, movies, music tracks, and books are of low complexity. In this thesis we are concerned with items of low complexity.

Interactions between users and items are either inferred by interpreting user actions, or are explicitly expressed by users. These interactions are also known as **transactions** (Ricci et al., 2011). In most recommender systems, **ratings** are the most important type of interaction between items and users. Schafer, Frankowski, Herlocker, and Sen (2007) distinguish three types of ratings:

- **Ordinal** ratings express an extent to which users relate to an item. These ratings are often scalar, either expressed numerically (e.g. 1 – 5 stars), or through ordinal descriptions (e.g. "strongly agree, agree, neutral, disagree, strongly disagree").
- **Binary** ratings model choices in which the user is simply asked to decide if a certain item is conceived positively or negatively (e.g. "thumbs up or thumbs down").

- **Unary** ratings either indicate that a user has rated the item positively (e.g. Facebook "like"), or has interacted with an item, for example by observing, purchasing or borrowing it. As such, unary ratings are either explicitly expressed, or interpreted implicitly.

Besides ratings, interactions may also consist of other data that is useful for the prediction algorithm that the system is using. These interactions may be expressed explicitly by a user, such as tags (e.g. Bogers, 2009), or may be inferred implicitly by the system, such as viewing times (e.g. Parsons, Ralph, & Gallagher, 2004).

2.2 Recommender system tasks

Recommender systems help users reduce the information overload problem in their decision-making process. There are multiple reasons as to why users may need help making decisions. A recommender system's general task is to cater these user needs. In their acclaimed work on the evaluation of recommender systems, Herlocker and Konstan (2004) describe six generic tasks for recommender systems that help users to fulfill their goals. Some of these tasks are similar to those of information retrieval systems. For an overview of information retrieval tasks we refer to Manning, Raghavan, and Schütze (2008).

Annotation in context The original recommendation problem was defined as a filtering task. Tapestry (Goldberg et al., 1992) and GroupLens (Resnick, Iacovou, & Suchak, 1994) were two early recommender systems that filtered through discussion postings to decide which messages were worth reading for the user. Because in such case context of the messages is of importance, the task required retaining the order of messages and use a prediction model to annotate messages in context. Based on these predictions users decide which messages to read; they help users distinguish between relevant and non-relevant content.

Find good items After Tapestry and GroupLens, systems were developed that focused more on actual recommendations. Ringo (Shardanand & Maes, 1995) and the Bellcore Video Recommender (Hill et al., 1995) provided users with ranked lists of the recommended items along with the predicted rating for these items. This turned out to become the most common core recommendation task in most commercial recommender systems (Herlocker & Konstan, 2004; Ricci et al., 2011). Nowadays, in some cases the predicted ratings are not shown to the user. Instead, some systems display the items that the recommendations were based upon (i.e. "Because you like *Pulp Fiction*, you might like these movies").

Find all good items The problem that led to the need for recommender systems was one of information overload (Herlocker & Konstan, 2004). In most cases, recommender systems aim to find some good items and filter

out a lot of bad ones. However, in some cases users are interested in *all* relevant items. Lawyers, for example, often can not afford to miss a single relevant legal document. If a recommender system is designed for such a purpose, it should suggest all the items that satisfy the user needs, even if this means that some items that are not interesting are presented to the user.

Recommend sequence Instead of focusing on the generation of a single recommendation, the task is either to recommend the next item given a sequence of items that have been rated by a user, or to recommend a sequence of items that is pleasing as a whole (Ricci et al., 2011; Herlocker & Konstan, 2004). Typical examples include recommending book or movie series, or an ensemble of musical tracks. Little research has been conducted on this type of task (Hayes & Cunningham, 2001; Shani et al., 2005; Su et al., 2000).

Recommend bundle Recommend a group of items that fit well together. A set of books in a library, for example. Ricci, Cavada, Mirzadeh, and Venturini (2006) present such a system in the context of travel recommendations. A travel plan may be composed of a destination, a flight, an accommodation, and some attractions that are located close to the accommodation. From a user's point of view such a bundle can be selected as a single travel plan.

Just browsing In talking with users of Amazon.com and of several other websites, Herlocker and Konstan (2004) discovered that many of them use the sites even when they have no intentions of purchasing an item. Instead, they simply find it pleasant to browse. In such case, the task of the recommender system is to help the user to find the items that are likely to be interesting to a user for that specific browsing session (Ricci et al., 2011). Here, not only the accuracy of algorithms is of importance, as the interface, the ease of use, and the level and nature of information provided also contribute to the quality of a browsing session (Herlocker & Konstan, 2004).

Find credible recommender Users do not automatically trust a recommender system. Many of them "play around" for a while to see if the recommender matches their tastes well. Herlocker and Konstan (2004) report on users who are looking up their favorite movies on their system to check up on the quality of the recommendations.

> "Some users even go further. Especially on commercial sites, they try changing their profiles to see how the recommended items change. They explore the recommendations to try to find any hints of bias. A recommender optimized to produce 'useful' recommendations (e.g., recommendations for items that the user does not already know about) may fail to appear trustworthy

> because it does not recommend movies the user is sure to enjoy but probably already knows about" (Herlocker & Konstan, 2004, p.10).

2.3 Recommender system functions

Not only a *user's* motivation for using a recommender system can vary. There are various reasons as to why the **service provider** uses the technology as well. It is important to be aware of these functions while designing, deploying and evaluating the system. Here, we mention a selection of the various recommender system functions that are described by Ricci et al. (2011):

Increase conversion rate From a commercial service provider's point of view an increase in the number of items sold is often the most important function for a recommender system. For such systems the conversion rate is the number of items sold compared to the number of items viewed over time. For non-commercial service providers the conversion rate is expressed differently. For example, for a news recommender it might be expressed as a relative increase in article reads.

Sell more diverse items Prediction models try to accurately predict the items that users like. However, quite often the recommendations tend to be biased towards popular, well-known items.

> "... in 2007 only 1% of all digital tracks accounted for 80% of all sales. Similarly, 1,000 albums accounted for 50% of all album sales, and 80% of all albums sold were purchased less than 100 times" (Celma, 2008, p.vii).

In some cases, this bias towards popular items decreases the effectiveness of the recommendations. In order to sell more diverse items, prediction models recommend unpopular items from the **Long Tail** to the right users. The Long Tail describes a distribution that is composed of a small number of popular items, and a large amount of unpopular ones (Anderson, 2008). Such a distribution is not unique for the number of albums sold. The words we use in our every day language, names, scientific citations and earthquake magnitudes are also distributed in this way. In general, when the probability of measuring a particular value of some quantity varies inversely as a power of that value, the quantity is said to follow a **power law** (Newman, 2005). Depending on the domain, this phenomenon is known as the Long Tail, Zipf's law or the Pareto distribution.

Increase user satisfaction The combination of effective, accurate recommendations and a usable interface will increase the users' evaluation of the system. This in turn will increase system usage and the likelihood that the recommendations will be accepted (Ricci et al., 2011). In essence, this

function is strongly related to *Sell more diverse items*, as accurate recommendations of items in the Long Tail are generally positively perceived. In his book, Anderson (2008) argues that the future of business is in the Long Tail of items: selling less of more. He propagates what Celma and Cano (2008) describe as the *Hits vs. Niche* paradigm:

> "Until now the world was ruled by the *Hit or Miss* classification, due in part to the shelf space limitation of the brick-and-mortar stores. A world where a music band could only succeed selling millions of albums, and touring worldwide. Nowadays, we are moving towards the *Hit vs. Niche* idea, where there is a large enough availability of choice to satisfy even the most 'Progressive-obscure-Swedish-metal' fan. The problem, though, is to filter and present the right artists to the user, according to her musical taste" (Celma & Cano, 2008, p.5:1).

In Section 2.2 we have mentioned some important motivations as to why users may want to use a recommender system. In this section we have described a selection of recommender system functions from a service provider's perspective. It is important to note here that a recommender system must balance its user tasks and their goals with its function for the service provider in order to offer a service that is valuable to both.

2.4 Evaluating recommender systems

Recommender systems have a variety of properties that affect the user experience, such as prediction accuracy, scalability, and robustness (Shani & Gunawardana, 2011). In previous sections we have argued why it is important to balance recommender systems' tasks (from a user's perspective) with its functions (from a service provider's perspective). Depending on these needs we decide on which properties we need to focus when we evaluate the system. Following the terminology by Herlocker and Konstan (2004) and Shani and Gunawardana (2011), we describe three types of evaluation methods: offline evaluation, user studies and online evaluation. We conclude this section, with a description of what can and cannot be evaluated with each of these methods.

Evaluation methods for recommender systems are motivated by those for related research areas. We refer to publications in the fields of machine learning (e.g. Witten, Frank, & Hall, 2011) and information retrieval (e.g. Voorhees, 2002; Manning et al., 2008) for a more detailed discussion of the topic.

Offline evaluation methods measure the extent to which the system can predict ratings that a user has previously assigned. As such, this evaluation metric is also known as **system-centric evaluation** (Celma, 2008). In offline evaluation, computational experiments are conducted in order to determine how a prediction model would perform in practice. The most

common way to do this is, is by means of a repeated "all but n" experiment. Here, for each user n item ratings are "hidden" in the dataset. The prediction model is trained on visible partition of the data, the **train set**. The "hidden" partition, or **test set**, is used to evaluate the model. This process is then repeated a couple of times to account for biases in the train set.[1]

Offline evaluation methods can be subdivided in three categories. **Prediction-based** metrics compare the predicted ratings to the actual values. **Decision-based** metrics evaluate the fitness of the top-n recommendations for a user. Finally, **rank-based** metrics compare the ordered list of a user's known ratings to the list of predicted ratings.

User studies are conducted by recruiting users, and asking them to perform some tasks that involve interaction with the recommendation system (Shani & Gunawardana, 2011). In contrast to offline evaluation methods that are system-centric, user studies focus on user judgements of the proposed recommendations. The category refers to a wide range of methodologies that often focus on evaluating more subjective aspects of recommendations, such as serendipity (e.g. Bogers, Rasmussen, Sebastian, & Jensen, 2013), or other user-perceived qualities of the recommendations. Often the evaluations consist of user surveys in which qualitative questions are posed.

Online evaluation methods measure the change in user behavior that a recommender system imposes. Typically, online evaluation consists of A/B testing. This is a methodology in which two variants of a system, A and B, are presented to a random, equal distribution of users that simultaneously make use of the system (i.e. in the same time span). For example, in case of an e-commerce site, A could be a version of the website that presents the user top sellers, while B could include personalized recommendations. The goal of A/B testing is to identify changes to a system that maximize an outcome of interest, such as profit, the number of users, or the diversity of the items sold.

From the three evaluation methods that we have surveyed here, user studies provide the most insights in the extent to which a recommender system fulfills its tasks for the users (which we have described in Section 2.2). User studies allow the service provider to gain insight in properties of recommendations that typically cannot be measured with the other two evaluation methods that we survey here. On the downside, user studies are expensive to conduct compared to offline evaluation methods. As the recruited test subjects should represent the actual users of the system as closely as possible, the organization of user

[1] An "all but n" experiment is preferred over the more common n-fold cross validation experiment (where the data is divided in n partitions and the model is trained on $n-1$ partitions), because it is better to evaluate a recommender system on a per-user basis (Shani & Gunawardana, 2011). In a n-fold cross validation experiment the size of the test set differs for each user, which makes it harder to average the results.

studies require much more attention compared to offline and online evaluation methods (Shani & Gunawardana, 2011).

From a service provider's perspective, online evaluation is often most useful, as it provides the strongest evidence for the true value of a recommender system or a specific approach (Section 2.3). Through A/B testing, an increase in profit, the number of user, or the diversity of items sold can directly be identified as the result of a specific recommendation approach.

However, as both user studies and online evaluation are subject to resources that are beyond the scope of this research, the evaluation in our research is limited to offline evaluation methods. Offline evaluation has its advantages as well. They are attractive because they allow us to compare a range of candidate approaches at a low cost, as they require no interaction with real users. According to Shani and Gunawardana (2011), "the downside of offline experiments is that they can answer a very narrow set of questions, typically questions about the prediction power of an algorithm". As such, we cannot directly measure a system's influence on user behavior in this setting. Offline methods are often used to determine whether or not a certain recommendation approach is a valid candidate for further evaluation. In Chapter 5, we describe offline evaluation metrics that we use in our experiments.

Chapter 3

Collaborative filtering

Different roads sometimes lead to the same castle.

— George R.R. Martin *A Game of Thrones*

In Chapter 2 we explained the recommendation problem and the reasons *why* recommender systems are of such great importance in the current digital landscape. In this chapter we describe *how* recommender systems solve the recommendation problem.

The problem of recommending items has been studied extensively, and two main paradigms have emerged. **Content-based** recommendation approaches recommend items similar to those the active user has liked in the past, whereas **collaborative filtering** recommendation approaches identify users whose tastes are similar to those of the active user and recommend items they have liked (Balabanović & Shoham, 1997). In this chapter we focus on the latter group of approaches, as the focus of this thesis is on recommendations based on usage data.

There are different ways to predict ratings based on collaborative filtering. **Neighbourhood-based** approaches estimate the interest of a user for an item by using the ratings for this item by other users that have similar rating patterns, called **neighbours** (Desrosiers & Karypis, 2011). We discuss these neighbourhood-based collaborative filtering approaches in Section 3.1.

Model-based systems use the collection of ratings to learn a probabilistic model, which is then used to make rating predictions (Adomavicius & Tuzhilin, 2005). The field of model-based recommender methods is very diverse, as any system that trains a prediction model belongs to this group. In Section 3.2, we discuss an approach known as latent factor modeling. This approach is considered as the state-of-the-art today (Koren, Bell, & Volinsky, 2009).

The limitations of collaborative filtering are often addressed in studies on more specific recommendation approaches that have emerged in the past few years. Among these are:

hybrid approaches that combine collaborative filtering and content-based methods,

knowledge-based approaches that exploit deep knowledge about the underlying item domain,

context-aware approaches that take into account additional contextual information, such as time and location.

It it beyond the scope of this thesis to discuss these approaches, as they rely on different types of input data. For more information on these approaches, we refer to Adomavicius and Tuzhilin (2005), Felfemig, Friedrich, Jannach, and Zanker (2011), and Adomavicius and Tuzhilin (2011) respectively.

3.1 Neighbourhood-based approaches

Collaborative filtering approaches predict user preferences for items by learning from user-item interactions. In neighbourhood-based approaches (also known as **memory-based** or **nearest-neighbour** approaches) the system guesses which items are interesting by comparing the user's rating pattern to other that of other users.

> "Recommender systems based on nearest-neighbours automate the common principle of word-of-mouth, where one relies on the opinion of like-minded people or other trusted sources to evaluate the value of an item (movie, book, articles, album, etc.) according to his own preferences. Due to their simplicity and efficiency, memory-based methods enjoy a huge amount of popularity among collaborative filtering recommender systems" (Desrosiers & Karypis, 2011, p.114).

Consider the user-item matrix in Table 3.1. Based on Desrosiers and Karypis (2011), Example 3.1 intuitively explains how collaborative filtering works.

Example 3.1. Peter is trying to decide whether or not to watch *Django Unchained*. He knows Jeff shares very similar tastes when it comes to movies: both of them liked *Inglorious Basterds* and *300*, and disliked *V for Vendetta* and *Iron Man*. Lea, on the other hand, seemingly has different tastes, since she enjoyed the latter two movies. While Kim and Peter have a similar opinion on *Iron Man*, Kim has not seen any of the other movies. Based on their rating similarity, Peter favors Jeff opinion over Kim's and Lea's. He decides to watch *Django Unchained*.

In **User-based** collaborative filtering methods, such as the one described in Example 3.1, each user $u \in \mathcal{U}$ is represented as a vector in an n dimensional rating space, where n is the number of items i in the system. We estimate the rating $\hat{r}_{u,i}$ of a user for an unknown item by averaging the ratings that the users most similar to u gave to that item. These like-minded users are called

Table 3.1: Example user-item matrix depicting ratings for five items by four users

	Django Unchained	Inglorious Basterds	300	V for Vendetta	Iron Man
Lea	1	2	1	5	4
Jeff	5	4	5	2	1
Kim	1				2
Peter	?	5	4	1	2

the **nearest-neighbours**. Suppose for each user $v \neq u$ we have a value $w_{u,v}$ representing the similarity between u and v.[1] The k nearest neighbors of u, denoted by $\mathcal{N}(u)$, are the k users v with the highest similarity to user u. When we estimate a user's rating for item i, we only consider its nearest-neighbours who have rated i. We denote the set of users most similar to u who have rated i as $\mathcal{N}_i(u)$.

The value of the unknown rating $r_{u,i}$ is an aggregate of the ratings in $\mathcal{N}_i(u)$ (Equation 3.1):

$$\hat{r}_{u,i} = \underset{v \in \mathcal{N}_i(u)}{\text{aggr}} \; r_{v,i} \tag{3.1}$$

Various aggregation functions have been proposed in literature.[2] We consider the aggregation function of Desrosiers and Karypis (2011), because it is both recent, simple and widely accepted. The estimated rating $\hat{r}_{u,i}$ is a function of the average ratings given to i by the neighbors in $\mathcal{N}_i(u)$, where the contribution of each neighbor v is weighed by their similarity to user u ($w_{u,v}$) (Equation 3.2). Here $|w_{u,v}|$ is the number of nearest neighbors.

$$\hat{r}_{u,i} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{u,v} r_{v,i}}{\sum_{v \in \mathcal{N}_i(u)} |w_{u,v}|} \tag{3.2}$$

Instead of relying on the opinion of like-minded users to predict a rating, **item-to-item** (or **item-based**) collaborative filtering matches each of the user's purchased and rated items to similar items (Linden et al., 2003). We illustrate how item-to-item collaborative filtering works by paraphrasing an example by Desrosiers and Karypis (2011). In Example 3.2 we revisit our user-item matrix from Table 3.1.

Example 3.2. Instead of consulting with Lea, Jeff and Kim, Peter determines whether *Django Unchained* is right for him by considering the movies that he

[1] User similarities can be calculated in different ways. As the methods for computing the similarity between two vectors are not specific for user-based collaborative filtering approaches, we discuss these methods later in this section.

[2] For an overview, see Adomavicius and Tuzhilin (2005).

has already seen. He notices that people that have rated this movie have given similar ratings (both positive and negative) to *Inglorious Basterds* and *300*. Given that Peter liked these two movies he concludes that he will also like *Django Unchained*.

Formally, we denote by $\mathcal{N}_u(i)$ the set of items most similar to item i that have been rated by user u. The rating $\hat{r}_{u,i}$ is an aggregate of the ratings of the set of items j most similar to i that have been rated by user u, $\mathcal{N}_u(i)$ (Equation 3.3):

$$\hat{r}_{u,i} = \underset{j \in \mathcal{N}_u(i)}{\text{aggr}} \; r_{u,j} \tag{3.3}$$

The reader might note that Equation 3.3 is very similar to Equation 3.1. Formally, item-to-item and user-based collaborative filtering approaches only differ in their orientation of the item-user matrix. As such, any user-based rating estimation function can be applied in an item-based approach as well. Consider Equation 3.2. If we wish to use this rating estimation in an item-based approach, we take the weighted average of the ratings given by u to the items of $\mathcal{N}_u(i)$ (Equation 3.4):

$$\hat{r}_{u,i} = \frac{\sum_{j \in \mathcal{N}_u(i)} w_{i,j} r_{u,j}}{\sum_{j \in \mathcal{N}_u(i)} |w_{i,j}|} \tag{3.4}$$

Despite their formal similarities, the choice between an item-to-item or a user-based approach can have significant impact on the quality of the recommendations. When choosing an approach, five criteria should be considered:

Accuracy The accuracy of neighbourhood-based recommendation approaches depends mostly on the ratio between the number of users and the number of items in the system. Often, a small number of high-confidence nearest-neighbors is preferred to a large amount of neighbors with few overlapping interactions. Therefore, in case of large commercial systems with millions of users and only thousands of items, such as *Amazon* or *YouTube*, an item-to-item approach yields more accurate recommendations (Linden et al., 2003; Davidson et al., 2010).

Efficiency Computational costs also depend on the ratio between the number of users and items. When the number of users exceeds the number of items, item-to-item approaches require less time to compute the similarity weights (w), making them more scalable (Linden et al., 2003).

Stability Item-based methods are preferred if the list of available items is fairly static in comparison to the users of the system. Similarity weights can then be computed at infrequent time intervals. In applications where the list of available items is constantly changing, user-based methods could prove to be more stable (Desrosiers & Karypis, 2011).

Transparency Item-to-item collaborative filtering can easily justify a recommendation by listing the items that caused an item to occur in the list of

recommendations (Davidson et al., 2010). User-based recommendations can be justified less easily in this sense, because the user typically does not know the other users of the system.

Serendipity Because they work with user similarity, user-based approaches are more likely to make serendipitous recommendations. A user's nearest neighbors, despite being similar to the active user, might have rated a broad variety of interesting items that are unknown to the user. Item-based recommendations tend to recommend items that are more strongly related to those appreciated by the user at hand.

Regardless of the user-based or item-to-item approach, two decisions need to be made when we apply memory-based collaborative filtering. We need to decide on how the similarity weights are computed, and whether or not we normalize the rating. Below, we describe the considerations for these decisions. The descriptions and the equations are partly derived from Desrosiers and Karypis (2011):

Computation of the similarity weights Various metrics have been used to compute the similarity weight w for two users $u \in \mathcal{U}$ and $v \in \mathcal{V}$. A common method is to measure the cosine of the angle between their rating vectors $\vec{u}$ and $\vec{v}$ (Equation 3.5):

$$w_{u,v} = \text{similarity}(u, v) = \cos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} \tag{3.5}$$

The cosine similarity can be calculated for two items, $i \in \mathcal{I}$ and $j \in \mathcal{I}$ as well. Each vector then corresponds to an item rather than a user, and the vector dimensions correspond to the users who have rated that item. The values correspond to the ratings by these users for the item.

Mobasher et al. (2000) and Davidson et al. (2010) propose another metric to calculate similarity weights for item-based approaches. They make use of a concept known as **co-visitation** to compute the similarity weights between items. For each pair of items (i, j) they count how often they were co-visited by users. A pair of items is co-visited if a user has rated both items within a certain **transaction** (usually defined as a one-day time period). Let us denote this co-visitation count with $c_{i,j}$. Then the similarity of item i to item j is defined according to Equation 3.6.

$$w_{i,j} = \text{similarity}(i, j) = \frac{c_{i,j}}{c_i c_j} \tag{3.6}$$

The denominator $c_i c_j$ acts as a normalization function for the co-visitation counts, as it takes the popularity for i and j in account. Here, c_i and c_j are the total occurrence counts for items i and j respectively. The approach we describe in following chapters, known as **association rule mining**, is closely related to the one proposed by Mobasher et al. (2000) and Davidson

et al. (2010). Instead of using the co-visitation counts, we use **association rules** to calculate the similarity weights. In Chapter 4 we will describe this method in more detail.

Normalization of the ratings Each user has its own personal scale when it comes to assigning ratings to items. We need to decide if we want to take these personal rating scales in account. The most straightforward way of doing this is by **mean-centering**, or normalizing a user's ratings. In user-based approaches, a rating $r_{u,i}$ for an item i by a user u is normalized by by subtracting the average rating $\bar{r}_u$ of the user. (Equation 3.7):

$$h(r_{u,i}) = r_{u,i} - \bar{r}_u \tag{3.7}$$

When predicting a rating $\hat{r}_{u,i}$ using this approach, Equation 3.2 is not sufficient. A user-based prediction of a rating is obtained by taking in account both the active user's average rating $\bar{r}_u$ and neighbors' average ratings $v \in \mathcal{N}_i(u) : \bar{r}_v$ (Equation 3.8):

$$\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv}(r_{vi} - \bar{r}_v)}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|} \tag{3.8}$$

Mean-center normalization of $r_{u,i}$ can also be applied in an item-based collaborative filtering approach. Here, a rating is transformed to a mean-centered one by subtracting to $r_{u,i}$ the average $\bar{r}_i$ of the ratings given to item i by users in $\mathcal{U}_i$. When predicting ratings, we now need to take in account the average rating $\bar{r}_i$ for item i and the average ratings for neighboring items $j \in \mathcal{N}_u(i) : \bar{r}_j$. Equation 3.4 thus becomes Equation 3.9.

$$\hat{r}_{u,i} = \bar{r}_i + \frac{\sum_{j \in \mathcal{N}_u(i)} w_{ij}(r_{uj} - \bar{r}_j)}{\sum_{j \in \mathcal{N}_u(i)} |w_{uv}|} \tag{3.9}$$

Memory-based collaborative filtering methods have been applied in various domains, such as e-commerce (Linden et al., 2003; Cunningham et al., 2001), movie and video recommendations (Hill et al., 1995; Good et al., 1999; Davidson et al., 2010), music recommendations (Celma, 2008; Shardanand & Maes, 1995; Hayes & Cunningham, 2001), web recommendations (Balabanović & Shoham, 1997), and netnews filtering (Resnick et al., 1994). The most notable systems are listed in Table 3.2. Besides operating in different domains, these systems use different collaborative filtering approaches (item-based or user-based), and different methods for selecting neighbors. Table 3.3 lists the various combinations of approaches and methods that have been studied.

3.2 Latent factor models

Due to their simplicity and efficiency, neighbourhood-based approaches enjoy a huge amount of popularity among collaborative filtering recommender systems.

Table 3.2: An overview of pioneering collaborative filtering recommender systems and their domains.

Name	Reference	Domain
Amazon.com	Linden et al. (2003)	e-commerce
Bellcore	Hill et al. (1995)	movie recommendations
Fab	Balabanović and Shoham (1997)	web recommendations
FOAF	Celma (2008)	music recommendations
GroupLens	Resnick et al. (1994)	netnews filtering
MovieLens	Good et al. (1999)	movie recommendations
Ringo/FireFly	Shardanand and Maes (1995)	music recommendations
WebSell	Cunningham et al. (2001)	e-commerce
YouTube	Davidson et al. (2010)	video recommendations

Table 3.3: Filtering approaches and neighborhood selection methods for memory-based collaborative filtering systems.

Name	Filtering approach	Neighborhood selection
Amazon.com	item-to-item	cosine similarity
Bellcore	user-based	Pearson r correlation
Fab	user-based	cosine similarity
FOAF	hybrid	RDF graph similarity
GroupLens	user-based	Pearson r correlation
MovieLens	user-based	cosine similarity
Ringo/FireFly	hybrid	mean squared difference, Pearson r correlation
WebSell	user-based	case-based reasoning with Pearson r correlation
YouTube	item-to-item	association rule mining

However, a significant amount of research has been done on **model-based** collaborative filtering in the past ten years. In Section 3.1 we have described how neighbourhood-based methods use ratings to calculate similarities between users or items. These similarities are then used to predict unknown ratings. Because this generalization is delayed until a request is made to the system, neighbourhood-based approaches are classified as *lazy* learning methods. In contrast to lazy learning methods, *eager* model-based approaches use the set of user-item ratings to learn a prediction model in an initial training stage. This model is then used to predict ratings (Adomavicius & Tuzhilin, 2005).

In the past years various probabilistic models have been proposed. In this section, instead of surveying the entire domain, we focus on a specific group of prediction model. **Latent factor models** approach collaborative filtering with the goal to uncover latent features inferred from the known rating patterns. High-dimensional input data is represented in a latent factor space of lower dimensionality. In a sense, these dimensions, or **factors** are a computerized alternative to the human notion of genres.

> "For movies, the discovered factors might measure obvious dimensions such as comedy versus drama, amount of action, or orientation to children; less well-defined dimensions such as depth of character development or quirkiness; or completely uninterpretable dimensions. For users, each factor measures how much the user likes movies that score high on the corresponding movie factor" (Koren et al., 2009, p.43).

Examples of latent factor models include **Latent Dirichlet Allocation** (LDA), in which users are represented as random mixtures over latent topics, where each topic is characterized by a distribution over items (Blei, Ng, & Jordan, 2003); **Probablistic Latent Semantic Analysis** (pLSA), which introduces latent features in a mixture model to discover user clusters and prototypical user profiles (Hofmann, 2004); and **Restricted Bolzmann Machines** (RBMs), a type of artificial neural network for modeling tabular data.

A technique that is of particular interest to us is **matrix factorization** (also known as **singular value decomposition**, or SVD). Matrix factorization is a well known dimensionality reduction technique to uncover latent semantic factors in high dimensional data (Deerwester, Dumais, Furnas, & Landauer, 1990). Typically usage data for recommender systems is of a very high dimensional nature. In user-based nearest-neighbour approaches each item is considered a dimension, and in item-based approaches each user is considered a dimension.

Factorization is a technique that groups strongly correlated dimensions together to form more descriptive dimensions. Once the prediction model has factorized the input data, both users and items are represented in a joint latent factor space of lower dimensionality. This reduces the problem that data sparsity imposes, because ratings are now represented in a space of lower dimensionality. This allows us to find interesting combinations more easily. Items and users are expressed as a vector of factors inferred from their original rating vectors. These factors may represent the extent to which an item or user is "geared

towards males or females" or "serious or escapist", for example. Because items and users are represented in a single latent space, ratings can be estimated by simply computing the cosine similarity of any user - item combination.

Another benefit of latent factor models compared to memory-based collaborative filtering models is that it is more flexible in dealing with various aspects of input data (Koren & Bell, 2011). Rating normalization can not only be applied to account for personal user scales, but also for other biases that can not be measured in neighbourhood-based approaches, such as temporal dynamics. Moreover, ratings can be predicted as simple dot products, as both items and users are represented in the same latent semantical space.

Formally, we denote the joint latent factor space of dimensionality f with $\mathcal{M}^f$. Each item $i \in \mathcal{I}$ is associated with a vector $q_i \in \mathcal{M}^f$, in which each element measures the extent to which the item possesses a factor. Each user $u \in \mathcal{U}$ is associated with a vector $p_u \in \mathcal{M}^f$, in which the elements measure the extent of interest the user has in items that are high on the corresponding factors (Koren et al., 2009). An item is recommended to a user when high correspondence between their factor vectors is found. As such, once the prediction model has learned the mappings users and items to factors, it can estimate a rating $\hat{r}_{u,i}$ by taking the dot product of these factor vectors (Equation 3.10). This captures the latent similarities between user u and item i – the user's interest in the item's characteristics (Koren et al., 2009).

$$\hat{r}_{u,i} = q_i p_u \tag{3.10}$$

The main challenge for recommender systems based on latent factor models is to learn the mappings of the original user-item space to the latent factor space. It is this task that matrix factorization attempts to solve. To get an intuition on how this is achieved one needs to understand how regression analysis and dimensionality reduction by means of principal component analysis (PCA) work. In Example 3.3 we illustrate this in a simple example.

Example 3.3. In this example we describe the process of reducing dimensionality with principal component analysis. Consider the user-item rating matrix in Equation 3.11):

$$\mathcal{K} = \mathcal{R} = \begin{pmatrix} r_{1,1} & r_{1,2} \\ r_{2,1} & r_{2,2} \\ \vdots & \vdots \\ r_{n,1} & r_{n,2} \end{pmatrix} \tag{3.11}$$

Here, ratings by users u for items i are in the form $r_{u,i}$. Our example system thus contains two items and n users. Furthermore, ratings are known for all user-item combinations, thus $\mathcal{K} = \mathcal{R}$.

The attentive reader should note that there is no use for a recommender system in our example. After all, in a user-item space that is fully dense, there are no ratings left to predict. Nonetheless, this example allows us to clearly relate the concepts of users, items and ratings to matrix factorization. Like in

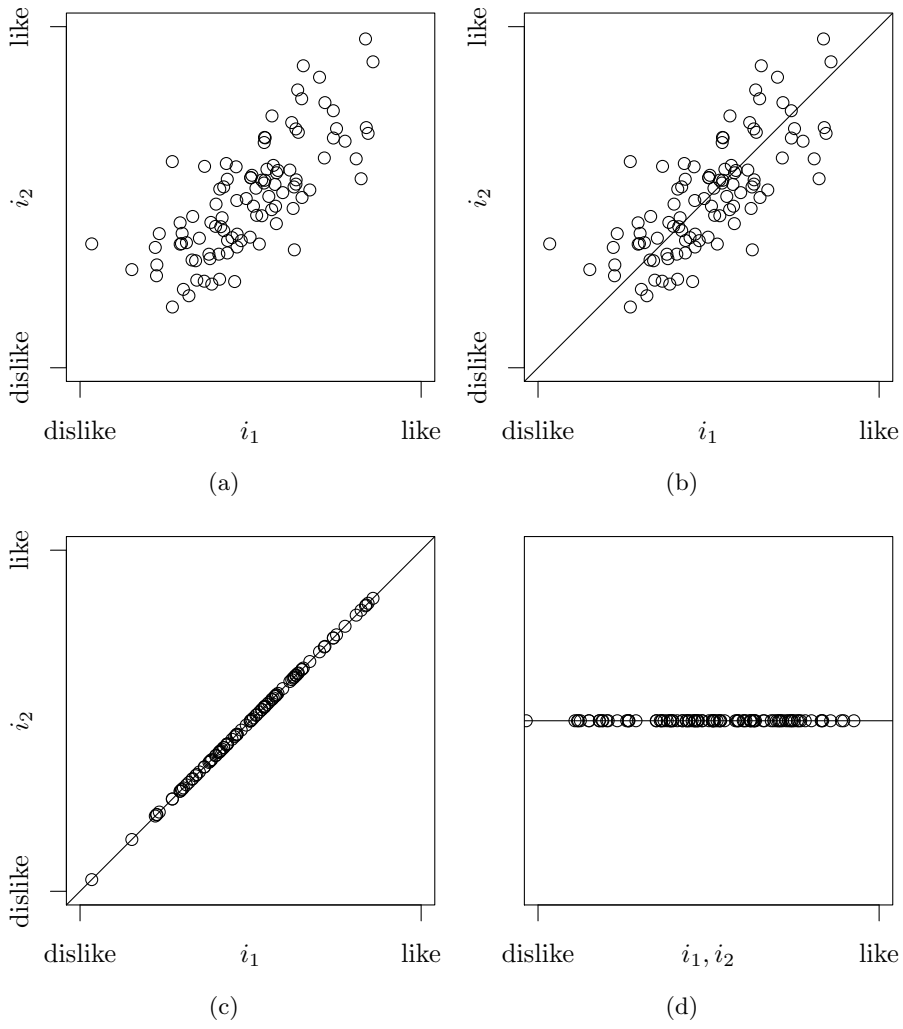


Figure 3.1: Reducing dimensionality with principal component analysis

a regular, sparse user-item space, users u are represented by their ratings r for items i_1 and i_2. Figure 3.1a is a graphical representation of this user-item space. Each user's ratings are plotted in our two-dimensional item-space. Intuitively, we see a strong correlation between ratings for i_1 and i_2. We would like to exploit this correlation by reducing the dimensionality of each user vector $[r_{u,1}, r_{u,2}] = \vec{r}_u$ to a single value p_u. This is done by finding a model q for our data, that defines the direction on which we map our single values p_u. This process is known as **principal component analysis** (PCA), and the ideal model that describes the maximum variance is known as the **first principal component** of our data. We calculate the first principal component by minimizing the mean squared error between our data $\vec{r}_u$ and the reconstructed data points p_u (Equation 3.12). Figure 3.1b depicts the first principal component of our data.

$$\min_{p,q} \sum_u (\vec{r}_u - p_u q)^2 \tag{3.12}$$

If we fix q, the best p_u is simply the projection of $\vec{r}_u$ on our model q. What is important to note here is that, given the strong correlation between i_1 and i_2, qp_u approximates the rating for i_1 and i_2, and thus describes the extent to which user u likes *both* items (Equation 3.13).

$$qp_u \approx \vec{r}_u \tag{3.13}$$

In Figure 3.1c, the p_u values are mapped to the principal component q. Let us denote these predicted ratings with $\hat{r}_{u,i}$. Given that $\hat{r}_{u,1} = \hat{r}_{u,2}$ in our example, having two dimensions to describe our approximations is redundant. In Figure 3.1d we see these values projected on a single dimension. Here, we have factored our original data by reducing its dimensionality.

Let us apply this example in practice. Say i_1 and i_2 refer to *Pulp Fiction* and *Kill Bill* respectively. Then in a sense, a value on the principal component (Figure 3.1d) describes one's attitude towards prototypical Quentin Tarantino films. As such, we have uncovered a "more semantical" latent feature in our data. It is important to note that we can not only project user attitudes on this feature, but item associations as well. We denote these item correspondences with q_i. In our toy example, *Pulp Fiction* and *Reservoir Dogs* have both have strong positive correspondence to our *Tarantino* factor, but negative relations to factors are also possible. By applying matrix factorization, we have thus uncovered a "latent semantical space" in which we can describe both users and items by means of p_u and q.

We can apply the techniques we described Example 3.3 to a recommender system's usage data in its full complexity. Here, we would uncover a f-dimensional joint latent factor space $\mathcal{M}^f$ for users and items, by calculate the top f principal components.

One can think of matrix factorization as an alternative way to discover latent semantical features in a data set. Amatriain et al. (2011) describe that matrix factorization is based on the the idea that it is always possible to decompose

the rating space $\mathcal{R}$ (i.e. $\mathcal{U} \times \mathcal{I} \times \mathcal{S}$) into $\mathcal{R} = U\Sigma V$. Given the original $u \times i$ matrix, we can obtain a $u \times \sigma$ matrix U, a $\sigma \times \sigma$ diagonal matrix Σ, and a $i \times \sigma$ matrix V. In these matrices σ are the **singular values** that refer to the latent semantical features (the singular values in matrix factorization can be thought of as the principal components in PCA). As such U is interpreted as the "user-feature" matrix and V is interpreted as the "item-feature" matrix. Σ contains the feature strengths and sorts them in decreasing order (so the first singular value σ_1 can be thought of as the first principal component). If we would reduce our original data to a f-dimensional latent feature space, we denote with $U_f\Sigma_f V_f$ the matrices obtained by selecting f highest singular values.

In our formal definition of the recommendation problem (Section 2.1), we described that our goal is to extrapolate $\mathcal{K}$ to the entire user-item space $\mathcal{R}$. We also explained that recommending is not merely a classification task, but a regression task: for each user we wish to return the item(s) for which the predicted utility is *highest*. A property of a prediction model based on matrix factorization is that it can express the predicted rating $\hat{r}_{u,i}$ as an ordinal value, even if our data model consists of merely unary classifications ($\mathcal{S} = 1$).

We argue that a more detailed explanation of matrix factorization at this point would merely interrupt and clutter the narrative. As our best interest is to focus on the contributions this research makes, we therefore refer to Koren and Bell (2011, pp.151–154), Koren et al. (2009), and Desrosiers and Karypis (2011, pp.132–134) for a more detailed description of matrix factorization for recommender systems. For a more in-depth definition of matrix factorization and its roots in information retrieval we refer to Berry, Dumais, and O'Brien (1995).

In 2009, Koren et al. (2009) won the *Netflix Prize* competition with a prediction model based on matrix factorization. The Netflix Prize was an open competition for collaborative filtering algorithms to predict user ratings for movies. Koren, Bell and Volinsky's team *BellKor's Pragmatic Chaos* won the 1 million dollar competition, besting Netflix's own algorithm for predicting ratings by 10.06%.[3] By winning the competition, they showed that the prediction quality of latent factor models is superior to that of neighbourhood based approaches.

[3]For more information on the Netflix Prize competition, see `http://www.netflixprize.com`.

Chapter 4

Association rule mining

There are only patterns.
What we call chaos is just patterns we haven't recognized.
What we call random is just patterns we can't decipher.

— Chuck Palahniuk, *Survivor*

Our goal is to build a recommender system that can improve its prediction model by mining association rules in usage data. In Section 3.1 we have introduced the subject of association rule mining when we discussed co-visitation counts as a method to compute similarity weights between items. In this chapter we describe two recommender systems that use association rule mining as the base for their rating predictions. We begin this chapter with a detailed description of association rule mining (Section 4.1). In this section we describe our item-based collaborative filtering approach that is based on association rule mining. In Section 4.2 we revisit matrix factorization and we describe a problem that data sparsity might impose to a latent factor model. We aim to solve this problem by applying association rule mining as a preprocessing step for matrix factorization.

4.1 Mining for rating predictions

Association rules capture the relationships between items based on their patterns of co-occurrence across transactions (Mobasher et al., 2000). Here, a transaction is a tuple of items that have been bought by a user over a given amount of time. An example of a transaction is the list of films on *Netflix* by a user in a given month. Association rule mining is the task of identifying patterns in so called **basket data**. Basket data differs from traditional collaborative filtering usage data in that the transactions possibly consist of multiple items, instead of a single one. An example of an association rule based on basket data is that 90% of the people who watch *Star Wars* and *The Empire Strikes Back!*

in a given week, also watch *A New Hope* later that week. Given the transactions in our basket data our goal is to:

1. Find all the significant association rules.
2. Build a prediction model based on these association rules.
3. Recommend to users the top-n items that are associated with the items that they have already rated.

There are different properties of association rules that we can use to predict ratings. What follows is a formal definition of the relevant concepts, derived from Agrawal, Imielinski, Swami, and Jose (1993, p.208) and Agrawal, Srikant, and Jose (1994, p.487).

Let $\mathcal{I} = \{i_1, i_2, i_3, \ldots, i_m\}$ be the set of items, and $\mathcal{D} = \{T_1, T_2, T_3, \ldots, T_n\}$ be the set of transactions, where each transaction T is a set of items such that $T \subseteq \mathcal{I}$. The set of transactions is pruned by removing transactions that contain only one item, because these can not be used to form association rules. We say that a transaction T contains an **itemset** X if $X \subseteq T$. An itemset is defined as a collection of one or more items. Moreover, a k-itemset is an itemset that contains k items. The frequency of a given itemset is known as its **support count** and the proportion of transactions in $\mathcal{D}$ that contain the itemset is known as its **support** (Equation 4.1).

$$s(X) = \frac{\text{count}(X)}{|\mathcal{D}|} \tag{4.1}$$

An **association rule** $X \Rightarrow Y$ is a relation between two itemsets X and Y. These itemsets are non-overlapping, i.e. $X \cap Y = \emptyset$. In an association rule, X is called the **antecedent** itemset and Y is called the **consequent**. Let us define the metrics that describe the relations between the antecedent and the consequent itemset: support, confidence and lift.

The support s for an association rule is defined as the probability that we observe X and Y in any given transaction, i.e. the proportion of transactions in $\mathcal{D}$ that contain $X \cup Y$.[1] We can calculate the support for an association rule in the same way as we calculate the support for an itemset (Equation 4.2):

$$s(X \Rightarrow Y) = s(Y \Rightarrow X) = s(X \cup Y) \tag{4.2}$$

The **confidence** c of a rule is defined as the probability that we observe Y given that we observe X, i.e. the proportion of times that the association rule is correct (Equation 4.3).

$$c(X \Rightarrow Y) = \frac{s(X \cup Y)}{s(X)} \tag{4.3}$$

[1] The union of the itemsets $(X \cup Y)$ should be interpreted as "transactions where X and Y *both* appear" and not as "transactions where either X or Y appears". This ambiguity arises because set union is similar to logical disjunction (which is more commonly denoted with $\vee$).

The **lift** of an association rule measures the mutual dependence of X and Y. It is the same as the **interest factor** or **pointwise mutual information** of X and Y, which is described by Bouma (2009, p.33) as:

> "... a measure of how much the actual probability of a particular co-occurrence of events differs from what we would expect it to be on the basis of the probabilities of the individual events and the assumption of independence."

The lift $l(X \Rightarrow Y)$ can be seen as the ratio of the observed support for $X \cup Y$ to that of the expected support if X and Y were unrelated, see Equation 4.4.

$$l(X \Rightarrow Y) = l(Y \Rightarrow X) = \frac{s(X \cup Y)}{s(X) \cdot s(Y)} \tag{4.4}$$

If our observed support for the association rule is equal to the support that we would expect for two independent itemsets, i.e. $s(X \cup Y) = s(X) \cdot s(Y)$, then X and Y are statistically independent. Based on this we can interpret the measure as described in Equation 4.5 (derived from Tan, Steinbach, & Kumar, 2006).

$$l(X \Rightarrow Y) \begin{cases} = 1, & \text{if } X \text{ and } Y \text{ are unrelated.} \\ > 1, & \text{if } X \text{ and } Y \text{ are positively correlated.} \\ < 1, & \text{if } X \text{ and } Y \text{ are negatively correlated.} \end{cases} \tag{4.5}$$

In our system we use the confidence, support and lift of the association rules in transaction data to predict recommendations.[2] In our implementation the consequent is always a 1-itemset and the antecedent is a k-itemset where k is an variable larger or equal to 1. This means that we aim to find the association rules between a single item, or an itemset of multiple items $X \in \mathcal{I}$ and a single consequent $i_k \in \mathcal{I}$.

Let us explain how we predict ratings from the association rules. First, let us focus on how we find the rules that are relevant for our prediction model. The brute-force approach would be to list all possible association rules, compute the confidence, support and lift for each of these rules and then prune the rules that do not meet all conditions (Amatriain et al., 2011). However, this is computationally expensive as all the possible item combinations have to be considered. Instead, we decompose the problem of association rule mining in two sub-problems:

Frequent itemset identification Our goal is to find all k-itemsets that satisfy a minimum support threshold – which we call *minsup* – as efficiently as possible. A **frequent itemset** F_k is a k-itemset with a support that is greater or equal to *minsup*. Because we know the number of total transactions beforehand ($|\mathcal{D}|$), we convert *minsup* to a minimum support

[2] We use support, confidence and lift as a measure of relatedness and not as a measure of causality.

frequency *mincount* that makes it easier to process the itemsets (Equation 4.6).

$$mincount = minsup \cdot |\mathcal{D}| \tag{4.6}$$

We use the APRIORI algorithm (Agrawal et al., 1994) to identify all F_k itemsets in our data as efficiently as possible. First, we simply count item occurrences to determine the frequent 1-itemsets in our data. The algorithm to determine the frequent k-itemsets (starting with k=2) consists of two steps. First, we construct all the candidate k-itemsets C_k by joining F_{k-1} and F_{k-1}. Next, we delete all itemsets $c \in C_k$ for which some $(k-1)$-subset is not in F_{k-1}. What we are left with are the candidate k-itemsets for which we calculate the support in order to determine if they are frequent or not. The itemsets with a support that is greater or equal to *minsup* are stored. This process is repeated until there are no candidates left to consider. Let us illustrate the frequent itemset generation step with a simple example (Example 4.1).[3]

Example 4.1. Let F_3 be $\{\{i_1\ i_2\ i_3\},\ \{i_1\ i_2\ i_4\},\ \{i_1\ i_3\ i_5\},\ \{i_2\ i_3\ i_4\}\}$. After the join step, C_4 is $\{\{i_1\ i_2\ i_3\ i_4\},\ \{i_1\ i_3\ i_4\ i_5\}\}$. Next, we delete $\{i_1\ i_3\ i_4\ i_5\}$ because $\{i_1\ i_4\ i_5\}$ is not in F_3. We are then left with the itemset $\{i_1\ i_2\ i_3\ i_4\}$ in C_4, for which we check if belongs to F_4 (i.e. whether it is frequent or not).

The process rules out all k-itemsets that can never be a frequent, based on the *a priori* knowledge that for a frequent itemset, all its subsets are also frequent and thus for an infrequent itemset, all its supersets must also be infrequent. This property of support is known as **downward closure** (Tan et al., 2006).

Recommendation generation The top-n recommendations are generated by finding the n strongest association rules of a user's frequent itemsets $F_k(u)$ and previously unknown items. For each frequent item i in the set of items that have not been rated by user u ($\mathcal{I}/\mathcal{I}_u$), we construct association rules for $f \Rightarrow i$, where $f \in F(u)$ are the frequent itemsets of u for which holds that $f \cup i$ is frequent (i.e. has minimum support). For each item in $i \in \mathcal{I}/\mathcal{I}_u$ the rating is predicted according to Equation 4.7 and Equation 4.8.

$$x = \log\left(\frac{\sum_k \sum_{f \in F_k(u)} l(f \Rightarrow i)}{\sum_k |F_k(u)|}\right) \tag{4.7}$$

$$\hat{r}_{u,i} = \frac{1}{1+e^x} \tag{4.8}$$

We can interpret this rating estimation for an unknown item as its average mutual dependence with a user's frequent itemsets. The values for the

[3] Example 4.1 is derived from Agrawal et al. (1994).

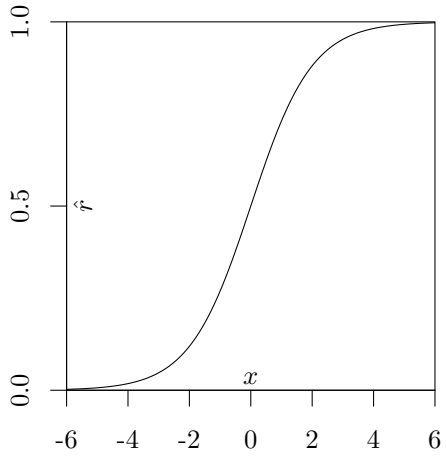


Figure 4.1: Logistic curve for converting rating predictions.

predicted ratings are related to the interpretations in Equation 4.5, as for x we take we take the logarithm of these values (Equation 4.7), and for $\hat{r}$ we map x to the logistic curve (Equation 4.8). The logistic curve is a sigmoid with two horizontal asymptotes. As such, the predicted ratings will always lie in the range $0 < \hat{r} < 1$. Here, $\hat{r} = 0.5$ ($x = 0$) indicates no relation, $0 < \hat{r} < 0.5$ ($x < 0$) indicates a negative correlation and $0.5 < \hat{r} < 1$ ($x > 0$) indicates positive correlation between the user's transactions and the item at hand. In practice, this means that values for $x > 6$ get a maximum rating, and values for $x < -6$ get a minimum rating (Figure 4.1). This range proved to be most suitable for our purposes.

Because we have already calculated the support for all frequent itemsets (in our *frequent itemset generation* step), we can easily calculate the lift for the association rules by looking up the support for f, i and $f \cup i$. To do this as efficiently as possible we use an algorithm which we call Apriori-Rec. Like the Apriori algorithm, Apriori-Rec is based on the downward closure of itemsets, which states that for an infrequent itemset, all its supersets must also be infrequent. Apriori-Rec works as follows:

1. Given $f \in F_1(u)$, First we identify the itemsets $f \cup i$ that are stored in our frequent 2-itemsets. The lifts are summed for association rules $f \Rightarrow i$ and the antecedents (f) are stored in a database.
2. For each consecutive value for k (starting with $k = 2$), the $(k + 1)$-itemsets are identified that contain i and a superset of the $(k - 1)$-itemset that we stored in our database. The lifts of the association rules are added to the sum and the k-size antecedents are added to our database.

3. If no frequent $(k+1)$-itemset is found that can be constructed from i and a superset of $(k-1)$-itemsets in our database, we divide the sum of the lifts by the length of the database and return the logarithm of the result as our predicted rating.

We favor the lift of an association rule over its confidence, because the lift normalizes the association rules by taking the support for both the antecedent and the consequent in account. This allows us to average the associations across itemsets. Our rating estimations automatically put more emphasis on association rules that are based on k-itemsets that have a high k. These itemsets are more likely to exhibit strong positive or negative correlations, because the expected difference in support between f and $f \cup i$ becomes smaller as k increases. This observation is based on the downward closure property of support, which we have described before when we discussed the APRIORI algorithm.

Note that the predicted ratings should be normalized when the actual values of the predictions are of importance. We can normalize a value $\hat{r}$ to a value r' that lies within an expected rating range $\mathcal{S}$ by performing a simple linear conversion (Equation 4.9).

$$r'_{u,i} = \hat{r}_{u,i} \cdot (\max \mathcal{S} - \min \mathcal{S}) + \min \mathcal{S} \tag{4.9}$$

Our rating prediction method is based on the approach by Davidson et al. (2010). Instead of taking the sum of all association rules they base their predictions on the association rules of a single seed itemset.

4.2 Preprocessing latent factor models

In this section we describe how association rule mining can be used in model-based recommendation approaches. We first revisit matrix factorization and we describe a problem that data sparsity might impose on a latent factor model. Then we explain how association rule mining can be used to solve this problem.

In Chapter 3 we have illustrated how matrix factorization works in a space where the ratings for all user-item pairs are known (i.e. $\mathcal{K} = \mathcal{R}$). In real-life systems this is never the case for recommender systems. The density[4]of the user-item matrix depends on the application, but can exceed 10^{-10} in certain commercial systems (Bambini, Cremonesi, & Turrin, 2011). In Section 3.1 we have touched upon the subject of data sparsity when we compared item-based and user-based approaches. Let us recapitulate the two problems that data sparsity imposes to traditional collaborative filtering approaches:

Limited coverage refers to the notion that user or item similarity weights are calculated based on the intersection of their vectors. Therefore users (or

[4]Density is the ratio between the number of ratings and the product of the number users and the number of items.

items) can not be neighbors if they have no common items (or users), despite the fact that they might have similar interests.

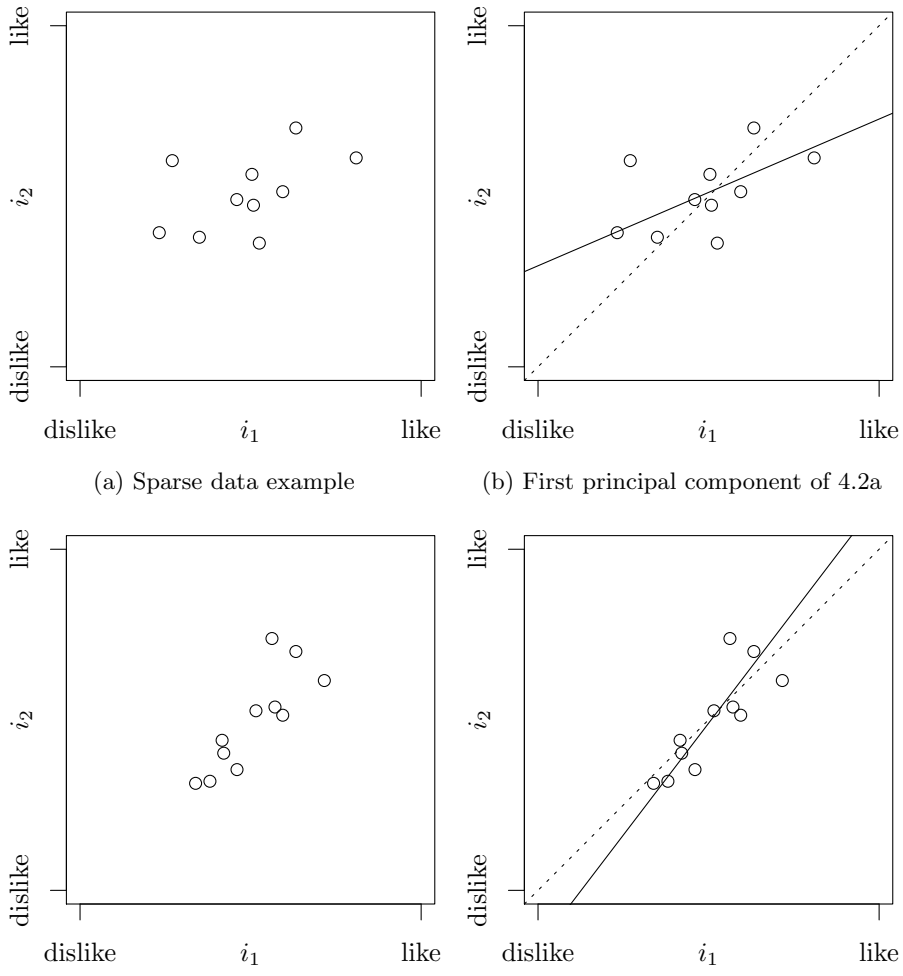
Sensitivity to sparse data that arises from the limited coverage. In Section 3.1 we argued that we are forced to choose between a small number of high-confidence neighbors or a large number of small-confidence neighbors. In such a case, a small number of high-confidence neighbors is generally preferred. However, there is also the possibility that both neighbors and their intersections are rare, despite the fact that similar users or items exist. In such a case, both item-based and user-based approaches would lead to biased recommendations.

Matrix factorization was introduced to offer a solution to both of these problems. By minimizing the regularized squared error on the set of known ratings we derive factors that reduce the dimensionality of our original space. This makes our derived matrix relatively more dense and therefore increase the coverage of our data. The factors supposedly uncover the latent "semantics" in our input space. However, how do we *really* know that our principal components or singular values are an accurate representation of these latent semantics? Let us illustrate the problem we envision in Example 4.2. Here, we revisit the example that we used in Section 3.2.

Example 4.2. The rating matrix in Section 3.3 contained two ratings (one for each item) for each of the n users. As such, the rating matrix was *full*, i.e. $\mathcal{K} = \mathcal{R}$. By analyzing the rating pattern, we were able to extract the principal component, which allowed us to reduce the dimensionality of our data. To illustrate an additional utility for the principal component, imagine that a single rating for one of the users of our system is missing. We know that the best rating for *both* items combined p_u (the rating in the latent feature space) is simply the projection of the user's rating we do have r_u on the direction of the principal component q (see Section 3.2). As such the principal component provides us with a method to estimate unknown ratings.

Let us apply the same process of calculating the principal component in a more realistic setting where we use only a subset, or **sample** of the users (i.e $\mathcal{K} \subset \mathcal{R}$).[5] Figure 4.2a depicts the same item space as before, except that each user has a sample probability of 0.1 (thus each user has 90% chance to be sampled out). Needless to say, sampling the data should have no effect on the ideal latent semantical relation between i_1 and i_2. After all, we *know* what this relation is, because we have observed the full rating matrix $\mathcal{K} = \mathcal{R}$ and its first principal component before (Example 3.3). "Hiding" a part of the data (by looking at a sample $\mathcal{K}$ instead of the full matrix $\mathcal{R}$) does not change the fact that $\mathcal{R}$ is our ground truth.

[5]The term *sample* is common in the field of statistics, where the superset of a sample (i.e. our full matrix $\mathcal{R}$) is coined with the term *population*. As this term intuitively does not translate well to describe usage data for recommender systems, we continue using the term *full matrix*. Similarly, in the context of recommender systems we would rather refer to $\mathcal{K}$ with the notion *observed data*, but in this example the term *sample* is more adequate.



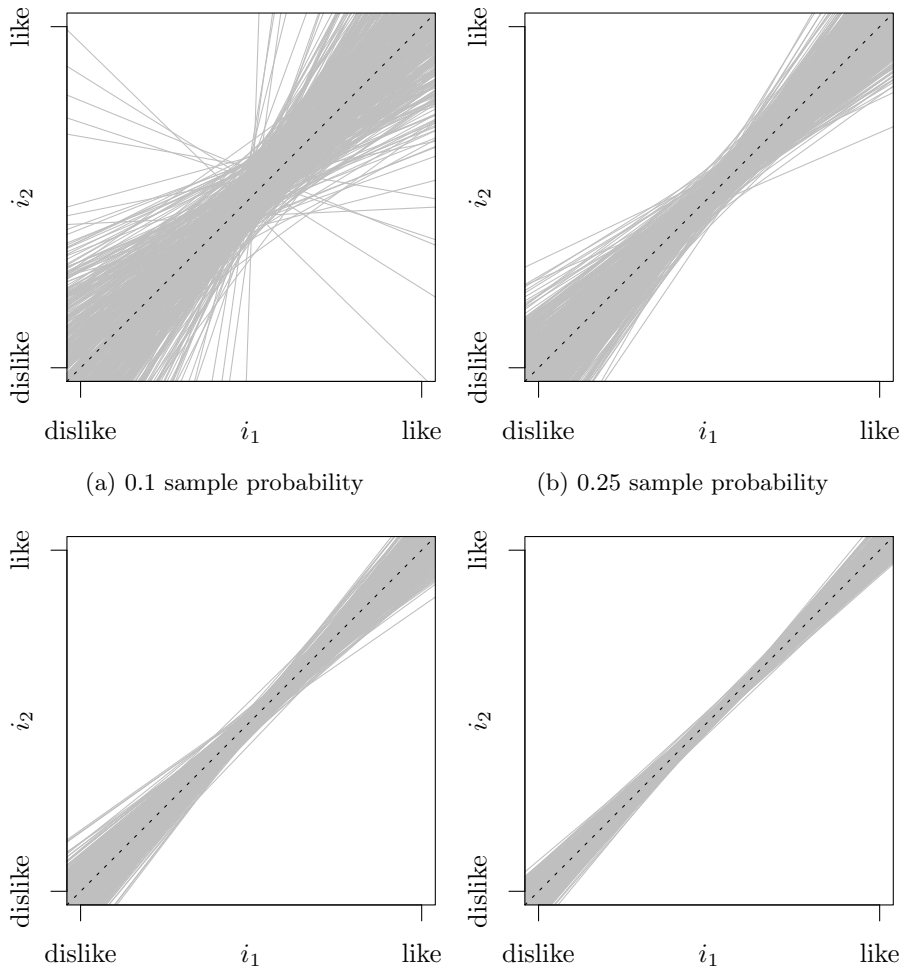
(a) Sparse data example

(b) First principal component of 4.2a

(c) Another sparse data example

(d) First principal component of 4.2c

Figure 4.2: First principal component of sparse data compared to the original first principal component.



(a) 0.1 sample probability

(b) 0.25 sample probability

(c) 0.5 sample probability

(d) 0.75 sample probability

Figure 4.3: Principal component variation for different sample probabilities.

Yet, Figure 4.2b shows that sampling down the data can greatly affect the direction of the first principal component. Figure 4.2c and Figure 4.2d even show that yet another sample of the same data has a entirely different first principal component altogether. In Figures 4.3a through 4.3d we have iterated the sampling process and we have plotted the observed first principal components for different sample probabilities. For a 0.1 sample probability, we find that the direction of the first principal component is so dependent on the sample that at times we are unable even to observe the positive correlation between our items. The probability that our sample's first principal component closely reflects the original principal component remains high, but Figure 4.3a shows that we even find a *negative* correlation every so often – the exact opposite of the actual relation in our original data. Figures 4.3b, 4.3c and 4.3d show, an increase in sample probability leads to a decrease in the variation of our principal components.

From this we conclude that the principal components (and accordingly, the singular values) in our observed (sampled) data do not necessarily reflect the true latent semantical relations in our full item space. Therefore, we argue that preprocessing the input data of a matrix factorization model might improve the precision of its dimensionality reduction.

Let us describe how association rule mining can be used to make the rating matrix more dense prior to to training the prediction model. In this application of association rule mining we only consider single items (1-itemsets) for the antecedent and consequent of the association rules. From these items we select the frequent items that have a support greater or equal to the *minsup* threshold. We establish association rules for all the frequent items. For simplicity we denote the items with i and j instead of F. Given two items i and j and their association rule $i \rightarrow j$, we use the confidence the rule to determine whether or not we can assume that a user who has rated i, but not j, would rate j the same as i. If we are confident enough that a user would rate j the same as item i, we transpose its rating for i to j.

In order to determine if the confidence of an association rule is significant enough, we compare it to a predetermined threshold *minconf*, which is a value between 0 and 1. We transpose all ratings between i and j for which Equation 4.10 holds. This means that for each user u that has rated either i or j, $r_{u,i} = r_{u,j}$.

$$c(i \Rightarrow j) \geq minconf \tag{4.10}$$

While the confidence of an association rule might be might an inaccurate indicator at times, we expect the information gain in the dense matrix to outweigh the error rate. We hypothesise that matrix factorization prediction models are more accurate if we extend our data model with transposed ratings of items that have high confidence association rules.

Chapter 5

Experimental setup

Information overload is a symptom of our desire to not focus on what's important. It is a choice.

— Brian Solis

In our experiments we compare the three systems that we have described in Chapter 4. The prediction models of these systems are based on:

1. Item-based association rule mining.
2. Matrix factorization of original data.
3. Matrix factorization of preprocessed data.

In Section 5.1 we describe the dataset that we use to evaluate our systems. Then, in Section 5.2 we explain the procedure of our experiments and the different parameterizations of our systems. We conclude this chapter with a description of our evaluation methods in Section 5.3.

5.1 Dataset

We perform our experiments on the *MovieLens* dataset.[1] This dataset has regularly been used in literature to evaluate other other recommendation approaches (Herlocker & Konstan, 2004; Berkovsky et al., 2008; Shani & Gunawardana, 2011; Park & Tuzhilin, 2008). The data was collected for the MovieLens project by the GroupLens Research Project at the University of Minnesota between 1998 and 2000. It consists of 100,000 ratings from 943 users on 1,682 items, where each user has rated at least 20 items. The rating matrix has a density of 6.305×10^{-2}. Data tuples are in the form $\langle$ user ID, item ID, rating timestamp $\rangle$, where ratings are expressed as an integer between 1 and 5. We do

[1] Available at `http://www.grouplens.org/node/73`.

not consider these ratings in all of our experiments, as our systems are designed to recommend items based on implicit (unary) feedback as well. As such, we conduct two groups of experiments in which we compare the performance of our recommender systems. In the first group of experiments we train the prediction models on the original explicit ratings in the dataset, and in the second group of experiments we process the data tuples as unary classifications.

5.2 Parameter selection

All three prediction models that we evaluate are parameterized. In this section we describe the rationale behind the values that we choose. Table 5.1 lists the values and provides an overview of the parameter combinations for each of our systems.

For our item-based association rule mining method we have to specify a *minsup* parameter for the minimum support of an association rule. We only use the prediction rules for which the antecedent has minimum support to calculate ratings. This means that for most *minsup* values our model can not predict ratings for all items $i \in \mathcal{I}$, because some items might depend on infrequent rules.[2] Support is expressed a value between 0 and 1 that is dependent on the size of the dataset. We base our values for *minsup* on the frequencies *mincount* that yield *minsup*. The values for *mincount* are in turn multiples of the average item frequency in our dataset (which is 60). Due to the lack of prior research in our domain we find it hard to hypothesize what the effect of *mincount* is on the performance of our system. Therefore, we consider a wide range of values for this parameter.

The prediction model of our first method depends on a logarithm to express the ratings in a sensible scale (see Equation 4.5). Because our prediction model is novel in the context of recommender systems, we are unsure what the most appropriate base for the logarithm is. For an application in a related domain the natural (i.e. base e) logarithm has been proposed, but here the metric was used on a dataset that was a couple orders of magnitude smaller than ours (Bouma, 2009). We expect that the size of the dataset is of importance for choosing the most appropriate logarithm base. For this reason, we experiment with different base values.

The systems that are based on matrix factorization have a parameter f that denotes the dimensionality of the latent factor space $\mathcal{M}^f$. In our choice on selecting values for f we consider the advice by Koren et al. (2009). In their experiments they use values up to 1,500 and report that performance improves as the factor model's dimensionality increases. However, as their latent factor model was trained on a dataset that is a couple orders of magnitude larger than ours, we hypothesize that our prediction model is better off using smaller values for f. Therefore, we experiment with a range of low values for f as well as with the values that were suggested by Koren et al. (2009).

[2] In most practical use cases rating sparsity is not an issue, because we are only interested in the top-n recommendations.

Table 5.1: Parameters for the experiments.

parameter	values	models
Minimum frequency (*mincount*)	0–30, interval 5	1
Logarithm base ($\log_b$)	2, e, 10	1
Dimensionality (f)	10–100, 200, 500, 1000	2, 3
Minimum confidence (*minconf*)	0.05 – 0.30	3

For preprocessing input data for matrix factorization we consider a range of *minconf* values that has been proposed in a similar application domain (Tan et al., 2006). The *minconf* values that do not lead to any transposed ratings are not included in our results. We hypothesize that sufficiently high values for *minconf* will lead to the best results, because these lead to item pairs with high confidence association rules. We expect that for low values of *minconf* too many ratings get transposed.

5.3 Evaluation

In this section we describe the metrics and experiments that we use to evaluate our systems. We will not blur our narrative with excess details on the metrics that we discuss. For a comprehensive study of prediction-based and rank-based metrics for recommender systems, we refer to Shani and Gunawardana (2011) and Herlocker and Konstan (2004).

5.3.1 Measuring rating predictions

Prediction-based metrics compare the predicted ratings to the actual ratings in the test set. In our evaluation we use the MAE and RMSE metrics.

Mean absolute error (MAE) measures how close predicted ratings are to the actual values. As the name suggests, it is the average of the absolute errors, see Equation 5.1.

$$MAE = \frac{1}{n} \sum_{k=1}^{n} |\hat{r}_k - r_k| \tag{5.1}$$

Here, $\hat{r}_k$ is the predicted value and r_k is the corresponding actual rating. We denote with n the size of our test set. As the individual users and items are not of importance for this evaluation metric, k can refer to any (u, i) pair.

Mean square error (MSE) is similar method to measure the deviation between the predicted ratings and the actual values. The difference between MAE and MSE is that the latter puts more emphasis on large errors (by means

of the "square"), see Equation 5.2.

$$MSE = \frac{1}{N}\sum_{k=1}^{n}(\hat{r}_k - r_k)^2 \tag{5.2}$$

The **root mean square error** (RMSE) is more commonly used to evaluate a system in this sense. This metric has also been used to judge the *Netflix Prize* competition, for example. As the name implies, it is the square root of the MSE value (Equation 5.3).

$$RMSE = \sqrt{MSE} \tag{5.3}$$

MAE and RMSE are closely correlated in practice, but the two metrics might discover subtle differences between approaches or configurations.

5.3.2 Measuring ranking performance

Rank-based metrics compare a ranked list of rated items to a list of predictions of these items. We try to determine the correct order of a set of previously rated items for each user (their reference ratings) and measure how similar our systems' predicted rankings are to the correct order. In our experiments, we use Spearman's ρ and Kendall's τ metrics for this purpose. Unlike prediction metrics, ranking metrics are more appropriate to evaluate systems that will be used to present ranked recommendation lists to the user (Herlocker & Konstan, 2004). In our implementation we measure the ability of a system to rank-order a set of items for which we know the true ranking.

Pearson correlation p (Equation 5.4) measures the extent to which there is a linear correlation between two lists of values. It is widely used as a measure of the covariance of their z-scores (i.e. the number of standard deviations above the mean). Here $\bar{\cdot}$ denotes the mean value for some variable $\cdot$.

$$p_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{5.4}$$

Spearman's ρ measures the extent to which two different rankings agree, independent of the actual values. It is computed in the same manner as Pearson's correlation, except that the values for r and $\hat{r}$ are transformed into their ranks r' and $\hat{r}'$ (thus, given the highest $r_{u,i}$, $r'_{u,i} = 1$ and for the lowest $r_{u,j}$, $r'_{u,j} = |r_u|$). The correlations are computed on these ranks (Equation 5.5).

$$\rho_u = \frac{\sum_{i=1}^{n}(r'_{u,i} - \bar{r'_u})(\hat{r}'_{u,i} - \bar{\hat{r}'_u})}{\sqrt{\sum_{i=1}^{n}(r'_{u,i} - \bar{r'_u})^2} \cdot \sqrt{\sum_{i=1}^{n}(\hat{r}'_{u,i} - \bar{\hat{r}'_u})^2}} \tag{5.5}$$

Kendall's τ is similar to Spearman's ρ in that it measures the similarity between two rankings, but instead uses the number of concordant rating-prediction

pairs and the number of discordant rating-prediction pairs to calculate the covariance between ratings and predictions. (Equation 5.6). Here, C is the number of rating-prediction pairs that the system has recommended in the proper ranked order, and D is the number of pairs that the system has recommended in the wrong order. Moreover, TR is the number ratings that have tied rank, and TP is the number of predictions that have shared ranks.

$$\tau_{r_u,\hat{r}_u} = \frac{C - D}{\sqrt{(C + D + TR) \cdot (C + D + TP)}} \tag{5.6}$$

Despite their simplicity, Spearman's ρ and Kendall's τ have not been used extensively in the evaluation of recommender systems (Herlocker & Konstan, 2004). We use these metrics because they are an effective alternative for prediction-based metrics and because they are well understood across different research fields. Similar to MAE and RMSE, Spearman's ρ and Kendall's τ are often closely correlated in practice (Shani & Gunawardana, 2011), but might uncover subtle differences in performance between systems or configurations, as their focus differs slightly.

5.3.3 Evaluating performance

To evaluate our systems using the metrics that we have described in this section, we perform experiments that estimate how a prediction model will perform in practice. Preferably, this is done in a manner that simulates the target application as closely as possible. As such, we evaluate our system on a **per-user** basis, which means that address the significance of a metric based on the average and the variance of that metric's score for each user.

Offline evaluation is based on the idea of hiding some subset of the interactions in order to simulate the knowledge of how a user would rate unseen items. A possible procedure is to simply leave out the last n ratings for each user, and train on the prediction model on the remaining interactions. However, this would make the assumption that temporal aspects are of importance in our application domain. While this assumption is tempting given some studies (e.g. Shani et al., 2005; Zimdars et al., 2001; Su et al., 2000), we rather not make this assumption, as it is beyond the scope of this thesis to address it. Instead, we assume that temporal aspects do not influence user preferences. This coincides with our methodology, as association rules are not considered to address causal relations. Moreover, we design our experiments in a way that reflects this assumption.

To evaluate the performance of each parameter combination, we perform a repeated "all but n" experiment. Following Herlocker and Konstan (2004), we split the data in two unequal disjoint sets, where the test set has exactly 10 ratings per user. The remaining data is used to train the prediction model. This procedure is repeated in two "splits" to account for biases in the dataset.

For our prediction-based metrics we use the prediction model to estimate the 10 unknown ratings for each user in the test set, and for our rank-based metrics

we compare the ranking of the ratings to the ranking of the predictions. Because the number of unknown ratings for each user is equal ($n = 10$) we can average our metrics across users and splits. As such, our results for each configuration of a system contain two sets of evaluation metrics for each user – one for each split. Because our dataset consists of 943 users, this means that 1886 user-centric evaluation experiments are ran for each parameter combination. The results for each parameter combination, as well as the tests that we use address the significance of their results are presented in Chapter 6.

Chapter 6

Results

> *On two occasions I have been asked,*
> *"Pray, Mr. Babbage, if you put into the machine wrong figures, will the right answers come out?"*
> *I am not able rightly to apprehend the kind of confusion of ideas that could provoke such a question.*
>
> — Charles Babbage, *Life of a Philosopher*

This chapter is divided in three sections in which we discuss our experimental results. We begin this chapter with a description of the statistical tests that we use to account for the significance of our results (Section 6.1). Then, we compare the prediction and rating performance between systems and configurations in Section 6.2. This section is laid out as follows. First we compare the best performances that were obtained by our three systems. Then we examine each of our systems in detail, by reviewing the extent to which the individual parameters account for performance.

6.1 Significance

In this thesis we have conducted experiments to identify the most promising recommender system in a range of different approaches. Because we considered a wide range of different values for the parameters (or **independent variables**) of our systems, the resulting scores (or **dependent variable**) might only marginally differ between configurations. For these results there is a probability that the configuration that performed best did so because the experiment was more suitable for that specific configuration. It is important that we can be confident that the approach or configuration that we label as most promising (for real-world applications) will also be a good choice for yet unseen data the system will be faced with in the future (Ricci et al., 2011). Therefore, we perform statistical tests to identify configurations that performed significantly better than others.

Significance can be interpreted as follows. If the difference between two configurations is **significant**, the probability that these results were obtained by chance is below a given confidence level. In other words, two systems differ significantly from one another if the probability p that one system's score is obtained by the other is below a certain threshold. By convention of our research field (and many others), these thresholds are defined to be $p = .05$ and $p = .01$. In the following sections we denote these confidence levels with * and ** respectively.

Let us recall the hypotheses that we have drawn in Chapter 1. Even though we will not address our hypotheses until Chapter 8, we believe that it is important to relate our hypotheses to our notion of statistical significance at this point in our narrative.

H$_1$ Association rule mining can be used as a suitable alternative for current state-of-the-art collaborative filtering approaches.

H$_2$ Association rule mining can be used to improve state-of-the-art collaborative filtering approaches.

For our intents and purposes, we consider matrix factorization to be a state-of-the-art collaborative filtering approach. Let us denote with ARM, MF and ARMMF our three systems (item-based association rule mining, matrix factorization, and matrix factorization enhanced with association rule mining respectively). Recall that our offline evaluation method aims to identify the best candidates for user studies and online testing. For this reason, we compare the best performing configurations of our three systems. Let us denote the average scores obtained by these configurations with μ_{ARM}, μ_{MF} and μ_{ARMMF}. We infer the following rules to either accept or reject our hypotheses:

H$_1$ accept if $\mu_{\text{ARM}} \geq \mu_{\text{MF}}$, reject if $\mu_{\text{ARM}} < \mu_{\text{MF}}$

H$_2$ accept if $\mu_{\text{MF}} < \mu_{\text{ARMMF}}$, reject if $\mu_{\text{MF}} \geq \mu_{\text{ARMMF}}$

We use an **analysis of variance** (ANOVA) in conjunction with **Tukey's test** to determine if the scores of two systems are (significantly) different. ANOVA is a statistical test to determine whether or not the means (μ) of all of the three (or more) groups are equal. As such, it is a generalization of the common pairwise Student's t-test to more than two systems. A t-test determines if the means of *two* groups are equal – which is what our hypotheses require. However, doing multiple t-tests results in an increased chance of committing a type I error, which occurs when a null-hypothesis (i.e. an hypothesis that predicts no difference) is rejected, despite being true. Instead, we use Tukey's test to identify the means that are significantly different in the ANOVA. Tukey's test compares the difference of every pair of means to any other pair of means, and identifies the differences that are greater than the expected difference of the means, based on their standard deviation.

Table 6.1: Best performance for all three systems.

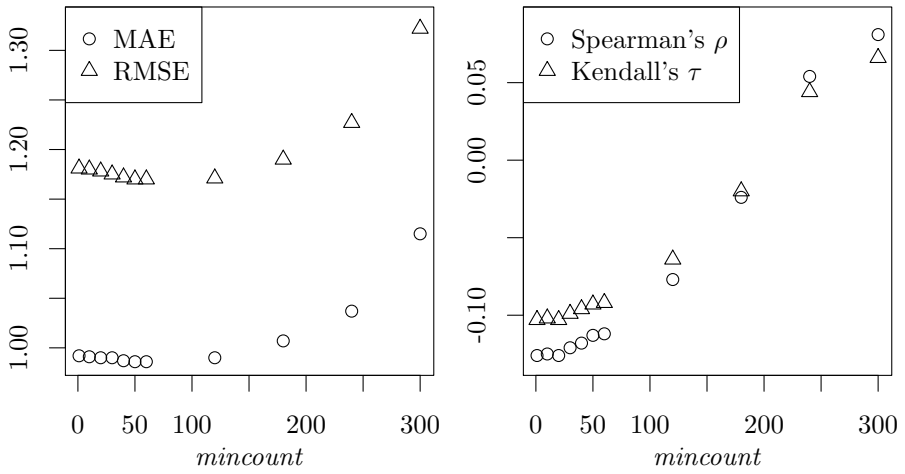
	MAE	RMSE	ρ	τ
ARM	0.986 (0.295)**	1.170 (0.336)**	.081 (0.346)**	.066 (0.284)**
MF	**0.768** (0.250)	**0.945** (0.288)	**.383** (0.332)	**.317** (0.280)
ARMMF	**0.768** (0.251)	**0.945** (0.288)	.382 (0.331)	.316 (0.278)

6.2 Performance

In Table 6.1 we list the MAE, RMSE, Spearman's ρ and Kendall's τ scores of the best-performing configurations for each of our three approaches. Our system that performs matrix factorization on the original data (MF) performs best, but not significantly better than the enhanced matrix factorization (ARMMF) approach (MAE $p = .999$, RMSE $p = .996$, $\rho p = .992$, $\tau p = .986$). Both matrix factorization systems have significantly lower (i.e. better) MAE and RMSE scores than our item-based association rule mining (ARM) system ($p < .01$ for both systems and metrics), and significantly higher (i.e. better) Spearman's ρ and Kendall's τ scores (also for both systems and metrics $p < .01$).

Table A.1, Table A.2 and Table A.3 (Appendix A) list the performance for each configuration of our ARM, MF and ARMMF systems respectively. In these tables, the best scores for each evaluation metric are in boldface, and the asterisks indicate the scores that are significantly worse for each metric (* for $p < .05$ and ** for $p < .01$). In the following subsections we elucidate the results for each system, by exploring the extent to which the individual parameters influence the performance of that system. Moreover, we examine the "predictiveness" of each system by exploring correlations between our evaluation metrics.

We present a multitude of figures in the following subsections that plot the results of our systems. It is important to note that the ranges for the y-axes of these plots are chosen to represent the maximum variance for the evaluation metric at hand. As such, these figures should not be used to compare performance between systems (for this purpose we refer the reader to Appendix A). Instead, these figures are presented to display the effects of the independent variables (i.e. our parameters) on the dependent variables (i.e. a system's performance). Because we have only examined a subset of the possible values for our parameters, we present each figure as a scatterplot rather than a continuous model. The properties of the relations that we describe (such as optima or asymptotes) are based on these "scattered" observations, but might not be definitive ("continuous").



(a) Prediction (lower is better)

(b) Ranking (higher is better)

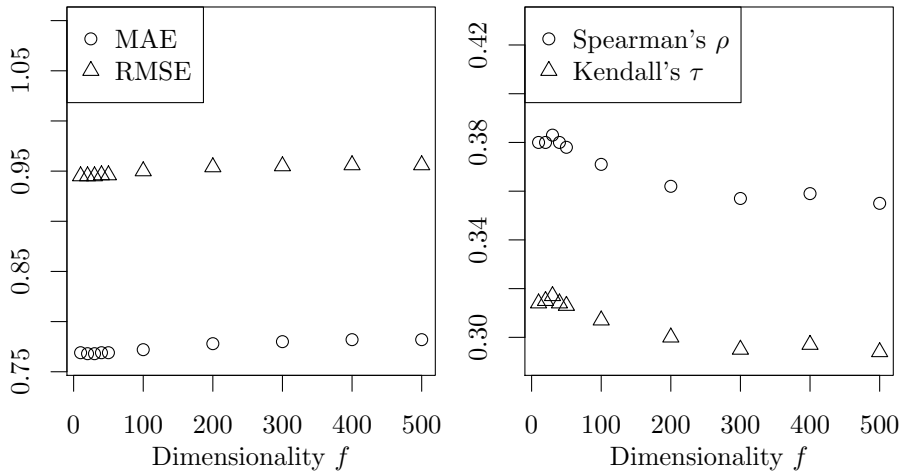
Figure 6.1: Performance of the association rule mining system (ARM) normalized with $\log_{10}$.

6.2.1 Association rule mining

The best prediction performance (MAE, RMSE) for the ARM system was achieved when the association rule lifts were normalized with their base-10 logarithm ($\log_{10}$). Figure 6.1 shows the system's prediction performance for this logarithm base. The best configuration had *mincount* set to 60. This configuration scored significantly better than all configurations that used the binary logarithm ($\log_2$) or base-e logarithm ($\log_e$) to normalize the association rule lifts ($p < .01$). The difference with most other configurations of *mincount* (mc) was not significant, except for $mc = 240$ and $mc = 300$ (both $p < .05$).

Figure 6.1a shows that there seem to be local optima for prediction performance when the association rule lifts are normalized with their base-10 logarithm. For $\log_2$ and $\log_e$ these optima do not seem to lie in the range of *mincount* values that we have tested.

Contrary to the prediction performance, the best ρ and τ were obtained with the highest value for the *mincount* parameter ($mc = 300$). All other *mincount* values resulted in significantly lower ranking performance ($p < .01$ except for $mc = 240$: $p < .05$). Ranking performance does not depend on the chosen logarithm for normalization, because the choice of logarithm is merely a question of scale. Figure 6.1b shows the relation between *mincount* and rating performance (ρ and τ).



(a) Prediction (lower is better)

(b) Ranking (higher is better)

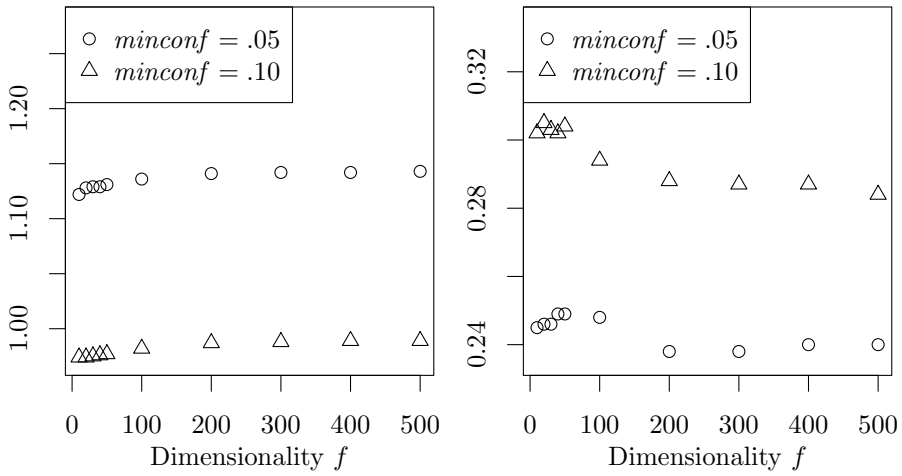
Figure 6.2: Performance of the matrix factorization system (MF).

6.2.2 Matrix factorization of original data

The results for the matrix factorization system are shown in Figure 6.2. The latent factor space for the best performing matrix factorization system on the original data had 30 dimensions ($f = 30$), but the difference in performance compared to other configurations is not significant (MAE $p = .509$, RMSE $p = .865$, Spearman's ρ $p = .112$, Kendall's τ $p = .101$). Moreover, both prediction and ranking metrics display a similar response to the prediction model's dimensionality, as all metrics seem to approach a horizontal asymptote as the dimensionality of the prediction model increases (see Figure 6.2a for the prediction based metrics, where a lower score is better, and Figure 6.2b for Spearman's ρ and Kendall's τ respectively, where higher scores are better).

6.2.3 Matrix factorization of preprocessed data

The ARMMF system was designed to provide us with a more accurate latent factor representation compared to the MF system, by making the input matrix more dense prior to applying matrix factorization. The original input matrix (for the MF system) had an average matrix density of 5.725×10^{-2}. In our ARMMF we have transposed ratings between item pairs based on how confident we were that there existed a positive correlation between the items. A *minconf* of .05 led to an average of 25796.5 transposed ratings (in the two "all but n" experiments), whereas an average of 3881.5 ratings were transposed with a *minconf* of .10 and an average of 384.5 ratings were transposed with a *minconf* of .15. These ratings led to average matrix densities of 6.666×10^{-2}, 5.920×10^{-2} and 5.749×10^{-2} respectively. Any *minconf* $\leq .05$ led to a higher matrix density,



(a) RMSE (lower is better)

(b) Kendall's τ (higher is better)

Figure 6.3: Performance of the preprocessed matrix factorization system (ArmMf) for *minconf* = .05 and *minconf* = .1 (*minconf* = .15 is similar to Figure 6.2).

but to lower performance. Moreover, any *minconf* $\geq$ .20 led to no transposed ratings. In such cases Mf and ArmMf perform identically. For this reason, Table A.3 reports on .05 $\leq$ *minconf* $\leq$ .15.

Any increase in matrix densitiy did not lead to better performance. Unremarkably, a minimum confidence of .15 did not lead to significantly different performance compared to the original matrix factorization system, as their matrix densities are very similar. All configurations for *minconf* = .05 led to significantly worse performance ($p < .01$), while most configurations for .10 led to similar performance, except for $f = 200$ and $f = 500$ ($p < .05$).

Contrary to the original latent factor model, the best predicting matrix factorization system on preprocessed data had 20 dimensions ($f = 20$). The best rating performance was obtained with $f = 30$, similar to the original model. As the performance for the Mf system and ArmMf system with *minconf* = .15 are nearly identical, we refer to Figure 6.2 for a visualization of its results. Figure 6.3a shows that the dimensionality parameter f has a similar effect on prediction performance (RMSE) for *mincount* = .05 and *mincount* = .10 (except that the scores are worse). As Figure 6.3b shows, the same holds for the effect on rating performance (Kendall's τ).

Chapter 7

Discussion

> *He knew that all the hazards and perils were now drawing together to a point: the next day would be a day of doom, the day of final effort or disaster, the last gasp.*
>
> — J.R.R. Tolkien, *The Return of the King*

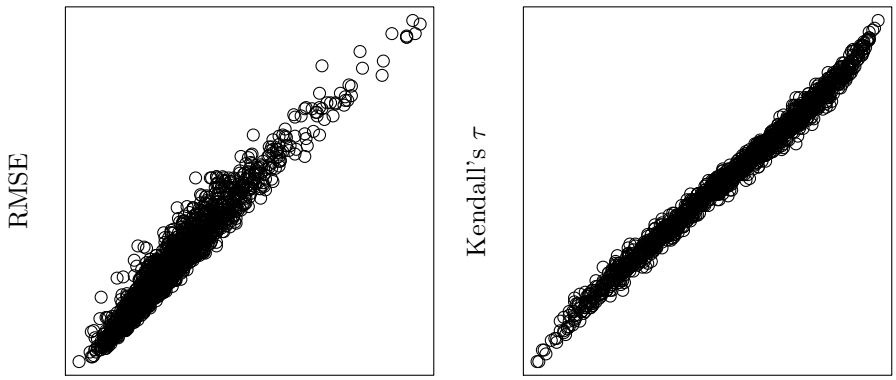
Regarding our evaluation methodology, two important questions remain unanswered in light of our experimental results:

1. To what extent are our results indicative for the "real-world" performance of our system?

2. Which of the advantages and disadvantages of our individual evaluation metrics a have significant effect on their outcomes?

It is beyond the scope of this thesis to address the former question. The metrics that we have used to evaluate our experiments are subject to the disadvantages of offline evaluation. In particular, we must assume that a user's past behavior is the only predictor for future behavior (Shani & Gunawardana, 2011). Moreover, we acknowledge that our choice for a repeated "all but n" experiment gives rise to the assumption that user behavior is not subject to temporal dynamics. In the conclusion of this thesis (Chapter 8) we suggest some directions for future research regarding these assumptions.

While it is beyond the scope of this thesis (as well) to answer the second question in detail we tentatively address them in this section. We address the this question by partly replicating a comparison experiment proposed by Herlocker and Konstan (2004). In their survey of offline evaluation metrics, the authors "examined the extent to which the different evaluation metrics agreed or disagreed" (p.33). Despite the fact that their research was not comprehensive (as their evaluation was based solely on the results a neighborhood-based collaborative filtering approach), the authors observe relationships between metrics that



(a) Prediction metrics (b) Rating metrics

Figure 7.1: Positive correlation between evaluation metrics.

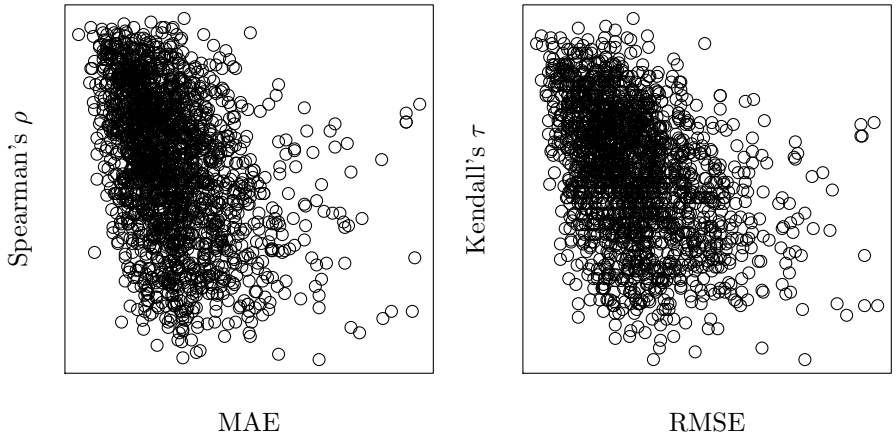
should be investigated further. As such, they "encourage researchers using other families of collaborative filtering algorithms to replicate [their] work" (p.34).[1]

In line with Herlocker and Konstan (2004), we compare the user-based results of different evaluation metrics for a random subset of user evaluations for all systems ($n = 1000$). In analyzing the data we notice that there is a strong correlation between the individual metrics for the two evaluation approaches. Plots of these positive correlations between metrics can be found in Figure 7.1. For MAE and RMSE (Figure 7.1a) the correlation coefficient was .966 ($p <$ .001, 95% confidence interval .965 to .968). For Spearman's ρ and Kendall's τ (Figure 7.1b) the correlation coefficient was .996 ($p <$ .001, 95% confidence interval .996 to .996).

The weaker correlations in Figure 7.2 suggest that Spearman's ρ and Kendall's τ provide substantially different outcomes from MAE and RMSE. For MAE and Spearman's ρ (Figure 7.2a) the correlation coefficient was $-.437$ ($p <$ .001, 95% confidence interval $-.458$ to $-.416$). For RMSE and Kendall's τ the correlation coefficient was $-.429$ ($p <$ .001, 95% confidence interval $-.450$ to $-.407$). It should be noted that the principal direction of the data does indicate a correlation between the prediction-based and ranking-based metrics.[2] However, the "agreement" between the two groups of evaluation metrics is much lower than the agreement within the groups (i.e. between MAE and RMSE, and Spearman's ρ and Kendall's τ respectively).

[1] This encouragement from Herlocker and Konstan (2004) and the argument for comparability served as two of our motivations to use the MovieLens dataset for evaluation.

[2] If prediction-based and ranking-based metrics were related the figures would display a strong *negative* correlation. This is due to the fact that for MAE and RMSE a lower score is considered better, while for Spearman's ρ and Kendall's τ a higher score is better.



(a) MAE and Spearman's ρ (b) RMSE and Kendall's τ

Figure 7.2: Absense of correlation between groups of evaluation metrics.

Our observations are in line with Herlocker and Konstan (2004), who observed a similar relation between prediction metrics (MAE and Pearson correlation) and ranking metrics (Spearman's ρ and Kendall's τ). As such, we answer our first question as follows. Prediction-based and ranking-based metrics do not fully agree on the (user-based) evaluation of a system. However, the individual metrics for each of these groups *do* agree. Therefore, we conclude that the advantages and disadvantages of individual prediction-based *or* rank-based metrics have no effect on their outcomes.

Chapter 8

Conclusion

> *A conclusion is simply the place where you got tired of thinking.*
>
> — Dan Chaon, *Stay Awake*

In this thesis we have explored some applications of association rule mining for recommender systems. In Chapter 2, the relevant concepts for recommender systems were introduced and the importance of such systems was discussed. Chapter 3 explained how collaborative filtering recommender systems work. In this chapter, we introduced latent factor models as the state-of-the-art collaborative filtering approach today.

The goal of this thesis is to explore the extent to which recommender systems based on association rule mining perform compared to state-of-the-art collaborative filtering approaches. We have presented two novel approaches for predicting ratings in Chapter 4. These approaches differ from traditional item-based collaborative filtering approaches in that they rely on basket data (individual transactions) to learn the relations between items. These relations are expressed as so called association rules. The first approach predicted a user's utility for an item by exploiting the association rules that were evident in previous interactions with the system. Our second approach used association rules for preprocessing a latent factor model.

In Chapter 5 we have described our dataset and experimental setup. Here, we have also presented the offline evaluation method that we used. Our experimental results were presented in Chapter 6 and discussed in Chapter 7.

This chapter draws a conclusion on the extent to which association rule mining can be used for recommender systems. First, we answer our research questions. Then, we comment on our problem statement. We conclude this thesis with some suggestions for future research.

In this thesis we posed the following research question:

RQ1 To what extent can association rule mining be used as an alternative to state-of-the-art collaborative filtering?

Based on some promising examples (i.e. Mobasher et al., 2000; Davidson et al., 2010), we hypothesised that association rule mining is suitable alternative for current state-of-the-art collaborative filtering approaches (H1). However, based on our experimental results we reject this hypothesis. Our system based on a latent factor model – which we consider to be a state-of-the-art collaborative filtering approach – performed significantly better than our association rule mining approach.

Our second research question was as follows:

RQ2 To what extent can association rule mining be used to improve state-of-the-art collaborative filtering recommender systems?

In Chapter 4 we have discussed the effect of data sparsity on latent factor models. Based on our observations in that chapter we hypothesized that association rule mining can be used to improve state-of-the-art collaborative filtering recommender systems (H2). However, our results indicate that preprocessing latent factor models, by transposing ratings between strongly correlated items, does not have a positive effect on performance. None of the latent factor models that used association rule mining (to make the input data more dense) performed better than our best performing "baseline" latent factor model. However, configurations that used only the most confident association rules did not perform significantly worse than the baseline. Therefore, we reject our second hypothesis as well, because no improvements in performance were found.

Based on our research we conclude that further research is required on association rule mining for recommender systems. Our offline research indicates that association rule mining can not be used to improve upon state-of-the-art collaborative filtering recommendations approaches for items of low complexity. For this reason, we discourage service providers (such as *Bibliotheek Midden Brabant*) to implement our systems for recommending books, movies and music tracks at this point. Instead, we encourage researchers to further develop the systems that were proposed, and to explore the extent to which association rule mining is applicable for recommending items of high complexity (such as insurance policies, jobs, and travel plans). Based on our results we discourage to perform user studies at this point. However, we note that our results are subject to the disadvantages of offline evaluation. As such, they are by no means a *definitive* indicator for "real world" performance.

We advise researchers that are interested in the applications of association rule mining for recommender systems to take our results in consideration when they design their algorithms. We encourage them to use similar (offline) evaluation methods, prior to performing user studies or online evaluation. However, we do note that some of the metrics were highly correlated. In this sense, our observations are in line with previous research (Herlocker & Konstan, 2004; Shani & Gunawardana, 2011). As such, it might suffice to use a single prediction-based metric and a single rank-based metric to evaluate the systems.

During our research we came across some noteworthy observations that are not specifically related to our research questions. Neither our literature study,

nor our experimental results provided us with satisfying explanations for these observations. As such, we suggest the following directions for future research:

- In our experiments we have tested a variety of different system configurations. In our choices for the configurations of our latent factor model we followed Koren et al. (2009), who won the *Netflix Prize* competition with a similar model. Despite the fact that they used a similar evaluation method, our results do not match with their findings. We expect this to be due to the size of the dataset. As such, we encourage researchers to investigate the effect of dataset size on the performance of collaborative filtering recommender systems.

- It is suggested that offline evaluation methods are subject to the assumption that a user's past behavior is the only predictor for future behavior (Shani & Gunawardana, 2011). We argue that cross-validation experiments (such as x-fold or our implementation of "all but n") give rise to a conflicting assumption. In taking random subsets of our dataset for evaluation (i.e. as test set), we assume that user preferences are not subject to temporal dynamics. Past research has presented strong evidence *against* this assumption (Shani et al., 2005; Su et al., 2000; Zimdars et al., 2001). In response some offline evaluation experiments have been proposed that do take (the possibility of) temporal dynamics into account (Herlocker & Konstan, 2004; Shani & Gunawardana, 2011). However, none of these experiments are as solid as cross-validation in guarding against Type I, Type II and Type III errors ("rejecting an hypothesis when it is true", "accepting an hypothesis when it is false", and "accepting or rejecting an hypothesis for the wrong reasons" respectively). As such, we encourage researchers to design more solid offline experiments that regard the recommendation problem as being time-dependent.

References

Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, *17*(6), 734–749. Retrieved from `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1423975` doi: 10.1109/TKDE.2005.99

Adomavicius, G., & Tuzhilin, A. (2011). Context-aware Recommender Systems. In F. Ricci, L. Rokach, B. Shapira, & P. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 217–253). Springer. doi: 10.1007/978-0-387-85820-3

Agrawal, R., Imielinski, T., Swami, A., & Jose, S. (1993). Mining Association Rules between Sets of Items in Large Databases. In *Sigmod '93 proceedings of the 1993 acm sigmod international conference on management of data* (pp. 207–216). ACM. Retrieved from `http://dl.acm.org/citation.cfm?id=170072` doi: 10.1145/170035.170072

Agrawal, R., Srikant, R., & Jose, S. (1994). Fast Algorithms for Mining Association Rules. In *Proceedings of the 20th vldb conference* (pp. 487–499).

Amatriain, X., Jaimes, A., Oliver, N., & Pujol, J. M. (2011). Data Mining Methods for Recommender Systems. In F. Ricci, B. Shapira, L. Rokach, & P. B. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 39–71). Springer. doi: 10.1007/978-0-387-85820-3

Anderson, C. (2008). *The Long Tail*. Hyperion.

Balabanović, M., & Shoham, Y. (1997). Fab: Content-Based, Collaborative Recommendation. *Communications of the ACM*, *40*(3), 66 – 72.

Bambini, R., Cremonesi, P., & Turrin, R. (2011). A Recommender System for an IPTV Service Provider: a Real Large-Scale Production Environment. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 299–332). Springer. doi: 10.1007/978-0-387-85820-3

Berkovsky, S., Kuflik, T., & Ricci, F. (2008, November). Mediation of user models for enhanced personalization in recommender systems. *User Modeling and User-Adapted Interaction*, *18*(3), 245–286. Retrieved from `http://link.springer.com/10.1007/s11257-007-9042-9` doi: 10.1007/s11257-007-9042-9

Berry, M. W., Dumais, S. T., & O'Brien, G. W. (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review*, *37*(4), 573–595.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *Journal of Machine Learning Research*, *3*, 993–1022. Retrieved from http://dl.acm.org/citation.cfm?id=944937

Bogers, T. (2009). *Recommender Systems for Social Bookmarking*. Phd thesis, Tilburg University.

Bogers, T., Rasmussen, R. R., Sebastian, L., & Jensen, B. (2013). Measuring Serendipity in the Lab: The Effects of Priming and Monitoring. In *Proceedings of iconference 2013* (pp. 703–706). iSchools. doi: 10.9776/13325

Bouma, G. (2009). Normalized (Pointwise) Mutual Information in Collocation Extraction. In *Proceedings of the biennial gscl conference* (pp. 31–40).

Burke, R. (2007). Hybrid Web Recommender Systems. In P. Brusilovsky, A. Kobsa, & W. Nejdl (Eds.), *The adaptive web* (pp. 377–408). Berlin, Germany: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-540-72079-9_12 doi: 10.1007/978-3-540-72079-9_12

Celma, O. (2008). *Music recommendation and discovery in the long tail*. Phd thesis, Universitat Pompeu Fabra.

Celma, O., & Cano, P. (2008). From hits to niches? or how popular artists can bias music recommendation and discovery. In *Proceedings of the 2nd kdd workshop on large-scale recommender systems and the netflix prize competition.* (pp. 5:1–5:8). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=1722154 doi: 10.1145/1722149.1722154

Chen, Z., Meng, X., Zhu, B., & Fowler, R. H. (2000). Websail: From on-line learning to web search. In *Proceedings of the first international conference on web information systems engineering* (pp. 206–213). Retrieved from http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=882394 doi: 10.1109/WISE.2000.882394

Cunningham, P., Bergmann, R., & Schmitt, S. (2001). Websell: Intelligent sales assistants for the world wide web. *KI*, *15*(1), 28–32. Retrieved from http://www.wi2.uni-trier.de/publications/2001_KIWebsell.pdf

Davidson, J., Livingston, B., Sampath, D., Liebald, B., Liu, J., Nandy, P., ... Lambert, M. (2010). The YouTube video recommendation system. In *Proceedings of the fourth acm conference on recommender systems - recsys '10* (p. 293). New York, New York, USA: ACM Press. Retrieved from http://portal.acm.org/citation.cfm?doid=1864708.1864770 doi: 10.1145/1864708.1864770

Deerwester, S., Dumais, S. T., Furnas, G. W., & Landauer, T. K. (1990). Indexing by Latent Semantic Analysis. *1Journal of the American Society for Information Science*, *41*(6), 291–407. Retrieved from http://www.cob.unt.edu/itds/faculty/evangelopoulos/dsci5910/LSA_Deerwester1990.pdf

Desrosiers, C., & Karypis, G. (2011). A Comprehensive Survey of Neighborhood-based Recommendation Methods. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 107–144). Springer. doi: 10.1007/978-0-387-85820-3

Felfemig, A., Friedrich, G., Jannach, D., & Zanker, M. (2011). Developing

Constraint-based Recommenders. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 187–251). Springer. doi: 10.1007/978-0-387-85820-3

Fischer, G. (2001). User modeling in humancomputer interaction. *User Modeling and User-Adapted Interaction*, *11*(1-2), 65–86. Retrieved from `http://link.springer.com/article/10.1023/A%3A1011145532042`

Goldberg, D., Nichols, D., Oki, B., & Terry, D. (1992). Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, *35*(12), 61–70. Retrieved from `http://dl.acm.org/citation.cfm?id=138867`

Good, N., Schafer, J. B., Konstan, J. A., Borchers, A., Sarwar, B., Herlocker, J., & Riedl, J. (1999). Combining Collaborative Filtering with Personal Agents for Better Recommendations. In *Proceedings of aaai'99* (pp. 439–446). AAAI Press.

Hayes, C., & Cunningham, P. (2001). Smart radio – community based music radio. *Knolwedge-Based Systems*, *14*(3-4), 197–201.

Herlocker, J., & Konstan, J. (2004, January). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems*, *22*(1), 5–53. Retrieved from `http://portal.acm.org/citation.cfm?doid=963770.963772http://dl.acm.org/citation.cfm?id=963772` doi: 10.1145/963770.963772

Hill, W., Stead, L., Rosenstein, M., & Furnas, G. (1995). Recommending and evaluating choices in a virtual community of use. In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 194–201). Retrieved from `http://dl.acm.org/citation.cfm?id=223929`

Hofmann, T. (2004, January). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems*, *22*(1), 89–115. Retrieved from `http://portal.acm.org/citation.cfm?doid=963770.963774` doi: 10.1145/963770.963774

Koren, Y., & Bell, R. (2011). Advances in Collaborative Filtering. In F. Ricci, L. Rokach, & B. Shapira (Eds.), *Recommender systems handbook* (1st ed., pp. 145–186). Springer. doi: 10.1007/978-0-387-85820-3

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42–49. Retrieved from `http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5197422`

Linden, G., Smith, B., & York, J. (2003). Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *Internet Computing: IEEE*, *7*(1), 76–80. doi: 10.1109/MIC.2003.1167344

Maes, P. (1994, July). Agents that reduce work and information overload. *Communications of the ACM*, *37*(7), 30–40. Retrieved from `http://portal.acm.org/citation.cfm?doid=176789.176792` doi: 10.1145/176789.176792

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval* (1st ed.). Cambridge University Press. Retrieved from `http://books.google.nl/books?id=t1PoSh4uwVcC`

Mobasher, B., Cooley, R., & Srivastava, J. (2000). Automatic personalization based on Web usage mining. *Communications of the ACM*, *43*(8), 142–151. Retrieved from http://dl.acm.org/citation.cfm?id=345169 doi: 10.1145/345124.345169

Montaner, M., López, B., & De La Rosa, J. L. (2003). A Taxonomy of Recommender Agents on the Internet. *Artificial Intelligence Review*, *19*(4), 285–330. Retrieved from http://link.springer.com/article/10.1023/A%3A1022850703159# doi: 10.1023/A:1022850703159

Newman, M. E. J. (2005). Power laws, Pareto distributions and Zipfs law. *Contemporary Physics*, *46*(5), 323–351. Retrieved from http://www.tandfonline.com/doi/abs/10.1080/00107510500052444 doi: 10.1080/00107510500052444

Park, Y.-J., & Tuzhilin, A. (2008). The long tail of recommender systems and how to leverage it. *Proceedings of the 2008 ACM conference on Recommender systems - RecSys '08*, 11. Retrieved from http://portal.acm.org/citation.cfm?doid=1454008.1454012 doi: 10.1145/1454008.1454012

Parsons, J., Ralph, P., & Gallagher, K. (2004). Using Viewing Time to Infer User Preference in Recommender Systems. In *Aaai workshop in semantic web personalization.*

Resnick, P., Iacovou, N., & Suchak, M. (1994). GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 acm conference on computer supported cooperative work* (pp. 175–186). Retrieved from http://dl.acm.org/citation.cfm?id=192905

Ricci, F., Cavada, D., Mirzadeh, N., & Venturini, A. (2006). Case-based travel recommendations. In D. R. Fesenmaier, K. W. Vöber, & H. Werthner (Eds.), *Destination recommendation systems: Behavioural foundations and applications* (pp. 67–93). CABI.

Ricci, F., Rokach, L., Shapira, B., & Kantor, P. (2011). *Recommender Systems Handbook* (1st ed.). New York, NY: Springer. doi: 10.1007/978-0-387-85820-3

Sakagami, H., & Kamba, T. (1997, September). Learning personal preferences on online newspaper articles from user behaviors. *Computer Networks and ISDN Systems*, *29*(8-13), 1447–1455. Retrieved from http://linkinghub.elsevier.com/retrieve/pii/S0169755297000160 doi: 10.1016/S0169-7552(97)00016-0

Schafer, J. B., Frankowski, D., Herlocker, J., & Sen, S. (2007). Collaborative Filtering Recommender Systems. In *The adaptive web* (pp. 291–324). Springer. doi: 10.1007/978-3-540-72079-9_9

Shani, G., & Gunawardana, A. (2011). Evaluating Recommendation Systems. In F. Ricci, L. Rokach, B. Shapira, & P. Kantor (Eds.), *Recommender systems handbook* (1st ed., pp. 257–297). Springer. doi: 10.1007/978-0-387-85820-3

Shani, G., Heckerman, D., & Brafman, R. I. (2005). An MDP-Based Recommender System. *Journal of Machine Learning Research*, *6*, 1265–1295.

Shardanand, U., & Maes, P. (1995). Social Information Filtering: Algorithms

for Automating "Word of Mouth". In *Proceedings of the sigchi conference on human factors in computing systems* (pp. 210–217).

Su, Z., Lu, Y., Yang, Q., Zhang, H.-j., Road, Z., & District, H. (2000). WhatNext: A Prediction System for Web Requests using N-gram Sequence Models. In *Proceedings of the first international conference on web information systems engineering* (pp. 214–221). doi: 10.1109/WISE.2000.882395

Tan, P. N., Steinbach, M., & Kumar, V. (2006). Association analysis: Basic concepts and algorithms. In *Introduction to data mining* (pp. 327–414). Addison-Wesley. Retrieved from `http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Association+Analysis:+Basic+Concepts+and+Algorithms#1`

Voorhees, E. M. (2002). The Philosophy of Information Retrieval Evaluation. In C. Peters, M. Braschler, J. Gonzalo, & M. Kluck (Eds.), *Evaluation of cross-language information retrieval systems* (pp. 355–370). Springer. Retrieved from `http://link.springer.com/chapter/10.1007/3-540-45691-0_34` doi: 10.1007/3-540-45691-0

Witten, I. H., Frank, E., & Hall, M. A. (2011). *Data Mining: Practical machine learning tools and techniques* (3rd ed.). Burlington, MA: Morgan Kaufmann.

Zimdars, A., Chickering, D., & Meek, C. (2001). Using temporal data for making recommendations. In *Proceedings of the seventeenth conference on uncertainty in artificial intelligence* (pp. 580–588). Retrieved from `http://dl.acm.org/citation.cfm?id=2074093`

Appendices

Appendix A

System performance

Table A.1: Association rule mining performance (ARM).

mc	MAE	RMSE	ρ	τ
		$\log_2$		
1	1.219 (0.463)**	1.483 (0.508)**	−.126 (0.370)**	−.103 (0.302)**
10	1.218 (0.462)**	1.482 (0.508)**	−.125 (0.369)**	−.102 (0.301)**
20	1.216 (0.460)**	1.479 (0.506)**	−.126 (0.369)**	−.103 (0.301)**
30	1.212 (0.458)**	1.474 (0.505)**	−.121 (0.369)**	−.099 (0.301)**
40	1.206 (0.458)**	1.468 (0.502)**	−.118 (0.369)**	−.096 (0.301)**
50	1.201 (0.451)**	1.462 (0.500)**	−.112 (0.370)**	−.092 (0.303)**
60	1.198 (0.448)**	1.458 (0.497)**	−.112 (0.370)**	−.092 (0.303)**
120	1.168 (0.432)**	1.427 (0.484)**	−.077 (0.364)**	−.064 (0.297)**
180	1.142 (0.427)**	1.400 (0.480)**	−.024 (0.353)**	−.020 (0.287)**
240	1.128 (0.434)**	1.385 (0.484)**	.054 (0.352)*	.044 (0.288)*
300	1.181 (0.471)**	1.441 (0.514)**	**.081** (0.346)	**.066** (0.284)
		$\log_e$		
1	1.110 (0.422)**	1.361 (0.473)**	−.126 (0.370)**	−.103 (0.302)**
10	1.109 (0.420)**	1.359 (0.471)**	−.125 (0.369)**	−.102 (0.301)**
20	1.106 (0.418)**	1.356 (0.469)**	−.126 (0.369)**	−.103 (0.301)**
30	1.102 (0.414)**	1.350 (0.466)**	−.121 (0.369)**	−.099 (0.301)**
40	1.096 (0.411)**	1.343 (0.463)**	−.118 (0.369)**	−.096 (0.301)**
50	1.091 (0.406)**	1.337 (0.458)**	−.112 (0.370)**	−.092 (0.303)**
60	1.088 (0.403)**	1.333 (0.455)**	−.112 (0.370)**	−.092 (0.303)**
120	1.063 (0.385)**	1.306 (0.439)**	−.077 (0.364)**	−.064 (0.297)**
180	1.051 (0.384)**	1.291 (0.438)**	−.024 (0.353)**	−.020 (0.287)**
240	1.057 (0.399)**	1.294 (0.451)**	.054 (0.352)*	.044 (0.288)*
300	1.124 (0.451)**	1.369 (0.496)**	**.081** (0.346)	**.066** (0.284)*
		$\log_{10}$		
1	0.992 (0.307)	1.181 (0.350)	−.126 (0.370)**	−.103 (0.302)**
10	0.991 (0.306)	1.180 (0.349)	−.125 (0.369)**	−.102 (0.301)**
20	0.990 (0.305)	1.178 (0.347)	−.126 (0.369)**	−.103 (0.301)**
30	0.990 (0.302)	1.175 (0.344)	−.121 (0.369)**	−.099 (0.301)**
40	0.987 (0.300)	1.172 (0.340)	−.118 (0.369)**	−.096 (0.301)**
50	0.986 (0.293)	1.170 (0.337)	−.113 (0.370)**	−.093 (0.302)**
60	**0.986** (0.295)	**1.170** (0.336)	−.112 (0.370)**	−.092 (0.303)**
120	0.990 (0.293)	1.171 (0.334)	−.077 (0.364)**	−.064 (0.297)**
180	1.007 (0.312)	1.190 (0.356)	−.024 (0.353)**	−.020 (0.287)**
240	1.037 (0.353)*	1.227 (0.396)*	.054 (0.352)*	.044 (0.288)*
300	1.115 (0.426)**	1.322 (0.464)**	**.081** (0.346)	**.066** (0.284)*

Table A.2: Matrix factorization of original data performance (MF).

f	MAE	RMSE	ρ	τ
10	0.769 (0.252)	0.945 (0.289)	.380 (0.334)	.314 (0.281)
20	0.768 (0.250)	0.945 (0.288)	.380 (0.331)	.315 (0.278)
30	**0.768** (0.250)	**0.945** (0.288)	**.383** (0.332)	**.317** (0.280)
40	0.769 (0.250)	0.946 (0.288)	.380 (0.333)	.314 (0.280)
50	0.769 (0.251)	0.946 (0.289)	.378 (0.333)	.313 (0.280)
100	0.772 (0.251)	0.950 (0.289)	.371 (0.330)	.307 (0.278)
200	0.778 (0.251)	0.954 (0.289)	.362 (0.335)	.300 (0.282)
300	0.780 (0.251)	0.955 (0.288)	.357 (0.338)	.295 (0.283)
400	0.782 (0.251)	0.956 (0.288)	.359 (0.338)	.297 (0.283)
500	0.782 (0.252)	0.956 (0.289)	.355 (0.341)	.294 (0.285)

Table A.3: Matrix factorization of preprocessed data performance (ArmMf).

f	MAE	RMSE	ρ	τ
		$minconf = .05$		
10	0.902 (0.338)**	1.122 (0.388)**	.296 (0.335)**	.245 (0.281)**
20	0.905 (0.336)**	1.128 (0.386)**	.296 (0.336)**	.246 (0.283)**
30	0.906 (0.337)**	1.129 (0.388)**	.295 (0.337)**	.246 (0.283)**
40	0.905 (0.339)**	1.129 (0.390)**	.298 (0.334)**	.249 (0.282)**
50	0.905 (0.338)**	1.131 (0.388)**	.299 (0.336)**	.249 (0.284)**
100	0.908 (0.336)**	1.136 (0.386)**	.297 (0.333)**	.248 (0.281)**
200	0.912 (0.334)**	1.141 (0.385)**	.286 (0.335)**	.238 (0.283)**
300	0.913 (0.335)**	1.142 (0.386)**	.285 (0.340)**	.238 (0.286)**
400	0.913 (0.336)**	1.142 (0.387)**	.287 (0.337)**	.240 (0.284)**
500	0.914 (0.337)**	1.143 (0.387)**	.288 (0.335)**	.240 (0.281)**
		$minconf = .10$		
10	0.789 (0.266)	0.974 (0.309)	.365 (0.335)	.302 (0.281)
20	0.787 (0.266)	0.974 (0.310)	.368 (0.337)	.305 (0.293)
30	0.787 (0.265)	0.975 (0.309)	.366 (0.333)	.303 (0.283)
40	0.787 (0.265)	0.976 (0.309)	.365 (0.334)	.302 (0.281)
50	0.788 (0.266)	0.977 (0.310)	.368 (0.333)	.304 (0.280)
100	0.793 (0.266)	0.982 (0.310)	.355 (0.332)	.294 (0.278)
200	0.798 (0.267)	0.987 (0.311)*	.347 (0.339)	.288 (0.284)
300	0.801 (0.267)*	0.988 (0.310)*	.348 (0.336)	.287 (0.281)
400	0.802 (0.267)*	0.989 (0.310)*	.347 (0.337)	.287 (0.282)
500	0.802 (0.267)*	0.989 (0.310)*	.344 (0.341)	.284 (0.284)
		$minconf = .15$		
10	0.769 (0.252)	0.945 (0.289)	.381 (0.334)	.315 (0.281)
20	0.768 (0.251)	0.946 (0.289)	**.382** (0.331)	**.316** (0.278)
30	**0.768** (0.251)	**0.945** (0.288)	.381 (0.331)	.316 (0.280)
40	0.770 (0.251)	0.947 (0.288)	.379 (0.334)	.314 (0.280)
50	0.769 (0.251)	0.947 (0.289)	.378 (0.333)	.313 (0.280)
100	0.773 (0.251)	0.951 (0.289)	.370 (0.328)	.307 (0.275)
200	0.779 (0.252)	0.956 (0.289)	.361 (0.336)	.299 (0.282)
300	0.782 (0.252)	0.957 (0.289)	.358 (0.338)	.296 (0.283)
400	0.783 (0.252)	0.958 (0.289)	.359 (0.338)	.297 (0.283)
500	0.783 (0.253)	0.958 (0.290)	.354 (0.341)	.293 (0.285)