Evaluating Recommender Systems

An evaluation framework to predict user satisfaction for recommender systems in an electronic programme guide context

Joost de Wit B.Sc.

TNO Information and Communication Technology

Delft, May 16, 2008

Supervising committee:
Dr. Ir. Djoerd Hiemstra (University of Twente),
Ir. Dolf Trieschnigg (University of Twente),
Ir. Erik Boertjes (TNO),
Drs. Stephan Raaijmakers (TNO).

Preface

It was January 2007 when Dolf Trieschnigg, my supervisor for the course Information Retrieval, first told me about the availability of graduation projects at TNO Information and Communication Technology. This was a bull's eye since I just started to orientate myself on a subject to investigate. I was also looking for a company where I could perform my research since I wanted to get familiar with working in a professional environment. TNO Information and Communication Technology was one of the companies that seemed interesting to me, so I contacted Stephan Raaijmakers for more information. The subject of the proposed research project, evaluating recommender systems, was completely new to me, but seemed fascinating. And it is.

In September 2008 I started my research by crafting the goals and research questions for the project. Now, almost nine months later, the research has resulted in the report that you just started reading. It marks the end of my life as a student (at least for now) and the start of my professional career at TNO. At TNO I can continue to work on personalisation and recommender systems.

Although I wrote this thesis the research would never have been completed without the support of other people. In the first place, I would like to thank my supervisors, Erik, Djoerd, Stephan and Dolf for supervising me throughout the research project. They were always there for me when I needed their help and/or expertise and provided loads of valuable comments and suggestions. I also like to thank Jan Telman for his help and advice with respect to statistics (and hiking through the Alps).

Furthermore I would like to thank all people that participated in the user study. Without their devotion to the demanding study, the research would not have been completed.

I would also like to thank my lovely girlfriend Linda, my family and my colleagues at TNO Information and Communication Technology for being all ears when I tried to explain my approach and ideas to them. They helped me to abstract from all the details and forced me to explain things clearly so I better understood it myself.

Finally, I owe special thanks to Mark Prins, who provided me with an inexhaustible source of Tiësto's Club Life, A State of Trance and other dance and trance music.

Joost de Wit,
May 16, 2008

Contents

1

Introduction to the Research Project

Nowadays the amount of content available to people is endless and it increases every minute. It is impossible to go through all the items to decide whether they are useful or not, and thus finding relevant items becomes more and more difficult. There are many situations in which there are too many options to choose from. Suppose you want to buy a book from an on-line bookstore selling millions of titles. Since most of the titles are unfamiliar to you, you cannot decide whether you would like it or not, but among all these books there must be a copy you will enjoy. Finding this item by browsing through the millions of titles offered by the bookstore is impossible, the more since you probably cannot judge the item's relevance without reading it first. This problem is analogous to finding relevant CD's, TV programmes, mp3 files, movies, news articles, movie rentals, papers, results from search queries, (micro) blog entries, and many many more. The growing popularity of user generated content like movies on YouTube, photos on Flickr and blog entries on Blogger add to a further growth of available content.

To overcome this so called "information overload" problem, in the mid-1990s researchers started to investigate recommender systems [AT05]. A recommender system (RS) uses knowledge about your preferences (and those of others) to recommend items you are likely to enjoy. Users can offer feedback on items they are familiar with for example, and the recommender system uses the information to predict their preference for yet unseen items and subsequently recommends items with the highest predicted relevance.

Many on-line shopping sites as well as other applications incorporate recommender systems in order to offer personalised services and to boost their sales. Some popular examples include Amazon.com, NetFlix, Last.fm, MyStrands and iTunes. Amazon.com recommends books, music, video's and many other products, while NetFlix recommends movies and Last.fm, MyStrands and the iTunes store recommend music. Recommender systems also emerge in Electronic Programme Guides (EPG), search engines and many other applications. TiVo is a popular example of a television-viewing service that offers both an EPG and a recommen-

dation system.

This research will primarily focus on recommender systems that recommend TV programmes as part of an EPG, since the number of TV channels (and thus the number of TV shows) that are available increased tremendously due to the introduction of digital television. Therefore people watching television also suffer from information overloaded and recommender systems start to emerge in this domain.

1.1. Problem background

Since recommender system research began in the mid-1990s many different algorithms for generating recommendations have been developed and nowadays a wide variety of recommender systems exist. This fosters the need for means to evaluate and compare different systems. Recommender system evaluation has proven to be challenging [HKTR04, AT05, HMAC02, ZMKL05] however, since a recommender system's performance depends on, and is influenced by many factors. Moreover, there is no consensus among researchers on what attributes determine the quality of a recommender system and how these attributes should be measured. Some call a recommender system successful when it delivers accurate recommendations, while others utilise user satisfaction as the main indicator of success. In fact over a dozen methods to determine a recommender system's quality can be identified in the literature [HKTR04].

Furthermore, the data set on which the recommender system operates greatly influences its performance [HKTR04, SKKR01]. A recommender system that performs well on a data set with many users and only a few items may perform worse on a set with a large number of items and much less users. Other examples of data set characteristics that affect a recommender system's performance are the rating density (number of ratings per user) and the rating scale.

Closely related to the first issue is the fact that the goal for which the recommender system is evaluated may differ. The evaluation objective can be to compare two recommendation algorithms with respect to accuracy for example, but one might also be interested in a systems potential to increase sales. Both goals require a completely different evaluation approach and the measurement of different attributes.

Finally, the recorded quality of a recommender system is only a snapshot in time. Users, for example, do not rate items consistently [CLA$^+$], since their liking of a certain item can differ, depending on mood, changing taste, and so on. On the other hand it is unlikely that users will notice a difference of say 0.001 in measured accuracy.

Despite these challenges, accuracy, the recommender system's ability to predict the user's rating for an item, is the evaluation attribute of choice for most researchers [HKTR04, AT05, SPUP02, SPUP05, MH04, ZMKL05, CKP07, BHL07, ZK07, CdVP$^+$07]. As a result they try to optimise a recommender system's performance with respect to accuracy. A recent example of this is in Netflix Prize, an initiative of Netflix, which is an on-line movie rental company that seeks to substantially improve the accuracy of their recommender system. The team that improves the accuracy of their system by at least 10% wins $1,000,000.

In the context of this research it is assumed that the ultimate goal of a recommender system is to satisfy its users and help them to cope with the information overload problem, since user satisfaction is thought of as the driving force behind other goals such as the increase

of sales. Although it is likely that accurate recommendations are important it might not be the only aspect that determines user satisfaction. To illustrates this consider a recommender system that recommends TV programmes for example. When a user likes *Friends* most, the system would be very accurate when it recommends 10 episodes of Friends. It is unlikely however that the user will be pleased with this list. So, user satisfaction probably does not always correlate with high recommender accuracy [ZMKL05, MAC+02, MRK06], therefore one should not only focus on accuracy when evaluating a recommender system.

1.2. Relevance of the problem

The introduction of digital television made it possible to broadcast more TV channels using the same amount of bandwidth as with traditional analogous television. Cable operators are therefore able to broadcast more content to their customers. The number of channels that can be broadcasted is still limited to the bandwidth of the cable however. When digital television is offered over IP (IPTV) there is no need to stream channels simultaneously and therefore the number of channels that can be offered is virtually endless. The amount of content that becomes available through the television will increase even more with the introduction of on-demand services such as video on demand. Recapitulating, people watching television are likely to become overloaded with content and recommender systems can help them find TV programmes that are relevant to them. Moreover, cable operators can offer a recommender system through their set-top box and can thus distinguish themselves from other operators.

It is important that these recommender systems can be properly evaluated. An accurate method for evaluation is mandatory to improve recommender systems help finding people what they seek and to build systems that are valued by its users.

1.3. Research question

The ultimate goal of the research is to develop a method that is able to predict the user satisfaction achieved by an EPG/RS system, based on various off-line metrics (*i.e.* metrics that do not require user feedback). The research question that will be answered by the thesis therefore reads:

How can the user satisfaction achieved by an EPG/RS system be predicted based on various off-line metrics?

The overall research question is subdivided into multiple sub questions. These sub questions are:

1. What recommendation techniques do exist, how do they work and what are their benefits / issues?
2. What is the state-of-the-art in recommender system evaluation? What aspects are important to evaluate?
3. What is user satisfaction and how can it be measured?
4. How do the individual properties of a recommendation list and user satisfaction cohere?

So, first detailed insight in the state-of-the-art in recommender systems and recommender system evaluation will be gained. Furthermore methods for measuring user satisfaction will be investigated. Then, after the user experiment has been completed, the coherence between the observed user satisfaction and the off-line metrics will be scrutinised. Finally, based upon the knowledge acquired by answering the sub research questions, the prediction method will be developed. The methods that are used to answer the (sub) research questions will be described in more detail in section 1.5.

1.4. About TNO Information and Communication Technology

The research will be performed at the Multimedia Technology division of TNO Information and Communication Technology [1] in Delft, The Netherlands. TNO Information and Communication Technology is a knowledge and innovation centre that brings together knowledge on ICT and telecom. With this knowledge it helps companies, governmental organisations and (semi) public organisations to innovate in ICT. TNO's approach to innovation is not limited to technology only, it also focuses on business processes, user-friendliness and financial aspects. The multimedia technology group's expertise is primarily focused on media mining and (digital) television.

1.5. Research methods

In order to perform a sound empirical research the use of proper and well-considered methods is important. This section discusses the methodology used to answer the overall research question as well as the sub research questions.

1.5.1. Engineering cycle

According to Wieringa the engineering cycle, the collection of tasks a rational problem solver would follow to solve a world problem (*i.e.* the difference between the way the world is and desired state of the world), has the following structure:

1. Problem investigation
2. Solution design
3. Design validation
4. Choose a solution
5. Implement the chosen solution
6. Implementation evaluation

So to solve a world problem the current situation must be investigated, possible actions must be considered, these actions must be ranked according to their problem-solving effect, one of these actions must be chosen and performed/implemented, and finally the result must be evaluated to see whether it has the desired effect [WH06].

In concrete terms this results in the following steps to be taken:

[1] http://www.tno.nl/content.cfm?context=overtno&content=overtnosub&laag1=32&item_id=63

1. Perform a literature study and an in-depth problem investigation.
2. Think of possible solutions that would solve the problem identified in 1. A possible solution might be to use an existing data set to develop the prediction method using a certain machine learning technique for example.
3. Verify the validity of each of the in step 2 proposed solutions (*i.e.* check to see whether the solution indeed solves the problem).
4. Choose the solution that best solves the problem.
5. Then the solution chosen in step 4 needs to be implemented. This might entail the construction of the evaluation application for example, or the development of the prediction method using a certain technique.
6. The final step is to verify whether the problem is indeed solved and to what extent. This means for example that the performance of the prediction method is evaluated.

Several of these steps are performed according to other research methods. These methods will be described in the following subsections.

1.5.2. Literature study

The research starts with an in-depth literature study of about six weeks. First the context of the problem (as described above) will be investigated and important aspects and issues with respect to recommender systems and recommender system evaluation will be identified. Next the aspects and issues identified will be investigated more thoroughly and detailed insight in recommender systems (*i.e.* existing systems, techniques, etc.) and the state-of-the-art in recommender system evaluation (*i.e.* methods, metrics, etc.) will be gained. The result of this literature study is described in chapters 2 and 3 and appendices A and B.

1.5.3. User Study

For the user study the following steps are taken:

1. Determine goals
2. Analyse what data is necessary to achieve these goals
3. Design the application that will be used to gather the data
4. Validate the design
5. Implement and test the evaluation application
6. Deploy the application

First the goals of the user study are determined, after which an analysis will be made of the data that should be gathered in order to be able to achieve these goals. Next the evaluation application needed to gather the data must be designed and a static prototype will be build. This prototype is then evaluated using a number of "friendly users", users that know the goals of the user study and help to verify whether these goals would be achieved with the prototype. Then the prototype is extended to the fully operational evaluation application. Again the friendly users help to test and evaluate the system. After the evaluation period, which will

last one week, the friendly users are interviewed and fill out a short questionnaire. After their feedback is incorporated in the evaluation application it will go "live" and everybody can subscribe to the research.

The user study will be described in more detail in Chapter 4.

1.5.4. Software Development

During the software development phase, in which the evaluation application will be developed, an iterative development process will be applied. An iterative development process allows you to cope with advancing insights that arise while building the system [Som00]. Each of the iterations encompasses a number of subsequent tasks: requirements elicitation, analysis and design, implementation and testing. Once multiple iterations are completed and the evaluation application is "finished" it will be deployed. The iterative development process is schematically displayed in Figure 1.1.

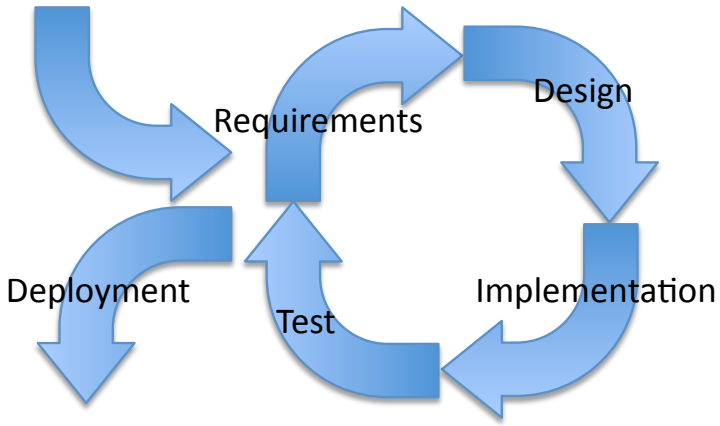

Figure 1.1.: Iterative development process

1.5.5. Validation of the model

To verify the validity of the model that is created to predict the user satisfaction based upon a number of off-line metrics, a cross-validation method will be applied. With cross-validation the data set is divided into training/validation set pairs that share data. This way even relatively small data sets can be used for validating the model [Alp04].

Section 6.2 will describe the validation of the prediction model in great detail.

1.5.6. Questionnaire

After the user study is completed the participants of this study are asked to complete a short questionnaire of ten open-ended and closed-ended questions. The purpose of the questionnaire is to learn from the participants, who are likely to be experienced users of a TV programme recommender system by then. The questionnaire will try to discover how important the dimensions that will be investigated are to the users (or at least how important they think

these dimensions are to them). Furthermore the questionnaire will be used to verify how the dimensions rated by the users are interpreted and whether important dimensions were missed.

An on-line questionnaire will be used to answer these questions since it is a quick and easy way to perform a survey under a large number of people [Bre06]. Moreover, an on-line questionnaire makes it easy to compile and analyse the data.

1.6. Reader's guide

The target audience of this thesis primarily consists of master students and researchers in the field of (EPG) recommender systems that focus on the evaluation of such systems. Another audience for which this thesis is relevant consists of professionals that actually want to evaluate a recommender system. Furthermore people that tend to build a recommender system or perform research on recommendation algorithms might benefit from this thesis. Some background knowledge with respect to mathematics and mathematical notations is desirable.

The structure of this thesis is largely analogous to the sub research questions presented in section 1.3 and the steps taken in the engineering cycle described in subsection 1.5.1. Chapter 2 describes the results of the literature study that investigated existing recommender system algorithms while Chapter 3 focusses on the state-of-the-art in recommender system evaluation. Next, Chapter 4 explains how the user study carried out in this research was conducted, what metrics were measured, how they were measures, and so on. Chapter 5 elaborates further on the coherence discovered among user satisfaction and numerous evaluation metrics. The ultimate goal of this research, the development and validation of a prediction method to predict user satisfaction, will be described in Chapter 6. Finally the thesis is concluded in Chapter 7. This chapter also describes future work provoked by this research.

2

State-of-the-Art in Recommender Systems

Since the appearance of the first papers on recommender systems in the mid-1990s a lot of research has been conducted in this area [AT05]. Many algorithms and techniques have been proposed by both industry and academia, and researchers are still working to improve recommender systems. This chapter gives an overview of existing recommendation approaches, as well as the state-of-the-art in the field.

In this chapter the following will be discussed

- Collaborative filtering algorithms (Section 2.2).
- Content-based filtering algorithms (Section 2.3).
- Hybrid forms that combine content-based and collaborative filtering (Section 2.4)

2.1. Recommendation approaches

Recommender systems try to predict what items a specific user would find interesting. These predictions are usually based on information about the user (called a user profile, user model, or user preferences), its interaction history (what the user bought, viewed, watched, read, etc.), properties of the items (called item features) and/or other users. Most systems try to predict how a user would rate a specific item, based upon this information, and then recommend the items with the highest ratings. In order to predict these ratings, different prediction techniques can be used. These prediction techniques come in three varieties: content-based, collaborative (or social) based and hybrid forms [AT05, vS05, AK07].

Content-based prediction techniques make recommendations based on the features of items that a user liked in the past. Recommender systems using this technique recommend items that are similar to these items. Collaborative filtering recommender systems recommend items

based on the items that people with a similar taste liked in the past. The third approach is a hybrid form that combines both collaborative and content-based techniques.

Recommender systems can also be classified according to their algorithmic technique [BHK98]. Two different classes can be identified:

- Memory-based techniques calculate recommendations in real-time over the entire data set.
- Model-based techniques learn a model from previous user activities and use this model to classify new content according to its interestingness.

Usually the prediction approaches mentioned above try to estimate a utility function $u : A \times B \rightarrow R_0$, that, given a user $a_i \in A$ and an item $b_k \in B$ returns how much the user will like that item. A and B are the sets of all users and items respectively. R_0 is a measure of utility, usually represented by an integer or real value. The items that are already rated (either implicitly or explicitly) can be gathered in a rating matrix $R = A \times B$. Table 2.1 shows an example of a rating matrix where each cell is a rating $r_i(b_k)$ of user a_i for item b_k. An empty cell means that the user did not rate the item. It is the recommender system's task to predict the value that should be in this cell.

	User			
Item	**1**	**2**	**3**	**4**
A	5	3	4	
B	4			5
C		2		4
D	4		4	
E	3		2	2
F	2	4		
G		5	5	

Table 2.1.: A sample rating matrix

In addition to algorithms that try to predict an item's utility to a user, there are also algorithms that try to order the items based on their interestingness without assigning an absolute utility value to it. This approach is called preference-based filtering and is focusing on the relative preference to a user [AT05]. Preference-based filtering will not be discussed in further detail. Collaborative filtering, content-based filtering and hybrid recommendation approaches will be discussed in the following sections.

2.2. Collaborative filtering

Collaborative filtering (CF) is the most widely used prediction technique in recommender systems [MH04, ZMKL05]. The technique makes recommendations based on ratings that users have assigned to items (*i.e.* the ratings in the rating matrix). There exist two different approaches to collaborative filtering: user-based and item-based CF. The underlying idea of user-based CF is that users who have rated the same items the same way (or at least similar) are likely to have a similar taste. These users are therefore likely to rate the same item the

same way. Item-based CF on the other hand, assumes that items that have been rated the same way in the past are probably similar items. A user will probably rate the item the same as he rated similar items.

2.2.1. User-based collaborative filtering

User-based CF consists of three basic steps. First a neighbourhood of users that are most similar to the active user (*i.e.* the user requesting recommendations) is constructed. This is done by calculating the similarity $sim(a_i, a_j)$ between user a_i and all other users $a_j \in A$. The l most similar users then form a_i's neighbourhood. In the second step the active user's ratings are predicted for items that are new to him (*i.e.* not yet rated by the user), but are rated by members of his neighbourhood. The third and final step is the construction of a recommendation list based on the predicted ratings. The prediction list does not necessarily consist of the top-N rated items, since list diversification might be applied for example [ZMKL05].

In the neighbourhood formation step the similarity between the active user and all other users must be calculated. There are several algorithms that provide a measure for user similarity. Two of them, Pearson's correlation and the cosine-based approach are described next.

$$sim(a_i, a_j) = \frac{\sum_{b_k \in B_{ij}} (r_i(b_k) - \overline{r_i})(r_j(b_k) - \overline{r_j})}{\sqrt{\sum_{b_k \in B_{ij}} (r_i(b_k) - \overline{r_i})^2 \sum_{b_k \in B_{ij}} (r_j(b_k) - \overline{r_j})^2}} \tag{2.1}$$

Pearson's correlation coefficient is a measure for the strength and direction of a linear relationship between two variables. Equation 2.1 shows the Pearson correlation coefficient that calculates the similarity between two users. In this equation B_{ij} indicates the set of items that are rated by both users a_i and a_j. $r_i(b_k)$ is a_i's rating from item b_k and $\overline{r_i}$ is the average rating value for user a_i as calculated in Equation 2.2. In this equation B_i is the set of items rated by user a_i.

$$\overline{r_i} = \frac{1}{|B_i|} \sum_{b_k \in B_i} r_i(b_k) \tag{2.2}$$

In the cosine-based approach the users are considered as vectors $\overrightarrow{a_i}$ and $\overrightarrow{a_j}$ in a m-dimensional space, where $m = |B_{ij}|$. The vectors thus represent the rating values for the items that were rated by both users. The similarity is then calculated as the angle between those two vectors, as shown in Equation 2.3.

$$\begin{aligned} sim(a_i, a_j) &= \cos(\overrightarrow{a_i}, \overrightarrow{a_j}) \\ &= \frac{\overrightarrow{a_i} \bullet \overrightarrow{a_j}}{|\overrightarrow{a_i}| \cdot |\overrightarrow{a_j}|} \\ &= \frac{\sum_{b_k \in B_{ij}} r_i(b_k) \cdot r_j(b_k)}{\sqrt{\sum_{b_k \in B_{ij}} r_i(b_k)^2} \sqrt{\sum_{b_k \in B_{ij}} r_j(b_k)^2}} \end{aligned} \tag{2.3}$$

When Pearson's correlation is applied to the running example as shown in Table 2.1, this results in coefficients presented in Table 2.2. The table shows that users 3 and 4 are very similar (correlation coefficient of 1.0), based on their ratings, while users 1 and 2 appear to be each others opposites (strong negative correlation).

	User			
User	**1**	**2**	**3**	**4**
1	1.0000	-0.9978	0.5335	0.9962
2	-0.9978	1.0000	0.8682	-1.0000
3	0.5335	0.8682	1.0000	1.0000
4	0.9962	-1.0000	1.0000	1.0000

Table 2.2.: Pearson's correlation coefficient indicates similarity among users

Once the user similarities are calculated, a neighbourhood $\widetilde{A}_i$ is formed of the l users that are most similar to the active user a_i. This neighbourhood is then used to make a prediction $p_i(b_k)$ for each item b_k that is rated by at least one neighbour, but not by the active user itself. The ratings of the neighbours are used to make this prediction. Equations 2.4a-c show three approaches, as identified in the literature, to aggregate the neighbours' ratings to form the prediction [AT05].

$$p_i(b_k) = \frac{1}{|\widetilde{A}_i|} \sum_{a_j \in \widetilde{A}_i} r_j(b_k) \tag{2.4a}$$

$$p_i(b_k) = \sigma \sum_{a_j \in \widetilde{A}_i} sim(a_i, a_j) \cdot r_j(b_k) \tag{2.4b}$$

$$p_i(b_k) = \overline{r_i} + \sigma \sum_{a_j \in \widetilde{A}_i} sim(a_i, a_j) \cdot (r_j(b_k) - \overline{r_j}) \tag{2.4c}$$

The normalisation factor σ is defined as:

$$\sigma = \frac{1}{\sum_{a_j \in \widetilde{A}_i} |sim(a_i, a_j)|}$$

Equation 2.4a just takes the average rating of all neighbours $\widetilde{A}_i$. Equation 2.4b takes the similarity between the active user and each of its neighbours into account and weights the rating of more similar users higher (*i.e.* a weighted sum approach). On top of this Equation 2.4c also takes into account how a certain user rates items on average.

In the running example, user 3 (a_3) his neighbourhood ($\widetilde{A}_3$) would consist of users 2 and 4 (when $l = 2$), since these two users are most similar to him. When the recommender tries to predict a_3's rating for item C it uses the ratings of a_2 and a_4, and combines them using one of the aggregation methods described in Equation 2.4. When Equation 2.4a is applied the

predicted value $p_3(C)$ would become $\frac{1}{2} \times 2 + 4 = 3$. Equation 2.4b results in $p_3(C) = 3.0706$ and Equation 2.4b in $p_3(C) = 3.2313$.

Once the active user's rating is predicted for all items rated by at least one of its neighbours, a recommendation list can be constructed and presented to the user. Usually this is done by taking the N items that have the highest predicted rating, but other approaches exist as well. One such approach is presented in [ZMKL05].

2.2.2. Item-based collaborative filtering

Item-based CF rests on the assumption that users that appreciate a certain item b_k probably also like a similar item b_l (and thus give them the same rating). The three step approach that is used in user-based CF is also applicable to item-base CF, besides that similarities are calculated over pairs of items rather than users.

Pearson's correlation can also be applied to calculate the similarity among items, based on the ratings of all users. For the running example this would result in de correlation coefficients as shown in Table 2.3.

	Item						
Item	**A**	**B**	**C**	**D**	**E**	**F**	**G**
A	1.0000	-1.0000	1.0000		0.8944	-1.0000	
B	-1.0000	1.0000	1.0000		-0.9487	1.0000	
C	1.0000	1.0000	1.0000		-1.0000	-1.0000	
D				1.0000			
E	0.8944	-0.9487	-1.0000		1.0000	-1.0000	
F	-1.0000	1.0000	-1.0000		-1.0000	1.0000	
G							1.0000

Table 2.3.: Pearson's correlation coefficient indicates similarity among items

These similarity measures can be used to determine the items that are most similar to the item for which the prediction is calculated (analogous to the neighbourhood in user-based CF). Based on the ratings for these similar items the predicted rating of the active user for the item is calculated.

As shown by the examples the number of similarities that need to be calculated differs when using user-based or item-based collaborative filtering. When user-based CF is applied roughly $\frac{1}{2}|A|^2$ similarities must be calculated, compared to $\frac{1}{2}|B|^2$ with item-based CF. So when the number of users is much smaller than the number of items it is computationally cheaper to apply user-based CF. Otherwise it is cheaper to use item-based CF.

2.2.3. Cold start problem

One of the main difficulties with collaborative filtering is that it suffers from the cold start problem. The cold start problem occurs when the recommender system cannot draw any inferences for users or items about which it has not yet gathered sufficient information. The first time a user visits a specific recommender system for example it did not have rated any of the items. Hence the recommender system does not know what that user's likes and dislikes

are. The same problem occurs when a new item is added to the recommender. Since nobody has ever rated that item the recommender system cannot know to which other items it is similar. Hence the recommender cannot recommend the item until a substantial number of users rated it.

2.3. Content-based filtering

In content-based recommendation systems, items are characterised by a set of predefined features. A movie for example, could be characterised by its cast, genres, theme, director and a description of the plot. This metadata can be provided explicitly or extracted from the content. Like items, users also have a profile. A user's profile contains information about its tastes, preferences and needs. Users can construct such a profile explicitly, but it can also be learned from the user's interaction with the recommender system. Content-based filtering uses these profiles to determine the interestingness of each item and recommends those that are most interesting.

There are two common approaches to determine an item's interestingness, a heuristics-based and a model-based approach. In a heuristics-based approach the profile of an item b_k is defined as a vector $\overrightarrow{w_k} = (w_{1k}, ..., w_{nk})$ of weights for each feature. These weights can be TF-IDF (term frequency-inverse document frequency, a concept from information retrieval) weights for example [Sal89, SB87]. A user's profile is specified likewise: $\overrightarrow{w_i} = (w_{1i}, ..., w_{ni})$. The interestingness of item b_k to user a_i can then be calculated using the cosine similarity measure as shown in Equation 2.3.

In a model-based approach predictions for an item's interestingness are not based on a heuristic formula, but on a model learned from the underlying data. Such a model is constructed using techniques such as Bayesian classifiers, machine learning, decision trees and neural networks [AT05].

Content-based filtering techniques are limited by the features of the items that are recommended by the system. When a feature is not associated with an item, the recommender system cannot use it in the recommendation process. Suppose that the movie "One flew over the cuckoo's nest" is not labelled with the actor "Jack Nicholson", a Jack Nicholson fan might not get this movie recommended. It thus is important to have a sufficient set of features, but feature extraction has proven to be either labour intensive (when it is done manually) or error prone (when it is done automatically) [AT05].

Another issue with content-based recommender systems is that they only recommend items that have a high similarity score with respect to the user's profile. This implies that only items that are similar to items that were rated in the past get recommended. This issue is known as overspecialisation and implies that users will never receive serendipitous recommendations [AT05].

2.4. Hybrid forms

Hybrid recommender systems combine both collaborative and content-based filtering to overcome the limitations (as described above) that each of the individual approaches suffers from. Adomavicius and Tuzhilin [AT05] identified four methods to combine collaborative and content-based filtering:

1. let both the content-based and collaborative technique make a prediction and then combine them,
2. incorporate some content-based techniques into a collaborative filtering approach,
3. incorporate some collaborative filtering techniques into a content-based approach, and
4. fully incorporate both techniques when building a hybrid recommender system.

The ratings predicted by each of the techniques can be combined by a linear combination or voting scheme for example, but it's also possible to choose one of the results based on certain metrics (like confidence). An example of incorporating content-based filtering in a CF recommender system is to construct a user's neighbourhood based on the user profiles, instead of rating history. The third approach can encompass dimension reduction techniques on a group of content-based profiles for example. A single unifying recommendation model can use both content-based and collaborative characteristics (like age and gender of users or the genre of movies) in a single rule-based classifier for example.

2.5. Conclusion

This chapter presented three types of prediction strategies that can be used to predict a user's rating for an item that is not yet rated by him. The first and most widely used strategy, called collaborative filtering, makes recommendations based on ratings that other users have assigned to items. Based on these ratings the recommender tries to find either users that are most similar to the active user or items that are similar to the item for which the user's rating is predicted. The first approach is called user-based collaborative filtering, the latter is called item-based collaborative filtering.

The second strategy presented in this chapter is called content-based filtering. With content-based filtering items are represented by a set of characteristics, called features. Each user has a user profile that describes which features are valued by the user and which are not. The recommender system then tries to match the feature vectors of the items against the user profile in order to predict the user's rating.

Since both techniques have limitations hybrid approaches were developed. By combining collaborative and content-based filtering techniques the cold start problem that collaborative filtering suffers from can be overcome for example. So, in conclusion, hybrid recommender systems combine the best of two worlds and help to overcome the limitations of both collaborative and content-based filtering.

3

State-of-the-Art in Recommender System Evaluation

The previous chapter presented the most commonly used recommendation techniques. Each technique has adherents that claim it to be an improvement over some other techniques, given a particular purpose. However, no common notion of recommender system quality exists and therefore comparing different systems is difficult. Many researchers recognised this problem and several contributions that try to solve it have been presented. This chapter will present the state-of-the-art in recommender system evaluation.

In this chapter the following will be discussed

- Approaches to evaluate recommender systems (Section 3.1).
- More than 15 different evaluation metrics in 7 dimensions (Sections 3.2–3.8)

3.1. Evaluation approaches

Recommender systems can be evaluated using off-line analysis, various live user experimental methods or a combination of these two approaches [HKTR04]. Off-line analysis and live user experiment approaches will be discussed in the following subsections.

3.1.1. Off-line evaluation

In an off-line evaluation, no actual users are involved and an existing data set (as described in the following section) is used. The data set is split in a test and a training set. Using the ratings in the training set the recommendation algorithm tries to predict the ratings in the test set, which can then be compared to the actual ratings in the test set.

In K-fold cross validation (a common cross validation technique), the data set is partitioned into K subsets. From these subsets, one is retained and used as the test set, the other subsets are used as training set. This process is repeated K times, each time with a different test set. Another commonly used approach is the leave-one-out cross-validation. This is in fact a special case of the K-fold cross validation, where K equals the number of users in the data set. Analogous to the leave-N-out approach the training set can have a specific size, like 2, 5, 10. This approach is referred to as "given 2", "given 5", et cetera.

The recommendation algorithm is evaluated by comparing the predicted and the actual ratings in the test set. The results can be analysed using several metrics, as discussed in sections 3.2–3.8.

Off-line evaluations have the advantage that they are quick, economical and easy to conduct on a large amount of data, several data sets and with multiple algorithms. When the data set includes timestamps indicating when a certain item was rated, it is even possible to repeat all the recommender system's interactions. Off-line evaluation, however, only allows the evaluation of predictions for items that were actually rated by the user, so sparsity limits the set of items that can be evaluated. Moreover off-line metrics cannot measure "true" user satisfaction.

3.1.2. On-line evaluation

In an on-line evaluation users interact with a running recommender system and actually receive recommendation. Feedback from the users is then collected by asking or observing them. Such a live user experiment may be controlled (*e.g.*, randomly assign users to different conditions) or it may be a field study in which a recommender system is deployed in real life and the effects of the system are observed. Conducting such evaluations is time consuming and difficult, but for this research it is inevitable to cunduct an on-line evaluation since it is the only way to measure true user satisfaction.

	Ranking		**Rating**	
Item	**User**	**RS**	**User**	**RS**
A	1	1	5	5
B	2	5	4	3
D	3	4	4	4
G	4	6	4	2
E	5	3	3	5
C	6	2	2	5
F	7	7	2	2

Table 3.1.: Rating example

3.2. Accuracy metrics

From the countless number of dimensions you could measure, accuracy is by far the most adopted [HKTR04, AT05, SPUP02, SPUP05, MH04, ZMKL05, CKP07, BHL07, ZK07, CdVP$^+$07]. An accuracy metric empirically measures to what extent a ranking, as predicted by a recommender system, differs from the actual ranking provided by the user. Accuracy metrics can

measure how well a user's ratings can be reproduced by the recommender system, but also how well a user's ranked list is predicted. To illustrate this subtle difference an example is shown in Table 3.1. On the left the recommender system (RS column) tried to reproduce the true user ranking (User column), while on the right it tried to reproduce the user's rating for each of the items A to F.

Herlocker et al. identified three classes of accuracy metrics: predictive accuracy metrics, classification accuracy metrics and rank accuracy metrics. The following subsections will describe each of these classes and present metrics that are most commonly used.

3.2.1. Predictive Accuracy Metrics

Predictive accuracy metrics measure to what extent a recommender system can predict ratings of users. These metrics are especially useful for systems that display the predicted ratings to their users. Since rated items have an order, predictive accuracy metrics can also be used to measure a system's ability to rank items.

A widely used predictive accuracy metric is the mean absolute error, or MAE. Mean absolute error takes the sum of the difference between the user's rating and the predicted rating and divides it by the number of items considered (as shown in Equation 3.1).

$$MAE = \frac{1}{|B_i|} \sum_{b_k \in B_i} |r_i(b_k) - p_i(b_k)| \tag{3.1}$$

With B_i denoting the items rated by user a_i. According to Equation 3.1 the recommender system's predictions as shown in Table 3.1 have a mean absolute error of $(|0| + |1| + |3| + |0| + |-2| + |0| + |2|)/7 = 1.143$.

Many variations of mean absolute error exist, such as mean squared error (MSE), root mean squared error (RMSE) and normalised MAE (NMAE). The mean squared error measure emphasises large errors by squaring each individual error, as shown in Equation 3.2. RMSE is simply defined as the root of MSE.

$$MSE = \frac{1}{|B_i|} \sum_{b_k \in B_i} (r_i(b_k) - p_i(b_k))^2 \tag{3.2}$$

In the running example this would result in a mean squared error of $(0^2 + 1^2 + 3^2 + 0^2 + (-2)^2 + 0^2 + 2^2)/7 = 2.571$. The RMSE would be $\sqrt{2.571} = 1.604$.

Normalised MAE [GRGP01] is mean absolute error, multiplied by a normalisation factor σ to normalise the value with respect to the range of rating values. This normalisation is done in order to allow inter data set comparisons.

$$NMAE = \sigma MAE = \frac{1}{r_{max} - r_{min}} MAE \tag{3.3}$$

Herlocker et al. found that mean absolute error is less appropriate when the granularity of true preference is small. Errors in the recommender system's predicted ratings only affect

the user when it results in erroneously classifying an interesting item as not interesting or vice versa. Carenini [Car05] elaborated further on this and identified that MAE is missing a key aspect of the recommendation process. Caranini states that a RS user wants to decide on whether an item is interesting or not, and therefore the value of a recommender system depends on how it will impact users in making the right choice. Consider for example a user a who will only watch movies rated 4.0 or higher (on a scale from 1-5). When the RS makes the following predictions:

1. $p_a(b_1) = 1$, while $r_a(b_1) = 3$
2. $p_a(b_2) = 3$, while $r_a(b_2) = 4$

The first prediction has an absolute error of 2, but won't affect the user since he is not interested in item 1 anyway. The second item, however, is of interest to the user, but because of the erroneous prediction (absolute error of 1) it will not be considered as interesting.

Carenini presented a user-specific accuracy metric that evaluates a RS's ability to help users selecting interesting items. This metric, called mean user gain, assumes a user-specific threshold θ_i in order to calculate the quality of a recommendation:

$$UG(p_i(b_k)) = \begin{cases} r_i(b_k) - \theta_i & \text{if } p_i(b_k) \geq \theta_i \\ \theta_i - r_i(b_k) & \text{otherwise} \end{cases}$$

This user gain function is then used to determine the mean user gain in a similar way as the mean absolute error is calculated (see Equation 3.4).

$$MUG = \frac{1}{|B_i|} \sum_{b_k \in B_i} UG(p_i(b_k)) \tag{3.4}$$

Assume for example that the user's threshold $\theta_i = 3$. Then the main user gain would be $(2 + 1 + -1 + 1 + 0 + 1 + -1)/7 = 0,429$. Note that, as opposed to the metrics presented above, a higher value means a better performance.

The mean user gain is tending towards the class of accuracy metrics presented in subsection 3.2.2, since it distinguishes two possible situations: the user is interested in the item or not.

3.2.2. Classification Accuracy Metrics

The second class of accuracy metrics consists of metrics that measure to what extent a RS is able to correctly classify items as interesting or not. The defect's magnitude with respect to the user's true rating and the rating predicted by the RS is ignored in these metrics.

Precision and recall are two dominant measures for evaluation from the Information Retrieval field, but they are popular in RS evaluation as well. Precision is defined as the ratio of relevant items selected to the number of items selected. In other words, precision is a measure for the efficiency of relevant item selection. Precision is defined in Equation 3.5, where B_{rs} is the set of selected items that are relevant, and B_s the set of selected items.

$$Precision = \frac{|B_{rs}|}{|B_s|} \tag{3.5}$$

Recall, shown in Equation 3.6, is defined as the ratio between the number of relevant items selected and the total number of relevant items in B, the set of all items. In other words, recall measures the breadth of the result.

$$Recall = \frac{|B_{rs}|}{|B_r|} \tag{3.6}$$

Precision and recall require the ratings to be mapped onto a binary scale (relevant vs. not relevant). This can be achieved by considering items rated 1-3 as non relevant and those rated 4-5 as relevant for example. In the running example the set B_r of relevant items would be $\{A, B, D, G\}$. Suppose a RS would present the set $B_s = \{A, C, E\}$ to the user, the precision of this recommendation would be $|\{A, B, D, G\} \cap \{A, C, E\}|/|\{A, C, E\}| = 1/3$ and the recall $|\{A, B, D, G\} \cap \{A, C, E\}|/|\{A, B, D, G\}| = 1/4$.

Both precision and recall rely on the notion of relevance. Relevance, however, is determined by a user's taste at a certain moment in time, which makes it impossible to depict one fixed threshold that classifies an item for a certain user as relevant or not (hence the mean user gain metric, presented earlier). Furthermore a RS recommends items based on the likelihood that a user will be interested in them, but ultimately it is the user who decides whether an item was indeed relevant.

Although precision and recall are two different measures, they cannot be considered apart from each other. It has been observed that precision and recall are inversely related. Furthermore the number of recommendations returned by the RS relates to precision and recall. Generally, when the number of recommended items increases, so does the recall, while precision decreases. There exist several methods that take both precision and recall into account and produce a single metric. Probably the most widely used one is the F-measure, as shown in Equation 3.7, with $\alpha = 1$.

$$F_\alpha = (1 + \alpha)\frac{Precision \times Recall}{\alpha \times Precision + Recall} \tag{3.7}$$

Other approaches that are often taken to relate precision and recall are to plot the average precision against different levels of recall or to compute the mean average precision (MAP). With mean average precision the average precision at the rank of each relevant document is calculated as shown in Equation 3.8.

$$AP = \frac{1}{|B_{rs}|} \sum_{b \in B_{rs}} Precision(b) \tag{3.8}$$

Two quite different classification accuracy metrics are the receiver operator characteristic (ROC), and the related customer ROC curves [SPUP05, HKBR99, SPUP02]. A ROC curve shows the hit/miss rates for different values of a classification threshold, but instead of varying

a threshold one can also vary the top n (a, b) tuples of an ordered recommendation list. The hit and miss rates are defined as follows:

$$\text{hit rate} = \frac{|B_{tp}|}{|B_{tp}| + |B_{fn}|}$$

$$\text{miss rate} = \frac{|B_{fp}|}{|B_{fp}| + |B_{tn}|}$$

Here B_{tp} (which equals B_{rs}), B_{fp}, B_{tn} and B_{fn} are the sets of true positive, false positive, true negative and false negative items respectively. ROC curves are constructed as described by Schein et. al. [SPUP05, SPUP02].

1. Construct an descending list of tuples $(a_i, b_j)_k$ according to the "$\geq$" relation on the predicted ratings $p_i(b_j)$.
2. For each $0 \leq n \leq k$, calculate the hit/miss rate caused by predicting the top n $(a_i, b_j)_k$ by magnitude and plot this point.

In practice n will be incremented by some fixed amount in order to approximate the curve. The ROC curve for the toy example shown in Table 3.1 would be calculated as follows:

Step 1: All predictions in Table 3.1 are made for the same user, say a_1. Ordering the tuples (a_1, b_j) in descending order with respect to $p_1(b_j)$ results in the list $[(1, A), (1, C), (1, E), (1, D), (1, B), (1, F), (1$ of length $k = 7$.

Step 2: Assume a step size $\Delta k = 1$. Starting with $n = 0$ will result in $(0, 0)$. $n = 1$ will result in $(1, 0)$, $n = 2$ in $(1, 1)$, $n = 3$ in $(1, 2)$, ..., $n = k = 7$ in $(4, 3)$. These co-ordinates are then plotted in a graph, as shown in Figure 3.1.

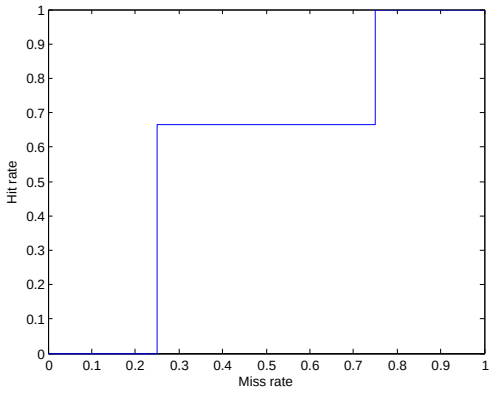

Figure 3.1.: An example ROC curve

One convenient property of ROC curves is that random recommendations can be represented as a straight line from the origin to the upper-right corner of the graph, since at each time half of the recommendations is a hit and the other half is a miss. A perfect recommender will result in a curve that goes from the origin straight to the upper-left corner and then straight to the upper right one, since first all "true" relevant items are identified as such (100% hit rate) and then all "true" irrelevant ones.

Another advantage of ROC curves is that they can be summarised by a single metric representing the area under the curve. This makes it easier to compare different recommender systems. A random recommender would have an expected area of 0.5, while a perfect recommender has area 1.0.

Schein et al. [SPUP05] criticised ROC curves since they reward recommender systems that skew the number of recommendations to the users who rate most highly or are most active. This can boost the performance and may lead to a misleading sense of performance. Therefore Schein et al. proposed the Customer ROC (CROC) metric as a complement to ROC. CROC curves measure to what extent a RS can provide good recommendations across a wide user base. The curve is constructed similar to ROC curves, but requires the number of recommendations to be equal to each user (and thus normalises the number of data point per user).

1. For each person a_i, order the predicted ratings $p_i(b_k)$ in a list by magnitude imposing an ordering: (b_k).

2. For each $0 \leq n \leq k$, calculate hit/miss rates caused by recommending the top predicted $\min(n, |B_{a_i}|)$ items to each user a_i from their own list and plot the hit rate against the miss rate.

Here B_{a_i} is the set of items in the test set recommended to user a_i.

For CROC curves, the two properties described above do not hold. The curve of a perfect recommender can have an area under curve that is less than 1.0 for example. This is the case when the test set contains three people each with six observations (so $|B_{a_i}| = 6$ for $i = 1, 2, 3$): user a_1 likes four out of six items, user a_2 likes two out of six items, and user a_3 likes all six items. When four items are recommended to each user, two false positives are encountered at user a_2 and thus the area under the curve decreases. In order to enable comparison, the curve for the perfect recommender, given the test set, is also plotted in the graph as shown in Figure 3.2.

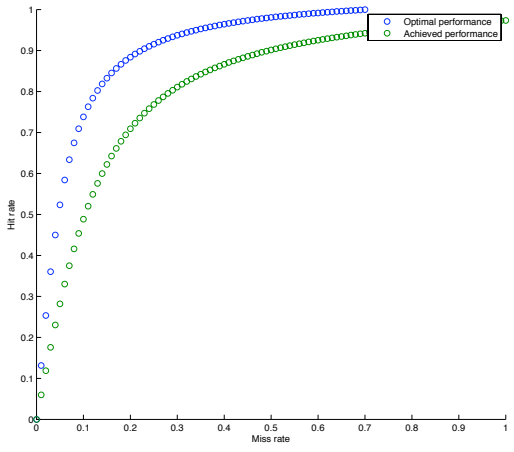

Figure 3.2.: Example of a CROC curve

3.2.3. Rank Accuracy Metrics

The third class of accuracy metrics identified by Herlocker et al. is the class of rank accuracy metrics. These metrics evaluate a recommender system's ability to recommend an ordered list of items to a user, assuming that the order of the items on the list is important. Suppose for example that a recommender system recommends the items A, B and D in this particular order to the user from the running example. Although the user finds all of these items relevant, he might prefer D over B and therefore would order the items (in order of preference) as A, D, B. Rank accuracy metrics take this into account and penalise the recommender for not producing the right order of items.

There are several correlation measures that can be used to determine the similarity of two lists of recommendations. The most widely used are Spearman's correlation coefficient and Kendall's Tau correlation. Spearman's ρ correlation is a special case of the widely used Pearson product-moment correlation coefficient (as described in section 2.2). The main difference is that the data is converted to rankings before calculating the coefficient [HKTR04]. Equation 3.9 shows Spearman's rank correlation coefficient.

$$\rho = \frac{\sum(x - \overline{x})(y - \overline{y})}{n \cdot stdev(x) \cdot stdev(y)} \tag{3.9}$$

Here $x_1, ..., x_n \in \mathbf{x}$ and $y_1, ..., y_n \in \mathbf{y}$ are the ranks of the items listed in the user's ranked list and the recommendation list respectively. Table 3.2 shows an example of the data point (right) used in the calculation of Spearman's rank correlation coefficient for the ranked lists shown on the left. The corresponding $\rho = 0.5$.

Rank	**User**	**RS**
1	A	A
2	G	B
3	B	E
4	E	F
5	C	C
6	D	G
7	F	D

Item	**Spearman**
A	(1,1)
B	(3,2)
C	(5,5)
D	(6,7)
E	(4,3)
F	(7,4)
G	(2,6)

Table 3.2.: Ranking example

Kendall's Tau correlation takes a quite different approach to determine the correlation between two ranked lists. This approach uses the number of concordant and discordant pairs-pairs. A pair of tuples (x_1, y_1) and (x_2, y_2) is concordant when $\text{sgn}(x_2 - x_1) = \text{sgn}(y_2 - y_1)$ and discordant when $\text{sgn}(x_2 - x_1) = -\text{sgn}(y_2 - y_1)$, where x_i and y_i are the ranks for item a_i as ranked by the user and predicted by the RS. The sgn function is defined as follows:

$$\text{sgn}(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & 1 > 0 \end{cases}$$

The number of concordant pairs C and discordant pairs D are used by Herlocker et al. to calculate τ as shown in Equation 3.10. TR and TP stand for the number of pairs of items that have the same ranking in the true ordering and predicted ordering respectively.

$$\tau = \frac{C - D}{\sqrt{(C + D + TR)(C + D + TP)}} \tag{3.10}$$

When the example recommendation list shown in Table 3.2 is evaluated with Kendall's Tau correlation method, C, D, TR and TP must first be determined. Table 3.3 shows the result of a pairwise comparison of the ranking tuples. Here + means a concordant pair and − a discordant pair. This results in $C = 13$, $D = 8$, $TR = 0$ and $TD = 0$ and therefore $\tau = 0,168$.

	A	B	C	D	E	F	G
A							
B	+						
C	+	+					
D	+	+	+				
E	+	+	+	+			
F	+	-	-	-	+		
G	+	-	-	-	-	-	

Table 3.3.: Kendall's Tau correlation example

Another rank accuracy metric presented in the literature is the Half-life Utility Metric, presented by Breese et al. [BHK98]. This metric considers the fact that users probably won't browse very deeply into a ranked list and therefore a recommended item becomes less significant when it's deeper in the list. The half-life utility metric estimates the expected utility of a certain ranked list to a user. The expected utility of a list $B_{a_i} = b_1, ..., b_k$, sorted by index k in order of declining $p_i(b_k)$, for a user a_i is defined as shown in Equation 3.11. In this equation d is the default or neutral vote and α is a natural number denoting the rank of an item on the list that has a 50% change of being viewed.

$$R_{a_i} = \sum_k \frac{\max(r_i(b_k) - d, 0)}{2^{(k-1)/(\alpha-1)}} \tag{3.11}$$

The overall Half-life Utility score R, over a test set is calculated as shown in Equation 3.12.

$$R = 100 \frac{\sum_{a_i} R_{a_i}}{\sum_{a_i} R_{a_i}^{\max}} \tag{3.12}$$

Here $R_{a_i}^{\max}$ is the maximum achievable utility for user a_i (*i.e.* the RS ranked the items exactly as the user did). For the running example presented in Table 3.1 the calculation is shown in Table 3.4, where $d = 3$ and $\alpha = 3$.

R_{a_1} becomes $2,729$ and since $R_{a_1}^{\max} = 3,561$, R becomes $76,631$.

k	b_k	$r_i(b_k)$	$r_i(b_k) - d$
1	A	5	2
2	E	3	0
3	C	2	0
4	D	4	1
5	B	4	1
6	F	2	0
7	G	4	1

Table 3.4.: Half-life Utility Metric example

The normalised distance-based performance measure (NDPM), originally proposed by Yao [Yao95], can be used to compare two weakly ordered lists. In this comparison the number of contradictory preference relations C^- and compatible preference relations C^u are used, as well as the number of preferred relationships in a user's ranking. A contradictory preference relation between two items b_k and b_l happens when the RS rates b_k higher then b_l, but the user did it the other way around. A compatible preference relation happens when the RS rates b_k higher then b_l, but the user rated b_k and b_l equally. A relationship between two items b_k and b_l is called a preferred relationship when the user did not rate the items equally. The *NDPM* metric is then defined as shown in Equation 3.13.

$$NDPM = \frac{2C^- + C^{\cdot\cdot}}{2C^i} \tag{3.13}$$

Table 3.5 illustrates the computation of the *NDPM* measure for the two ranked lists shown in Table 3.2. The table shows contradictory preference relations for items A to G. Items C and F for example, have a contradictory preference relations since the user prefers C over F, while the RS predicted it the other way around. For the example holds $C^- = 12$, $C^u = 0$ and $C^i = 42$, so $NDPM = 6/21$

	A	B	C	D	E	F	G
A							
B							!
C						!	!
D						!	
E							!
F			!	!			!
G		!	!		!	!	

Table 3.5.: NDPM example: conflicting preference relations

An alternative way to evaluate a recommender system's ability to recommend an ordered list of items to a user is inspired by the edit distance as used in information retrieval. Originally edit distance is defined as the number of operations needed to transform one string into another. Considering the strings "distance" and "distanse" the edit distance would be 1 since the "c" needs to be substituted by a "s" in order to transform the first string into the second

one. Other possible operations are the insertion and deletion of a character. There are various algorithms to calculate the edit distance between two strings such as the Hamming distance [Ham50], Levenshtein distance [Lev66], Damerau-Levenshtein distance [Dam64, Lev66] and the Jaro-Winkler distance [Jar89].

The list of recommended items and the user's ranking of this list can be considered to be strings over an alphabet consisting of N elements, where N is the number of items. The edit distance between these strings can then be computed analogous to strings over the Latin alphabet. Consider the running example presented in Table 3.2. The edit distance between the lists "AGBECDF" and "ABEFCGD" is 4, so it requires 4 operations to turn one list in the other.

The relative edit distance is defined as the edit distance between two lists $[b_0, ..., b_m]$ and $[b'_0, ..., b'_n]$ divided by the sum of the length of each list (*i.e.* $m+n$) as shown in Equation 3.14.

$$RED = \frac{ED}{m+n} \tag{3.14}$$

The metric can be implemented using one of the edit distance algorithms mentioned above. The pseudocode shown below shows the implementation of the relative edit distance metric using Levenshtein distance [Lev66].

```
public double relEditDistance(int[] userList, int[] rsList) {

    return LevenshteinDistance(userList, rsList) / (userList.length +
       rsList.length);

}
```

Listing 3.1: Relative edit distance algorithm

```
public int LevenshteinDistance(int[] userList, int[] rsList) {

    int d[0..m, 0..n]

    for(int i = 0; i <= m; i++) {
        d[i, 0] = i;
    }
    for(int j = 0; i <= n; j++) {
        d[0, j] = j;
    }

    for(int i = 0; i <= m; i++) {
        for(int j = 0; j <= n; j++) {
            if (userList[i] == rsList[j]) {
                cost = 0;
            } else {
                cost = 1;
            }
            d[i, j] = min(
                d[i-1, j] + 1,      // deletion
                d[i, j-1] + 1,      // insertion
```

```
                d[i-1, j-1] + cost // substitution
            );
    return d[m, n];
}
```

Listing 3.2: LevenshteinDistance algorithm

3.3. Coverage

Besides accuracy there are a number of other dimensions that can be measured [ZMKL05, HKTR04]. One of these dimensions that is mentioned in the literature is coverage. Coverage measures the percentage of items for which the recommender system can make predictions or recommendations. A recommender system cannot always generate a prediction since there might be insufficient data. When an item has never been rated before an item-based prediction technique cannot predict a particular user's rating for that item for example.

There are two types of coverage identified by Herlocker et al. [HKTR04], called prediction coverage and catalogue coverage. Prediction coverage is a measure for the percentage of items that the recommender system can form predictions for. Catalog coverage on the other side is a measure for the percentage of items that is ever recommended to any user. A higher coverage means that the RS is capable to support decision making in more situations. Coverage can be measured by taking a random sample of user/item pairs from the data set and then ask the RS to provide recommendations for all of them. Both predictive and catalog coverage can then be estimated.

Like precision and recall cannot be taken on their own, coverage cannot be considered independently from accuracy. A recommender system can possibly achieve high coverage by making spurious predictions, but this has its repercussion on accuracy.

According to Herlocker et al. [HKTR04] it might be more useful to compute coverage only over the items that a user is actually interested in. When a RS cannot form a recommendation for an item that the user is not interested in anyway then it is considered not much of a problem.

3.4. Confidence

Recommendations made by a RS can be characterised along two dimensions: strength and confidence. Strength indicates how much the user will like the recommended item. Confidence indicates how sure the RS is about the accuracy of the recommendation. Both dimensions are often conflated, resulting in the assumption that the higher an item's predicted rating, the better a user will like it. Extremely high predictions, however, are often based on a small amount of user ratings (*i.e.* high strength, low confidence). As the number of ratings grows the prediction will usually regress to the mean (*i.e.* low strength, high confidence).

Recommender systems have different approaches to deal with confidence. Most systems do not display items with a confidence level that is below a certain threshold. Another approach is to display the confidence of a recommendation to the user. Herlocker et al. investigated the latter approach and found out that good confidence displays achieved significant improvement in decision making. However, many recommender systems do not include some notion of confidence.

Although measuring confidence is complicated since there are many dimensions that affect it, McNee et al. [MLG+03] suggested a proposal. They investigated the strength of the user's neighbourhood (in a user-item based collaborative filtering RS), item similarity (in an item-item based CF system) and the number of ratings the system has for a user or item. The latter is based on the assumption that the system knows the user or item better when it has more information about it. McNee et al. showed that in their particular case accuracy improved when predictions were based on a larger number of ratings.

3.5. Diversity

Diversity of the items in a recommendation list is another aspect that might be important to users and therefore must be measured [McN06]. A user who bought a book of "The Lord of the Rings" on Amazon for example, might receive recommendations for all other books of Tolkien. Although the user probably likes all of these books (which results in a high accuracy) he might not be completely satisfied. According to McNee [McN06] difference in list diversity will have a large effect on the usefulness of recommendation lists and therefore McNee, Ziegler and others [McN06, ZMKL05] presented the intra-list similarity metric.

The intra-list similarity metric, shown in Equation 3.15 is a measure for the diversity of a recommendation list L_{p_i} for a certain user a_i, based upon predictions p_i (see section 4.7 for more information).

$$ILS(L_{a_i}) = \frac{\sum_{x \in L_{a_i}} \sum_{y \in L_{a_i}, x \neq y} sim(x, y)}{|L_{a_i}| \cdot (|L_{a_i}| - 1)} \tag{3.15}$$

Here $sim(x, y) : B \times B \rightarrow [-1, 1]$ is a function that maps two items onto a value from -1 to 1, indicating their similarity with respect to a certain criterion (like genre, author, etc.) or combination of criteria.

3.6. Learning rate

Many recommender systems incorporate learning algorithms that gradually become better in recommending items. Collaborative filtering algorithms are likely to perform better when more rating information is available for example. Content based filtering algorithms that learn what attributes are important to a specific user also need feedback in order to learn each attribute's weight. Learning rate is a measure for how quickly an algorithm can produce "good" recommendations. The metric for quality is usually accuracy. Some recommendation algorithms only need a small number of ratings to produce sufficiently good recommendations while others might need extensive training before they reach the same level of quality. Learning rates can be described by some asymptotic (and thus non-linear) function since the quality of the recommendations cannot increase indefinitely. An example of a graph in which the MAE is plotted against the number of ratings provided by the users is shown in Figure 3.3.

Herlocker et al. identified three different types of learning rates: overall learning rate, per item learning rate and per user learning rate. The overall learning rate is defined as cumulative quality over the number of ratings provided by all users for all items, while per item and per

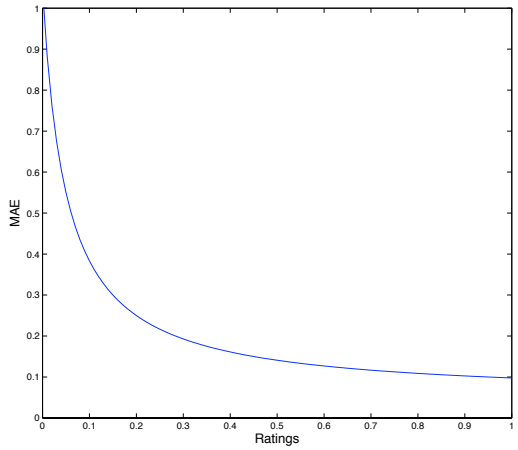

Figure 3.3.: Example of learning rate graph in which the MAE is plotted against the number of ratings

user learning rate only tak into account the ratings provided for a single item or by a specific user respectively.

Learning rate is a strong indicator for a recommender system's capability to cope with the cold start problem. But despite the fact that the cold start problem is widely recognised by researchers the evaluation of a recommender system's learning rate has not been extensively covered in the literature and no specific metrics exist.

3.7. Novelty and serendipity

A recommender system can produce highly accurate recommendations, have a reasonable good coverage and still does not satisfy a user. Suppose for example that an EPG with recommendation functionality recommends the news to every user. Although this probably results in high accuracy it has no added value to the user, since he already knew the news and would probably have viewed in anyway [TH01]. However, a recommendation that is obvious to one user, might be new for another. An expert on recommender system evaluation probably finds a recommendation of Herlocker's "Evaluating collaborative filtering recommender systems" obvious, while a user who is new to this field is dying to find such a paper.

Novelty and serendipity are two (closely related) dimensions for non-obviousness, as identified by Herlocker et al. [HKTR04]. Serendipity is the experience of discovering an unexpected and fortuitous item. This definition contains a notion of unexpectedness, which is the novelty dimension. Novelty and serendipity metrics thus measure the non-obviousness of recommendations and penalise "cherry-picking". The literature does not describe a metric that can be used to measure novelty, nor serendipity.

3.8. User satisfaction

User satisfaction is a somewhat vague and soft aspect and therefore it is difficult to measure. In the context of this research user satisfaction is defined as the extent in which a user is supported in coping with the information overload problem. The dimensions described in the above sections will probably support and/or inhibit user satisfaction to some extent. In order to be able to determine the effect of these dimensions on user satisfaction, user satisfaction itself must also be measured.

Herlocker et al. presented a number of dimensions along which user satisfaction evaluation methods can be classified [HKTR04]. These dimensions are listed next.

- *Explicit vs. implicit.* The evaluation method can explicitly ask how satisfied the user is, or it can be observed in some way. Implicit evaluation methods require assumptions in order to translate the observations into a measure for user satisfaction, like an increase in sales implies greater user satisfaction for example.
- *Laboratory studies vs. field studies.* Lab studies are studies in a controlled environment while field studies take place in the user's own real context.
- *Outcome vs. process.* The study can only focus on the outcome but it can also focus on the process that originates to it.
- *Short term vs. long term.* Short term user evaluations might miss results that only become apparent after a certain amount of time. The evolvability of a user's taste is an example of a phenomenon that can only be investigated in a long term user evaluation.

Studies that investigated user satisfaction with respect to *recommender systems* are scarce and studies that focus on user satisfaction with respect to *recommendations* are even rarer. Many studies that focus on user satisfaction are primarily concerned with human computer interaction and do not focus on the recommendation list's properties.

3.9. Conclusion

This chapter focused on how the performance of recommender systems is currently evaluated as described in the literature. This evaluation is important in order to compare different recommender systems and to be able to improve recommender system performance. The first section described two different approaches to evaluate recommender systems: off-line and on-line evaluation. With off-line evaluations no user participation is needed, which makes these evaluations quick, economical and easy to conduct. When performing an on-line evaluation, users interact with a running recommender system and actually receive recommendations. Such an experiment can be quite importunate to the users and takes much longer to perform.

The remaining sections of this chapter explained a number of dimensions along which a recommender system can be evaluated. The most widely evaluated dimension is accuracy, which is a measure for a recommender system's ability to reproduce the user's ranking or ratings. Three different classes of accuracy metrics can be identified: predictive accuracy metrics, classification accuracy metrics and rank accuracy metrics. Numerous of these metrics are described in detail in section 3.2.

Even though accuracy is by far the most widely used dimension to evaluate recommender systems, recent research has shown that it is not equivalent to user satisfaction. A recommender system that is extremely accurate might not be satisfying its users. Sections 3.3 to 3.7 therefore described other dimensions that were identified in the literature and can be used to evaluate a recommender system. Among these dimensions are diversity, novelty and serendipity. Methods for measuring user satisfaction, which is assumed to be the ultimate dimension along which recommender systems should be evaluated, are finally described in section 3.8.

The remainder of this thesis will describe how the coherence between user satisfaction and the other metrics was measured and how user satisfaction can be predicted based on measurements for these dimensions.

4

Conducting the User Experiment

Since there exist no notable data sets that incorporate a measure for user satisfaction, a user experiment needs to be conducted in order to be able to compare user satisfaction to off-line metrics such as accuracy. This chapter describes the user experiment in detail and explains how the metrics that were described in Chapter 3 are applied in an EPG context.

In this chapter the following will be discussed

- The general approach to the user experiment (Section 4.1).
- The construction of a representative user base (Section 4.2).
- The gathering of TV listing information (Section 4.3).
- The training of the recommender (Section 4.4).
- The strategies used to generate recommendations (Section 4.5).
- The way the recommendation lists are rated by the users (Section 4.6).
- The dimensions measured and the metrics used (Section 4.7).

4.1. Approach

The user experiment must be conducted to answer the following research sub question as defined in section 1.3:

Research sub-question: How do the individual properties of a recommendation list and user satisfaction cohere?

In order to answer this research question a recommender system is actually implemented as part of an evaluation application. This application recommends TV programmes to its users

and gathers feedback provided by the users about the recommendations. The recommendation lists generated by the system are evaluated with respect to five different dimensions: accuracy, diversity, novelty, serendipity and programme overlap, so the correlation between these dimensions and user satisfaction can be calculated.

The evaluation is subdivided in two primary phases, the training phase and the recommendation phase. During the training phase a user is asked to rate the TV programmes he viewed yesterday evening (or the evening before, in case he was not in the occasion to rate before). The recommender system embedded in the evaluation application uses this feedback to train its prediction algorithms. Once the recommender system has sufficient information about a user he can proceed to phase two, the recommendation phase. In this phase the recommendation system starts to provide the user with recommendation lists containing ten TV programmes. These lists are generated using a prediction technique that is likely to provide a list with certain properties (*i.e.* high accuracy, low diversity, etc.). After the second phase provided enough data points, the experiment comes to an end for the user.

The following sections describe the steps to be taken in more detail.

4.2. User Base

A representative user base is important to a sound user experiment [BKH01]. Since the intended audience of an electronic programme guide's recommender system can by really broad, the user base for this experiment should be diverse as well. To achieve this the users solicited to take part in the research are not restricted to colleagues (same field of expertise), fellow students (same level of education) or friends. Instead everybody that wants to participate can register and take part in the experiment.

The size of the user base needed to reach a certain confidence level is really hard to determine. Equation 4.1 shows how the number of users n coheres to the confidence interval $(1 - \alpha)$.

$$P\{\bar{X}_n - k_r \frac{S_n}{\sqrt{n}} < \mu < \bar{X}_n - k_l \frac{S_n}{\sqrt{n}}\} = 1 - \alpha \tag{4.1}$$

Variables k_r and k_l depend on how $X_1, ..., X_n$ are distributed. Assuming a normal distribution k_r and k_l are $t_{n-1,\frac{1}{2}\alpha}$ and $-t_{n-1,\frac{1}{2}\alpha}$ respectively. However, it is problematic to estimate the variables $\bar{X}_n$ and S_n. Moreover it is hard to say how many targeted users will actually participate in the research and how active they will be. Not everybody watches that much TV for example.

To assure an as large user base as possible and to stimulate active participation, a personalised DVD Box is offered to the user that participates best in the experiment. Furthermore participating in the research is made as less intrusive and time consuming as possible and the evaluation application is made easily accessible.

Users that decide to take part in the user experiment have to provide some demographic information in order to complete their registration. The information includes gender, age, marital status, education level and household size. It is used to see whether the user base is homogeneous and can be used as input for the recommendation engine.

4.3. Gathering the Content

In order to make the evaluation application as realistic as possible it will recommend actual TV programmes. This prevents the users from getting the feeling of doing senseless work and makes it more enjoyable to participate. Moreover, some users might be using a (non recommending) EPG anyway and just switch to the evaluation application.

The TV programme data that will be used during the experiment is provided by AKN, a service providing organisation of three Dutch broadcasting companies: AVRO, KRO and NCRV. The data provided by AKN is accurate, complete and is electronically available through an XML feed. The XML feed contains metadata such as genre, actors, textual description, year of premiere, director and presenters.

4.4. Training the Recommender

Users that decided to take part in the research and subscribed themselves to the evaluation application receive an e-mail message on a daily basis. This e-mail message requests the user to visit the evaluation application (simply by clicking on a URL provided in the message) and to rate the TV programmes they watched the evening before. Users also have the possibility to provide feedback on programmes they have watched the day before yesterday and three days ago, since it is not inconceivable that they forget it sometimes. Moreover, people that provided their business' e-mail address might otherwise not be able to provide feedback on Friday's TV programmes. Of course users are free to rate other programmes as well. Adjusting a previously provided rating is also no problem. Users rate programmes on a five-point Likert scale [Lik32] by selecting one to five stars. The five-point Likert scale is used in most recommender systems and users are therefore likely to be familiar with rating items this way. Rating a TV programme with one star indicates the user does not like that particular programme, while rating it five stars means he really likes it. The meaning of the scale is explained to the users as it is explained in the previous sentence.

The ratings provided by users are used to train the different prediction algorithms used by the recommendation engine. The recommendation techniques that are applied are described in section 4.5.

In order to prevent the rating matrix from becoming really sparse and to reduce the time it takes to rate the programmes, the evaluation application only covers TV programmes broadcasted during prime time (19:00 - 00:00) on the mainstream channels in The Netherlands. During prime time broadcasters broadcast 5-6 programmes on average, resulting in a TV listing containing about 100 programmes. This number of programmes is comprehensible to the users and it will not take to much time to find and rate the programmes they have watched or want to rate.

The training phase for a specific user ends when the prediction algorithms have sufficient data to make recommendations for that user.

4.5. Generating the recommendations

Generating the list of recommendations for the users of the evaluation application is one of the most important activities in the user study. The properties of the recommendation lists

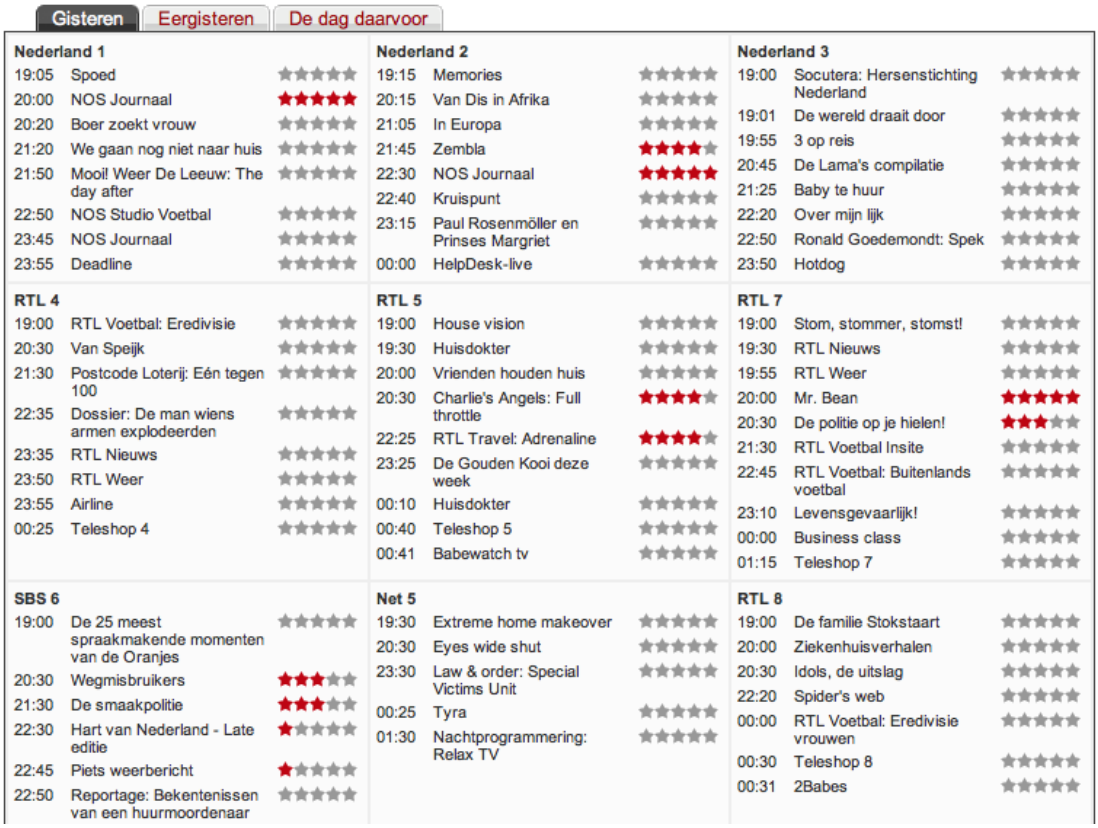

Figure 4.1.: Utente TV listing with stars to rate the programmes

must vary over the dimensions that will be presented in section 4.7. To assure an accurate distribution over each of the properties that are investigated, seven different recommendation strategies are applied. The following subsection will describe each of these recommendation strategies in more detail.

Each recommendation strategy generates the recommendations using a combination of a prediction strategy and a clipping strategy. The prediction strategy predicts a user's rating for a list of TV programmes. The clipping strategy turns this list of predictions into a list of ten recommendations. The clipping strategy can therefore heavily influence the properties of the list of recommended programmes.

4.5.1. Random recommendations

The random recommendation strategy is the baseline recommendation strategy in the experiment. It tries to give an accurate prediction of a user's rating for all TV programmes that are broadcasted that evening and then randomly selects 10 programmes that will be recommended to the user. Duplicates of programmes in the list are allowed and properties such as diversity, novelty and programme overlap are not influenced in any way. The predictions are made using series level collaborative filtering. This technique will be described next.

4.5.2. Series level Collaborative Filtering

This recommendation strategy combines a collaborative filtering technique on the series level with a simple top-N clipping strategy. The prediction strategy calculates for each user a_i the correlation with respect to each other user a_j. This leads to a $|A| \times |A|$ matrix containing all the correlations between users. Based on this correlation matrix a neighbourhood is created for each user a_i and TV programme b_k, consisting of the eight users that are most similar to a_i and rated at least one programme in B^c (*i.e.* the set of programmes belonging to the same TV show as b_k) in the past. Analogous to the aggregation functions described in subsection 2.2.1 the ratings of the user's neighbours are combined to predict the rating of user a_i for programme b_k. The aggregation method used in this strategy uses the weighted sum approach that takes into account the average rating of a user as defined in Equation 2.4c. A key difference however is that the series level CF strategy aggregates the average ratings of the neighbours for all episodes of a TV show instead of ratings for that specific episode. Thus, $r_j(b_k)$ is substituted by:

$$\hat{r}_j(b_k) = \frac{1}{|B_j|} \sum_{b_k \in B_j} r_j(b_k) \tag{4.2}$$

This approach is necessary since b_k itself is unlikely to be rated before by any user because it probably has never been broadcasted before. Therefore one has to deal with a continuous cold start problem. Since most users rate different episodes of the same TV show in a similar way this approach can be taken.

The top-N clipping strategy simply sorts the programmes in descending order of predicted rating and returns the top 10 predicted programmes. This top 10 list is likely to contain different episodes of the same TV show since different TV channels broadcast the same show or a programme is repeated later the same evening.

4.5.3. Unique series level Collaborative Filtering

The unique series level collaborative filtering strategy uses the same prediction strategy as the series level CF strategy, but instead of simply recommending the top 10 predicted programmes it uses a different clipping strategy. This clipping strategy returns the top 10 predicted programmes after removing all duplicates. Two programmes are considered duplicates if they are episodes of the same TV show. This strategy strives to be accurate, as well as diverse.

4.5.4. Demographics based Collaborative Filtering

The third recommendation strategy is a variation on collaborative filtering. Instead of creating a neighbourhood based on similar rating behaviour of users it uses similarity of user demographics. Demographic information that is used by this prediction strategy includes age, gender, marital status, level of education, household size and the industry in which the user is working/studying. For every two users the similarity of each of these aspects is derived. Then the correlation between the users is calculated. Again these correlations form a correlation matrix that is the basis for the creation of neighbourhoods. These neighbourhoods are

created analogous to those of the series level collaborative filtering strategy. The aggregators are also similar.

The demographics based collaborative filtering strategy uses a variation of the clipping strategy used in the unique series level CF strategy. Instead of removing all duplicates this strategy randomly removes some duplicates, so the recommendation list might contain several programmes from the same TV show.

4.5.5. User's inverse rating

The goal of the user's inverse rating strategy is to recommend TV programmes that the user does not like at all. The prediction strategy is used to achieve this, since it takes the user's average rating for the TV show from which the programme is part and predicts the inverse of it. The ratings are inverted by mapping them to the domain [-1, 1], multiplying them by -1 and then converting them back to the domain [1, 5]. After the user's ratings are inverted, the top-N clipping strategy is applied to the predictions.

The user's inverse rating strategy is likely to produce highly inaccurate recommendations and that is exactly its purpose.

4.5.6. Novel unique series level Collaborative Filtering

The novel unique series level CF strategy is another variation of the series level CF strategy. It uses series level collaborative filtering to predict a user's rating for a TV programme as described in subsection 4.5.2. The clipping strategy then removes all programmes from the list that are not novel to the user (*i.e.* all programmes that are part of a TV show for which the user have never rated a specific episode) and returns the top 10 predicted programmes from the remainder of the list.

The novel unique series level collaborative filtering strategy tends to produce recommendations that are novel to the user, sacrificing some accuracy.

4.5.7. Unique known series level Collaborative Filtering

This series level collaborative filtering strategy is the opposite of the novel unique series level CF strategy. Instead of recommending programmes that are novel to a user it only recommends episodes of series that the user is familiar with. Therefore the novelty of the recommendation list is likely to be really low and its accuracy very high.

4.6. Rating the recommendations

Once the recommender system has sufficient information about a user (see the previous section) the recommendation system starts to provide the user with recommendations. The system shows a list of ten recommendations for TV programmes that are broadcasted that evening. The user is asked to provide feedback on how satisfied he is with the list of recommendations, as well as which programmes are novel to him. Furthermore the user can provide feedback on how diverse and how surprising he thinks the list is. Lateron this feedback can be used to see whether any of the metrics described in Chapter 3 correspond to the user perception of these two aspects. Feedback on the recommendation list is also provided on a

five-point Likert scale. Figure 4.2 shows the user interface for displaying the recommendations and acquiring feedback.

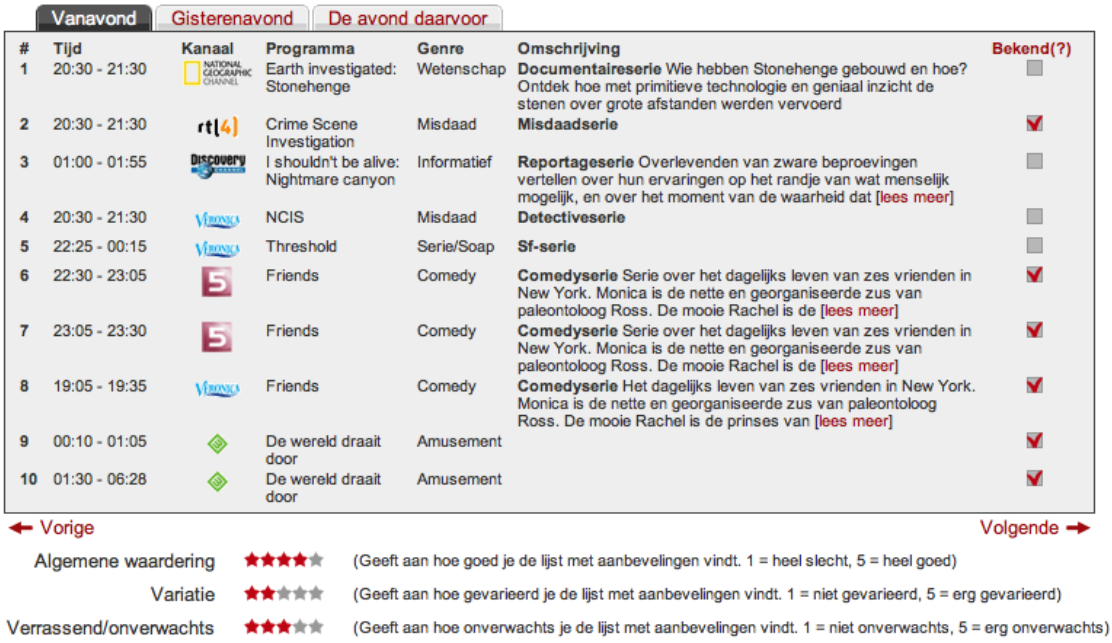

Figure 4.2.: Utente recommendation list

The recommendation system build into the evaluation application utilises several prediction algorithms, each generating recommendations with certain properties (*i.e.* high accuracy, low diversity, etc.). All of these predictions are stored in a database and when a user logs in, a prediction strategy is assigned to him in a round robin fashion. Suppose for example that the systems utilises three different recommendation strategies s_1, s_2 and s_3 and that the following 5 users log in in the sequence: a_4, a_{10}, a_2, a_{12}, a_8. Then user a_4 receives a recommendation list generated using prediction strategy s_1, user a_{10} gets s_2, a_2 get s_3, a_{12} gets s_1 again, and so on. The dimensions that are varied by using different prediction strategies are described in the following section.

During the recommendation phase of the user experiment, users are still required to rate individual TV programmes, analogous to the training phase described earlier. These ratings enable the derivation of the exact accuracy of a recommendation list for example.

4.7. Dimensions and Metrics

During the recommendation phase the recommendation engine recommends list of TV programmes along different dimensions (see section 4.6). A list can be very diverse but fairly inaccurate for example. The exact properties of each list cannot be controlled by the recommendation engine, since they differ per item and per user. This implies that the recommendation engine cannot tell in advance whether the list of recommended TV programmes that is displayed to a user will be accurate for example. It is possible, however, to determine the exact properties of a recommendation list afterwards, using the user's feedback. Assume for

example that the evaluation application displays a list of recommendations L_{p_i} to user a_i that was generated using recommendation strategy s_n, which on average makes highly accurate predictions. Once the user has rated the programmes that were recommended to him it is possible to calculate the exact accuracy of the list L_{p_i}, as well as its diversity, novelty, et cetera.

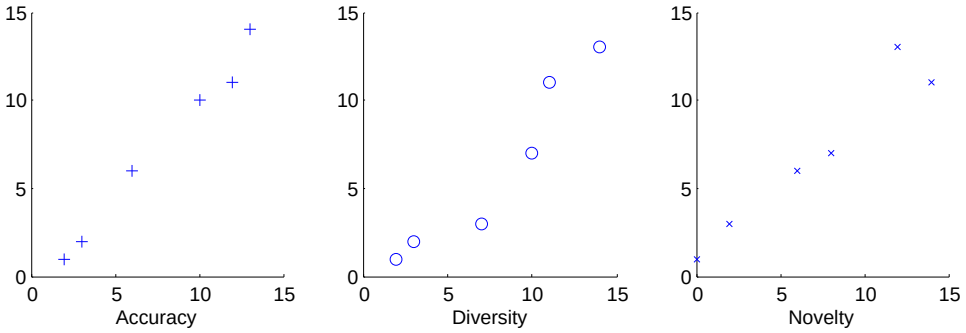

Figure 4.3.: Example scatter plots

When these values are plotted against the measured user satisfaction this results in a scatter plot as shown in Figure 4.3 for example. Note that each recommendation list results in exactly one data-point in *each* plot.

It is important to make a distinction between properties of a single recommendation, a list of recommendations and the prediction strategy that generated the recommendations. Accuracy, for example, is a property of a single recommendation [MRK06], while diversity is a property of a recommendation list [MRK06, ZMKL05]. Moreover, learning rate is a property of a recommendation strategy. Since the user satisfaction with respect to a recommendation list is investigated some of the properties must be aggregated to form a recommendation list property. This aggregation can be done simply by taking the average over all recommendations for example. Suppose that a recommendation list consisting of two recommendations makes an error of 2 for the first recommendation and 1 for the second. The average error of the list thus is 1.5. In fact, this is analogous to the mean absolute error metric.

This research is focusing on five different dimensions: accuracy, diversity, novelty, serendipity and programme overlap. All of these dimensions, except for programme overlap were identified by Herlocker et al. [HKTR04] as potentially influential with respect to user satisfaction. Programme overlap is a list property that is specific to recommending TV programmes since they usually are broadcasted on a specific time (at least now that is the case). The following subsections will discuss each of these dimensions in more detail and will present the metrics used in the research to measure them.

4.7.1. Accuracy

From the countless number of dimensions you could measure, accuracy is by far the most adopted in research to date [HKTR04, AT05, SPUP02, SPUP05, MH04, ZMKL05, CKP07, BHL07, ZK07, CdVP$^+$07]. Accuracy is believed to be a very important aspect of user satisfaction [ZMKL05] and the metric of choice for many researchers since it is widely adapted

and easy to measure [CdVP+07]. Accuracy is therefore measured in this research as well.

Herlocker et al. [HKTR04] identified three different classes of accuracy metrics: predictive accuracy metrics, classification metrics and rank accuracy metrics. These metrics were explained in more detail in Section 3.2. Herlocker et al. investigated the most popular accuracy metrics and empirically showed that these metrics can be clustered into three classes (which are different from the classes mentioned above). This research incorporates five different metrics:

- Mean Absolute Error
- Area under ROC curve
- Precision and recall
- Spearman's correlation
- Relative Edit Distance

The first three metrics cover the entire spectrum, as shown in Table 4.1. The fourth metric is a flash of intuition that might be interesting to investigate as well.

	Predictive	**Classification**	**Rank**
C1			Spearman, RED
C2		ROC, precision	
C3	MAE		

Table 4.1.: Accuracy metrics spectrum

In the context of this research mean absolute error of a list of recommendations is defined as shown in Equation 4.3.

$$MAE = \frac{1}{|L_{p_i}|} \sum_{b_k \in L_{p_i}} |\hat{r}_i(b_k) - p_i(b_k)| \tag{4.3}$$

Where L_{p_i} is defined as the ordered sequence $[b_1, ..., b_N]$ of items $b_k \in B$, representing the top-N recommendation list for user a_i. Note that the list is not necessarily sorted on $p_i(b_k)$. Since a user is unlikely to rate all the TV programmes in the listing, $r_i(b_k)$ might not be defined (*i.e.* $r_i(b_k) = \perp$). Therefore a strategy is needed to deal with this. It is possible to exclude the item from the computation when it is not rated by the user for example. The main disadvantage of this approach is that it might not be possible to evaluate a recommendation list. Another possibility is to take some default value as the rating [HKTR04]. The default value can be either user-dependent (*e.g.* the average rating of the user, or the average rating of the user for other episodes of the same series) or user-independent (*e.g.* a constant value). The downside of this approach is that the true user's rating can differ greatly from the default value.

This research tries to approximate the true user's rating and therefore uses the most specific data available for each user. $\hat{r}_i$ as used in Equation 4.3 is therefore specified as follows:

$$\hat{r}_i(b_k) = \begin{cases} r_i(b_k) & \text{if } r_i(b_k) \neq \perp \\ avg(B_i^c) & \text{if } r_i(b_k) = \perp \wedge B_i^c \neq \varnothing \\ avg(B_i) & \text{otherwise} \end{cases} \tag{4.4}$$

where

$$avg(B_i) = \frac{1}{|B_i|} \sum_{b_k \in B_i} r_i(b_k)$$

In this case B_i^c is the set of episodes of the TV series to which b_k belongs (*i.e.* all Friends episodes when b_k is an episode of Friends for example) that are rated by user a_i.

Classification accuracy metrics, such as ROC, measure a recommender system's ability to recommend interesting items (*i.e.* classify items as interesting / not interesting). Therefore these metrics require the items to be classified as interesting or not. This classification can be done, based on the user's true rating and the recommender system's predicted rating for an item. In other words, the rating scale needs to be transformed into a binary scale. This is done by assuming that items rated greater than or equal to a certain threshold θ_i are interesting. All other items are assumed to be not interesting. The predictions made by the recommender system are classified analogous. The situation where an item was not rated by the user can occur here too and the same strategies to deal with this can be applied.

The hit and miss rates used to construct the ROC curve are defined as:

$$\text{hit rate} = \frac{|B_{tp}|}{|B_{tp}| + |B_{fn}|}$$

$$\text{miss rate} = \frac{|B_{fp}|}{|B_{fp}| + |B_{tn}|}$$

Here B_{tp}, B_{fp}, B_{tn} and B_{fn} are the sets of true positive, false positive, true negative and false negative items respectively. These sets are formally specified as:

$$\begin{aligned} B_{tp} &= \{b_k \in B : r_i(b_k) \geq \theta \wedge p_i(b_k) \geq \theta\} \\ B_{fn} &= \{b_k \in B : r_i(b_k) \geq \theta \wedge p_i(b_k) < \theta\} \\ B_{fp} &= \{b_k \in B : r_i(b_k) < \theta \wedge p_i(b_k) \geq \theta\} \\ B_{tn} &= \{b_k \in B : r_i(b_k) < \theta \wedge p_i(b_k) < \theta\} \end{aligned}$$

TV programmes with a (predicted) rating greater than or equal to θ are considered to be relevant.

The construction of the ROC curve is described in subsection 3.2.2. Calculating the area under the curve is straightforward.

The second classification accuracy metric that is incorporated in this research is precision. Precision is defined as the number of recommended items that are considered relevant by the

user, divided by the number of recommendations. Again, a recommendation is considered relevant when the user's rating $\hat{r}_i(b_k)$ it greater than or equal to θ. Since there are ten items recommended to each user, the metric that is calculated is what is called the precision at 10.

A rank accuracy metric such as Spearman's rank correlation measure is interesting to incorporate in the research since it can help to determine whether users are sensitive to the ordering of the recommendations. In the context of this research Spearman's ρ is defined as shown in Equation 4.5.

$$\rho = \frac{\sum(x - \overline{x})(y - \overline{y})}{N \cdot stdev(x) \cdot stdev(y)} \tag{4.5}$$

Where $x_1, ..., x_N \in \mathbf{x}$ are the ranks of the items $b_k \in L_{p_i}$ (*i.e.* $1, ..., N$) and $y_1, ..., y_N \in \mathbf{y}$ are the ranks corresponding to the first N items $b_k \in R_{a_i}$. Here R_{a_i} is defined as the ordered sequence $[b_1, ..., b_M]$ of items $b_k \in B$ rated by user a_i, sorted on ratings $\hat{r}_i(b_k)$.

Instead of trying to recreate what the user's top-N recommendation list should have looked like based on his ratings and comparing it to the list created by the recommender system, another approach can be taken. By comparing the ranking of the N items in the recommended list L_{p_i} to the user's ranking of the same N items the recommender system's ability to rank the items relative to each other can be measured.

4.7.2. Diversity

Ziegler et al. [ZMKL05] observed that a more diverse recommendation list can lead to higher user satisfaction, even though its accuracy is lower. Therefore diversity is one of the dimensions investigated in this research. Ziegler et al. proposed a metric to capture the diversity of a list with respect to a certain criterion, called the intra-list similarity metric (as shown in section 3.5). Equation 4.6 aggregates the scores for several different criteria into one metric.

$$Diversity = \frac{1}{|C|} \sum_{c \in C} ILS(L_{a_i})^{-1} \tag{4.6}$$

The similarity function $sim(x, y)$ is redefined as $sim_c(x, y) : B \times B \rightarrow [0, 1]$, a function that maps two items onto a value from 0 to 1, indicating their similarity with respect to a certain criterion $c \in C \subseteq \{genre, actors, director, series, premiere, ...\}$. Since a intra-list similarity score implies a low diversity the inverse of the function is taken.

Because the similarity function for each of the criteria is simply defined as:

$$sim_c(x, y) = \begin{cases} 1 & \text{if } x \equiv y \text{ with respect to criterion } c \\ 0 & \text{otherwise} \end{cases}$$

the inverse of the intra-list similarity function can be defined as the number of items that differ with respect to criterion c, divided by the total number of items in the list.

Besides this off-line metric, diversity will also be measured by asking the user to rate the diversity of the list. Users can indicate how diverse they think the list of recommendations is

by assigning 1 (not diverse) to 5 (really diverse) stars to it.

4.7.3. Novelty

Novelty is an indicator for the "non-obviousness" of a recommendation. It is defined analogous to the metric introduced by Baeza-Yates and Ribeiro-Neto [BYRN99] as the ratio between the number of novel items in the recommendation list and the total number of items in this list. Equation 4.7 shows the exact definition of novelty adapted in this research.

$$Novelty = \frac{1}{|L_{p_i}|} \sum_{b_k \in L_{p_i}} s_i(b_k) \tag{4.7}$$

Here $s_i(b_k)$ is a function that returns 1 when the item b_k is novel to user a_i and 0 otherwise.

$$s_i(b_k) = \begin{cases} 1 & \text{if } b_k \text{ is marked as novel by user } a_i \\ 1 & \text{if } b_k \in B_i^c \\ 0 & \text{otherwise} \end{cases}$$

A recommendation is considered novel when the user has not seen the TV programme before. Since the evaluation application cannot possibly know whether the user has seen the programme before he needs to indicate this himself by (un)checking a checkbox for example.

4.7.4. Serendipity

Serendipity is defined in section 3.7 as "the experience of discovering an unexpected and fortuitous item" or by the Oxford American Dictionary as "the occurrence and development of events by chance in a happy or beneficial way". These definitions entail two different aspects: unexpectedness and fortuitousness. A recommendation is said to be unexpected when it is novel and fortunate when the user considers it interesting. This implies the following logical relation:

$$serendipitous \iff novel \wedge interesting$$

So, when a recommendation is serendipitous it is also novel and interesting (and the other way around). The previous subsection defined when a recommendation is novel and subsection 4.7.1 defined when a recommendation is considered interesting. Combining these definitions leads to Equation 4.8, which is the ration of the number of serendipitous items and the total number of items in the recommendation list.

$$Serendipity = \frac{1}{|L_{p_i}|} \sum_{b_k \in L_{p_i}} s'(b_k) \tag{4.8}$$

In this formula $s'(b_k)$ is a function that returns 1 when the item b_k is both novel and interesting. Otherwise it will result in 0.

$$s'(b_k) = \begin{cases} 1 & \text{if } b_k \text{ is novel} \wedge r_i(b_k) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

Like diversity this dimension will also be measured by asking the user to rate the serendipity of the list using a scale from one to five.

4.7.5. Programme overlap

Another aspect that might influence user satisfaction when recommending lists of TV programmes is programme overlap. Consider for example a list of ten TV programmes that is recommended to a user. The list might normally be rated high by the user, but when all ten TV programmes are broadcasted at the same time the user probably considers the list useless.

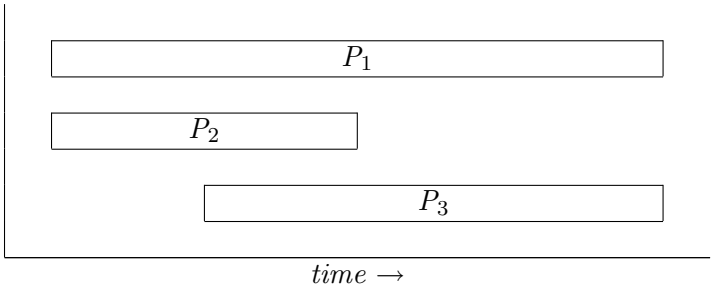

Table 4.2.: Three programmes in time

A possible way to determine the degree of overlap within a list of recommendations is to calculate the ratio of the time you can effectively watch any TV programme and the total duration of the TV programmes in the list. Take the three programmes shown in Table 4.2 for example. The maximum time you can watch any of these TV programmes is equal to the duration of P_1, say 1 hour, while the sum of the duration of the three programmes is 135 minutes. The overlap ratio for the list therefore is $1 - \frac{60}{135}$. This metric is formally defined in Equation 4.9.

$$Overlap = 1 - \frac{\sum_{b_k \in L_{p_i}} \sum_{b_l \in L_{p_i} \wedge b_k \neq b_l} overlap(b_k, b_l)}{\sum_{b_k \in L_{p_i}} duration(b_k)} \tag{4.9}$$

Another possible way to measure the degree of overlap within a list, is to take the ratio of the number of programmes you can watch from start to end and the total number of TV programmes in the recommendation list. In the example shown in Table 4.2 the user can only watch one programme out of three, so the overlap ration is $1 - 1/3 = 2/3$.

4.8. Evaluation setup

In order to run the user experiment the recommender system and evaluation application must actually be build. Building such a system is a challenge in its own right since many practical

issues like usability, scalability and flexibility for example become relevant. As described in Appendix A there do exist recommendation frameworks, but none of these systems allow users to provide feedback on the recommendations itself. Furthermore none of the existing systems enables you to easily use multiple recommendation strategies alongside each other and none of the systems provided a user interface. Therefore building the evaluation application is inevitable.

Figure 4.4 shows the evaluation application architecture. The architecture contains two main components, the recommender system and the web application used by the users to provide feedback on TV programmes and recommendations.

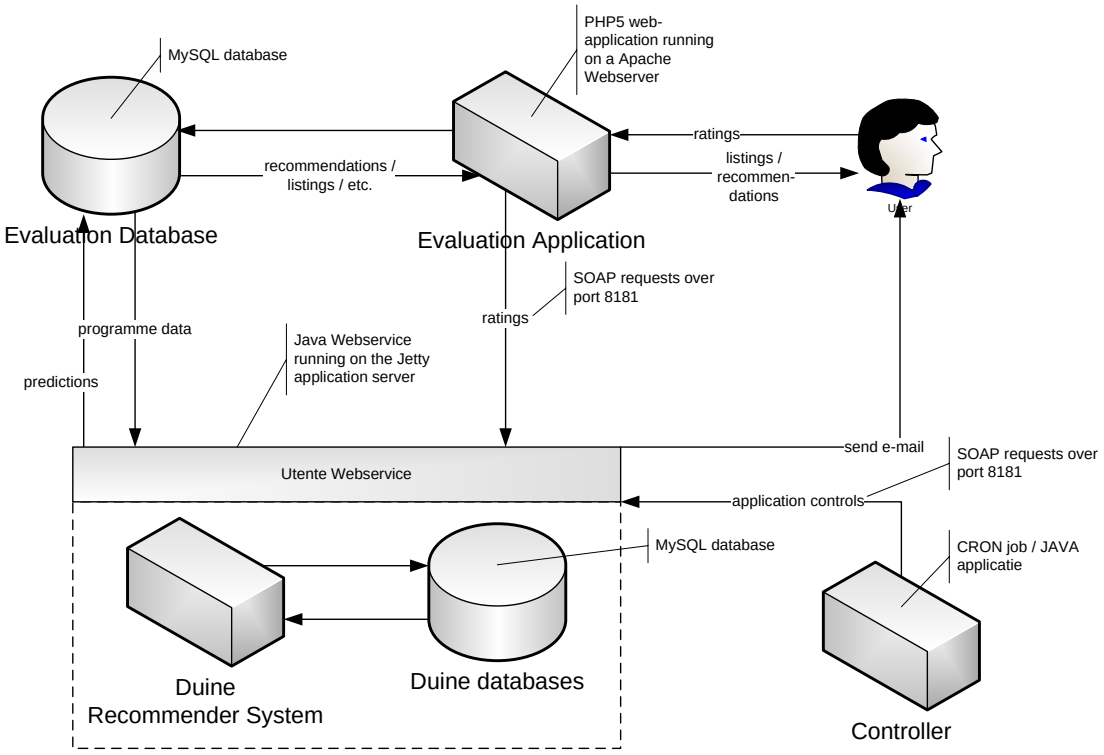

Figure 4.4.: System design

4.8.1. Recommendation application

The recommendation engine generates recommendations for TV programmes based on feedback provided by the users through the web application. One of the recommendation strategies is actually using an open source recommender system called Duine[1] [vS05] to generate predictions. The other recommendation strategies are build from scratch using the Java programming language.

[1] http://sourceforge.net/projects/duine

4.8.2. Web application

The web application acts as the user interface through which the participants provide feedback on TV programmes and recommendations. Furthermore it is responsible for the distribution of recommendation list generated using different strategies among the users of the application.

4.9. Conclusion

This chapter described in detail how the user experiment that will be conducted as part of this research will be performed. The goal of the user experiment is to gather sufficient data, so the coherence between user satisfaction and the individual properties of a recommendation list can be determined. To achieve this goal an actual recommender system will be implemented as part of an on-line evaluation application. The user study consists of two phases. During the first phase users will have to rate TV programmes they are familiar with or that they recently watched. Based on the feedback on TV programmes, the recommender system that is part of the evaluation application will start generating recommendations. In phase two of the study the application will recommend TV programmes that are broadcasted that evening to its users. These recommendation lists are deliberately varied with respect to the accuracy, diversity, novelty, serendipity and programme overlap dimensions. This is achieved by generating seven different lists of recommendations and presenting these lists to the users in a round-robin fashion. Users are asked to provide feedback on how satisfied they are with this list. Furthermore, 15 different metrics are applied to the recommendation lists so their values can be compared to the corresponding user satisfaction of the same list.

5

Determination of Coherence

The previous chapter described how the data was collected that is needed to determine whether user satisfaction and the other evaluation metrics cohere. This chapter describes how the data was processed and what correlations were discovered. Furthermore the results of a questionnaire among the participants of the user study, inclined to find out what other aspects might influence user satisfaction with respect to TV programme recommendations, is presented.

In this chapter the following will be discussed

- The pre-processing and filtering steps taken before the correlations can be calculated (Section 5.1).
- Some statistics on the participating users (Section 5.2).
- The correlations that exist between the different dimensions and user satisfaction (Section 5.3).
- The results of the questionnaire (Section 5.6).

5.1. Processing the data

The data collected during the user study needs to be processed in order to determine correlations between user satisfaction and the dimensions defined in section 4.7. The following steps were taken to come to the results of the user study:

1. Filter the raw data;
2. Implement the metrics;
3. Apply the metrics and

4. Plot the data and calculate correlations using Matlab.

In the first step the data was filtered. All users that registered themselves for the experiment, but did not made it to the recommendation phase were removed, together with all the data related to them (such as ratings, recommendations, their profile, and so on). This filtering step reduced the user base from 133 to 70 users. The effect of this filtering step is nil since the 63 users that were filtered out did not participate in the recommendation phase of the user study and thus never rated a recommendation list. Therefore they did not generate any data points and had no effect on the results.

The top-5 of users that rated most TV programmes and recommendations accounts for 51.1% of all ratings. Since it is possible that they randomly rated all the TV programmes and recommendations just to win the DVD box, the effects of these users on the results were investigated by filtering them out. The results differed only slightly when the top-5 rating users were removed so their rating behaviour appears to correspond to that of the other 65 users and there is no reason to believe that they randomly rated the items or tried to influence the research in any way.

Implementing the metrics described in section 4.7 in the Java programming language is quite straightforward. The mathematical formulas are easily transformed into Java code. The difficulty, however, was to write code that is efficient so it does not take too long to calculate a metric's values. After the metrics were implemented they were applied to the (filtered) data. Most implementations of the metrics have indicators that indicate the validity of each metric's value. They indicate for example how many of the recommended TV programmes were not rated by the user. The validity indicators determine whether the metric's value is considered as a valid data point or not. Valid data points were printed in .mat files which were imported into Matlab. Matlab and its statistical toolbox were used to further process the data.

5.2. User statistics

In order to come to a user base consisting of diverse users, everybody could register and participate in the research. Colleagues, fellow students, friends and family were explicitly asked to participate, while others were recruited through posters, bulletin boards, on-line messengers and websites. Eventually 141 users registered themselves. Eight of them never rated a TV programme while 133 users rated at least one programme. From the 133 users that rated one or more TV programmes, 63 never rated a recommendation list. The other 70 users participated actively during the entire user study. Each user that registered himself provided some demographic information. Figure 5.1 shows the age, household size and level of education for the 70 users that actively participated in the research. In the education level plot the bars represent the number of users with an academic, HBO, MBO (level 3 or 4), VMBO/MBO (level 2) and MAVO/HAVO/VWO degree respectively.

Figure 5.2 shows marital status (1 = married, 2 = single and 3 = living together) and the industry in which the user is working / studying. Most users are working in the field of information and communication technology (bar 17), health care (bar 13) and telecommunication (bar 20). The last bar (27) represents the users not working in any of the predefined sectors.

The number of TV programmes and recommendation lists rated per user differs greatly (as

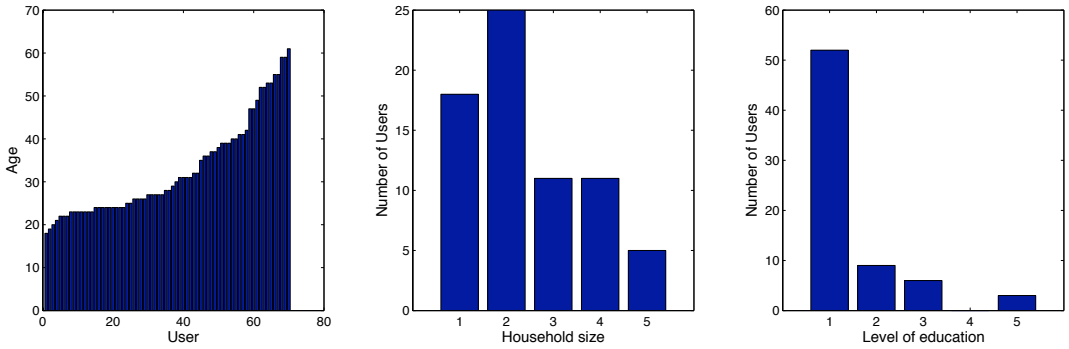

Figure 5.1.: User's age (left), household size (middle) and level of education (right)

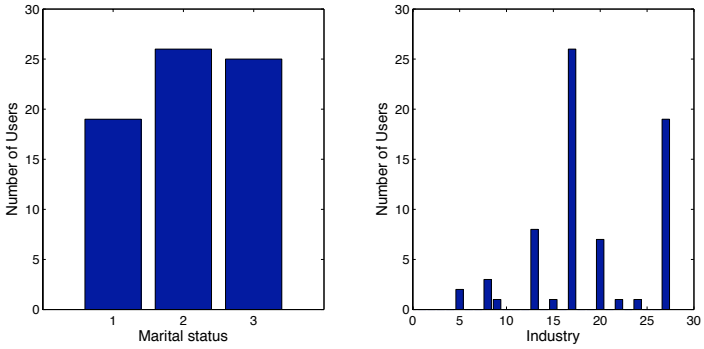

Figure 5.2.: User's marital status (left) and industry (right)

shown in Figure 5.3). The top-5 rating users provided more than half of the total number of ratings for TV programmes. The number of ratings for recommendation lists is less contrasting.

5.3. Determine correlations

For each of the predefined metrics its correlation with user satisfaction is calculated using Matlab. The measure of correlation that is adapted is Spearman's rank correlation coefficient, since it is a non-parametric measure for correlation that does not require the correlation to be linear. Furthermore this correlation coefficient does not make any assumptions on the frequency distribution of the variables [HW99]. The latter is important since users might not experience an equal distance between the successive ratings (*e.g.* the perceived distance between 3 and 4 stars might be smaller than the distance between 4 and 5 stars).

Whether a correlation coefficient is significant or not is determined by the corresponding p-value [Kre99]. The p-value is defined as

$$P(O \mid H_0)$$

so the chance that the observed situation (O) happens under the assumption that the null

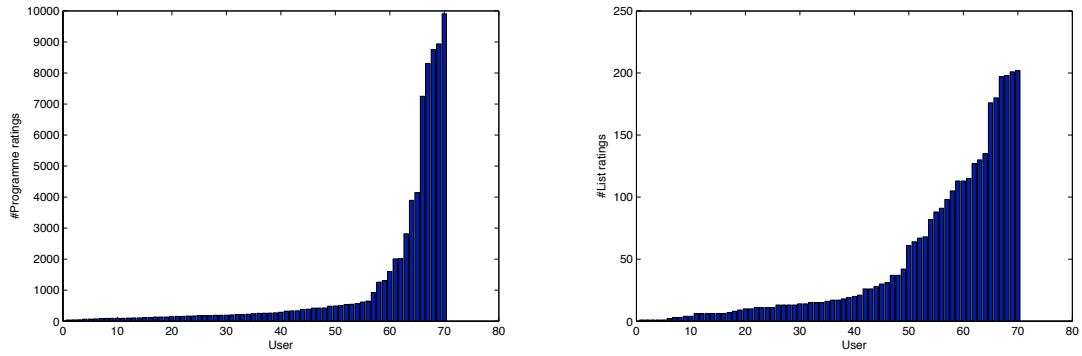

Figure 5.3.: Number of programme and list ratings per user

hypothesis (there is no correlation between the two variables) holds. The null hypothesis is rejected when $P(O \mid H_0) < \frac{1}{2}\alpha$, where α is the significance level of choice (usually 1% or 5%). The significance level is divided by two since a two-tailed significance test is performed. A small p-value thus indicates that there exists a correlation that is significantly different from 0.

Besides calculating the correlation between each metric and user satisfaction, the data points are plotted to see how they are distributed. The following subsections describe the results for each of the dimensions and metrics described in section 4.7.

5.3.1. Accuracy

Accuracy metrics empirically measure to what extent a rating or ranking, as predicted by the recommender system, matches the rating/ranking of the user. As mentioned in section 4.7.1, the user's rating for a recommended TV programme is not always known, so a strategy to cope with this is needed. The strategy adopted in this research tries to approximate the user's rating as close as possible. The validity indicators of each of the accuracy metric implementations indicate how certain the metric was of the user's ratings. When the uncertainty is too high, the data point will be discarded. Data points are discarded when less than θ_v programmes and/or series were rated by the user. Thus with $\theta_v = 3$ a list containing 10 recommendations is rejected when 7 or more TV programmes were not rated by the user.

The investigated thresholds are $\theta_v = 0$, $\theta_v = 3$ and $\theta_v = 10$. With $\theta_v = 0$ all lists are accepted since the threshold is effectively left aside. The average number of cases in which the programme nor the TV show was rated is 2.9768. So to discard all lists where the number of unrated programmes and/or series is above average, $\theta_v = 7$ is chosen. Finally, $\theta_v = 10$ requires all programmes and/or series to be rated. This constraint causes a lot of data points to be discarded, but ensures a close approximation of the user's rating. The resulting correlation coefficients for $\theta_v = 0$, $\theta_v = 7$ and $\theta_v = 10$ are shown in Table 5.1.

The number of data points on which the correlations are based differs for Mean Absolute Error, Area under ROC and precision depending on the value of θ_v. For $\theta_v = 0$ the number of data point for MAE is 2755 for example, but for $\theta_v = 7$ it is 1696 and for $\theta_v = 10$ it is further decreased to 1182. These numbers are the same for the Area under ROC metric,

while precision has slightly more valid data points (3085, 1917 and 1264 respectively).

	$\theta_v = 0$		$\theta_v = 7$		$\theta_v = 10$	
Metric	**Corr.**	**Sig.**	**Corr.**	**Sig.**	**Corr.**	**Sig.**
Mean Absolute Error	-0.406	0.000	-0.611	0.000	-0.698	0.000
Area under ROC curve						
- with $\theta_r = 3$	0.441	0.000	0.492	0.000	0.485	0.000
- with $\theta_r = 4$	0.289	0.000	0.326	0.000	0.367	0.000
Precision						
- with $\theta_r = 3$	0.559	0.000	0.673	0.000	0.743	0.000
- with $\theta_r = 4$	0.563	0.000	0.666	0.000	0.708	0..000
Spearman's correlation						
- User's list	-0.098	0.000	-0.098	0.000	-0.098	0.000
- User's top-10	0.175	0.000	0.175	0.000	0.175	0.000
Relative Edit Distance						
- User's list	-0.243	0.000	-0.243	0.000	-0.243	0.000
- User's top-10	-0.024	0.175	-0.024	0.175	-0.024	0.175

Table 5.1.: Accuracy metric correlations

Both the area under ROC and precision metric work on a binary scale (either relevant or not relevant). The threshold θ_r determines when a (recommendation for a) TV programme is considered relevant or not. A TV programme rated/predicted θ_r or higher is considered relevant, otherwise it is considered irrelevant. The effect of θ_r on the correlations is shown in the table.

The p-values in Table 5.1 show that all correlations are significant, except for the correlation between user satisfaction and the user's top-10 RED at the 0.05 significance level. Furthermore it shows that the strongest correlation exists between user satisfaction and precision. When $\theta_r = 3$ and $\theta_v = 10$ the correlation coefficient is 0.743. This strong correlation indicates that users find it important that the recommender system can accurately classify a TV programme as (non)relevant and subsequently only recommends the relevant programmes. Recommending TV programmes that a user does not like reduces user satisfaction.

The correlation between Mean Absolute Error and user satisfaction is slightly weaker and negative into the bargain. The correlation is negative since a higher Mean Absolute Error leads to lower user satisfaction. The correlation might be weaker because users probably do not care whether the recommender is able to predict their ratings correctly, as long as it is able to recommend relevant TV programmes. Predicting the user ratings correctly is closely related to recommending relevant TV programmes however, since all recommendations are based upon these predictions. This is substantiated by the existence of a strong correlation between precision and Mean Absolute Error (a correlation coefficient of -0.805).

The correlation coefficients for the rank accuracy metrics Spearman's correlation and Relative Edit Distance are the weakest accuracy metrics. This probably indicates that the order in which the recommendations are presented is not really important to the user. However, the weak correlations might also be caused by the metric's inability to construct the true user's top-10 list of recommendations. The metric constructs this list based upon the user's

ratings for series as provided before and after the list was recommended (see Equation 4.4), but whether it corresponds to the true user's top-10 is uncertain and hard/impossible to verify. Furthermore programmes that are not rated by the user are excluded from the list and therefore penalise the metric.

5.3.2. Diversity

The diversity dimension is measured with four different metrics. First of all each user is asked to rate the diversity of a list of recommendations on a scale from 1 to 5. Furthermore diversity is measured with respect to genre, series and channel. The resulting correlations are listed in Table 5.2.

Metric	**Corr.**	**Sig.**
User's diversity	0.464	0.000
Genre diversity	0.129	0.000
Series diversity	0.217	0.000
Channel diversity	0.171	0.000

Table 5.2.: Diversity metric correlations

All diversity metrics appear to be significant, but none of the off-line metrics is correlated to user satisfaction as strong as the on-line metric. From the three off-line metrics, series level diversity has the strongest correlation to user satisfaction, probably because users penalise lists that contain reruns of the same TV programme.

The perception of diversity is apparently not determined by one single aspect, but probably by a combination of diversity of aspects such as genre, series and channel. Other aspects that might be contributing to the perception of diversity are the kind of programme, actors, director or just the feeling of variation.

5.3.3. Novelty

The novelty aspect of a recommendation list is measured with the series level novelty metric as explained in subsection 4.7.3. The correlation between this metric and user satisfaction is shown in Table 5.3.

Metric	**Corr.**	**Sig.**
Series level novelty	-0.056	0.001

Table 5.3.: Novelty metric correlations

The correlation with series level novelty is significant at the 5% significance, but the correlation is very weak and slightly negative. This weak correlation might be caused by the fact that most participants only provide (positive) feedback on TV programmes they like. The series level novelty metric assumes that programmes that were not rated by the user are novel to him. Therefore many programmes the user does not like are unintentionally labelled as novel. To overcome this problem the feedback provided by the users on the novelty of each programme is used. Users could label each recommended TV programme as "known" or

"unknown". Unfortunately many users forgot to label the recommendations on a regular basis, making it impossible to distinguish between novel programmes and programmes that the user had seen before.

5.3.4. Serendipity

The series level serendipity metric uses the same method as series level novelty to determine whether a TV programme is novel to the user or not. Programmes that are novel and relevant are said to be serendipitous. Therefore the TV programmes must be mapped upon a binary scale as with classification accuracy metrics. Here the same threshold values for θ_r are used to determines when a (recommendation for a) TV programme is considered relevant or not. The resulting correlation coefficients are shown in Table 5.4.

	$\theta_v = 0$		$\theta_v = 7$		$\theta_v = 10$	
Metric	**Corr.**	**Sig.**	**Corr.**	**Sig.**	**Corr.**	**Sig.**
User's serendipity	-0.279	0.114	-0.279	0.114	-0.279	0.114
Reconstructed user's serendipity	0.517	0.000	0.628	0.000	0.609	0.000
Series level serendipity						
- with $\theta_r = 3$	0.140	0.000	0.167	0.000	0.136	0.000
- with $\theta_r = 4$	0.134	0.000	0.139	0.000	0.107	0.000

Table 5.4.: Serendipity metric correlations

The correlation between the ratings of users on serendipity and their satisfaction appears not to be significant since the p-value is larger than the significance level of 0.05. The measure for serendipity based on feedback provided by the users is probably negative since they also rate "surprisingly bad" lists of recommendations as serendipitous. Users thus provide feedback on how surprised they are with respect to the recommendation list instead of how serendipitous the list is experienced. To correct for this the user's serendipity is multiplied by the average rating for the recommended TV programmes, resulting in the reconstructed user's serendipity. This way a "surprisingly bad" list becomes less serendipitous than a list that is surprising and good. The correlations coefficients for the reconstructed user's serendipity are quite high. Moreover all reconstructed user's serendipity correlations appear to be significant. However, these high correlations might be caused by the involvement of the (strongly correlated) precision.

The off-line metric series level serendipity also appears to be significant, although it is a lot weaker than the reconstructed user's serendipity. The problems that series level novelty suffers from, namely that irrelevant programmes are not rated by users and therefore are labelled as novel, do not affect the series level serendipity metric. Programmes that are incorrectly labelled as novel are only labelled as serendipitous when the true user's rating (*i.e.* $\hat{r}_i(b_k)$ as defined in Equation 4.4) is larger than or equal to θ_r. The resulting correlation coefficients show that serendipity is important to users, but only to a limited degree.

5.3.5. Programme overlap

The correlations between user satisfaction and the two metrics for programme overlap are listed in Table 5.5.

Metric	Corr.	Sig.
Programme Overlap Ratio	0.015	0.401
Effective Overlap Ratio	-0.212	0.000

Table 5.5.: Programme overlap metric correlations

Table 5.5 shows that there is a large difference between the two programme overlap metrics with respect to their correlation with user satisfaction. The p-value of 0.401 indicates that there is a 40% change that a correlation coefficient of 0.015 is observed under the assumption that there exists no correlation between the Programme Overlap Ratio and user satisfaction. Therefore the null hypothesis cannot be rejected. The absence of a correlation between Programme Overlap Ratio might be caused by the fact that users did not take the time on which a TV programme is broadcasted into consideration and therefore do not notice any possible overlap.

The negative correlation between Effective Overlap Ratio (EOR) and user satisfaction seems really counterintuitive. It means that the more programmes overlap (lower EOR) the more satisfied a user is with the list of recommendations. The cause of this observation probably lies in the fact that the most popular TV programmes are broadcasted within a small timeframe (say 8:00pm - 10:00pm) on different channels. Users that are not concerned with programme overlap and rate the recommendation lists based on the presents of relevant TV programmes thus rate lists with much overlap (and a low EOR) higher. Furthermore, many TV programmes have one or more reruns late at night. These programmes are less likely to overlap with other recommendations, and thus lead to a high Effective Overlap Ratio. As described in subsection 5.3.2, lists with many duplicates on a series level are less valued by users however. This might contribute to the negative correlation as well.

5.4. Clusters / types of users

The correlations presented in the previous section are calculated over all users that participated in the user study. It is not unlikely however that the importance of the dimensions differs per user. Some users might prefer a diverse list over a highly accurate one, while others prefer a list that is serendipitous. These differences make that the overall correlation for a specific metric might not correspond to the correlation between the metric and the satisfaction of a specific user.

In order to investigate whether there exist groups of users that favour a specific dimension or combination of dimensions over another, users can be clustered based on the dimensions they deem to be important. For each of these groups the correlations to the metrics that are discriminating to that group must be stronger while others are weaker. To create the clusters of users with similar discriminating metrics the following steps need to be taken:

1. Calculate for each user the correlation between the user's satisfaction and each metric.

2. Create clusters of users that are most similar with respect to the correlations calculated in step 1.

3. Calculate the correlations per metric for each of the clusters created in step 2.

Each of these steps will be described in more detail in the following subsections.

5.4.1. User's correlations

The importance of each of the dimensions presented in section 4.7 with respect a specific user is determined by calculating its correlation with that user's satisfaction. Analogous to the correlation measure used to calculate the overall correlations, Spearman's rank correlation coefficient will also be adapted here. The result is a matrix with for each user a row and for each metric a column where each cell is the corresponding correlation coefficient.

5.4.2. Clustering the users

Clustering algorithms divide data into groups, called clusters, such that the similarity within a group (the intra-cluster similarity) is maximised according to some distance measure, while the similarity between groups (the inter-cluster similarity) is minimised. The distance measure used to cluster the users is the Euclidean distance between the user's correlations (as described in the previous subsection). This way the clustering algorithm groups users with similar correlations together.

The clustering toolkit Cluto [1] is able to cluster high dimensional data maximisedsets using a number of clustering algorithms that operate either directly in the object's feature space or in the object's similarity space. The toolkit is applied to create the clusters of users using the clustering criterion function shown in Equation 5.1. This function is maximised to get an optimal clustering.

$$\sum_{i=1}^{k} \sqrt{\sum_{v,u \in S_i} sim(v,u)} \tag{5.1}$$

In this equation S_i represents the ith cluster (with $1 \leq i \leq k$) and v and u are objects in a certain cluster.

Determining the number of "true" groups in a data set is a fundamental and largely unsolved problem in cluster analysis [SJ03, RWN03, TWH01]. This makes is really hard to determine the number of clusters k you are trying to find. The approach taken in this research to determine k is to generate clusters using $2 \leq k \leq 10$. For each of these clusters the descriptive features are calculated. The set of descriptive features is determined by selecting the metrics that contribute the most to the average similarity between the items in a cluster. Next, k is chosen as the largest number of clusters such that the intersection of the main descriptive feature for each of the clusters is empty. It appears that $k = 4$ meets this criterion and therefore the users will be grouped into four clusters.

[1] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

Table 5.6 shows the internal and external quality measures as provided by Cluto. In this table each of the rows corresponds to one of the clusters, indicated by its cluster id (Cid). The column labelled "Size" displays the number of users in that cluster. The column "ISim" displays the average similarity between the users in the cluster (*i.e.* the internal similarity) while the column "ISdev" displays the internal standard deviation of the average similarities. The columns "ESim" and "ESdev" display the same information for external similarities. The right most column labelled "Descriptive features" displays the three most descriptive features for the cluster together with a percentage that indicates how descriptive that feature is.

Cid	**Size**	**ISim**	**ISdev**	**ESim**	**ESdev**	**Descriptive features**
1	15	0.630	0.085	0.186	0.104	UsersDiversity (22.4%), UsersSerendipity (17.0%), Precision (13.4%)
2	24	0.614	0.106	0.217	0.070	Precision (25%), MeanAbsoluteError (19.2%), UserDiversity (10.5%)
3	12	0.407	0.097	0.011	0.157	SeriesLevelNovelty (31.0%), SeriesLevelSerendipity (18.8%), Precision (11%)
4	20	0.179	0.155	0.117	0.142	UsersSerendipity (23.1%), MeanAbsoluteError (16.0%), Precision (12.3%)

Table 5.6.: Internal and external quality measures for 4-way user clustering

Table 5.6 shows that the largest cluster (cluster 2, consisting of 24 users) contains users that are similar with respect to the correlation between their user satisfaction and the accuracy metrics precision and Mean Absolute Error. The intra-cluster similarity is high, indicating that the similarity between the users in the cluster is high as well. The external similarity however is also fairly high, probably because precision is also important to the users in the other clusters. The importance of precision to the other clusters is illustrated by the fact that this metric is also descriptive to these clusters, but to less extent.

The cluster with the highest intra-cluster similarity (cluster 1) has user's diversity and user's serendipity as its most descriptive features. The users in this cluster probably value diversity and serendipity above average while their correlation coefficients for the other metrics differ to greater extent.

Cluster 3 contains users that have equivalent correlations between user satisfaction and series level novelty, series level serendipity and precision. Series level novelty is the strongest descriptive feature for this cluster, indicating that this metric is most similar among the members of the cluster.

The fourth cluster has a low average similarity between the users in the cluster, probably because this cluster contains the residual user, *e.g.* the users that did not fit to one of the other clusters. The internal standard deviation is the highest for this cluster which is also an indication that this cluster is not really homogeneous.

5.4.3. Correlations per cluster

In order to verify whether the identified clusters indeed contain users that value the same dimensions, their correlations can be calculated for each of the metrics. The correlation coefficient of the descriptive feature should differ significantly from the correlation found when all users are considered.

To calculate the correlations between user satisfaction and each of the metrics, only data points associated with users contained in a specific cluster are taken into account. The results are listed in Table 5.7. The last row in both tables show the overall correlation for the metrics.

Cid	Mean Absolute Error	Area Under ROC	Precision	Users Diversity	Channel Level Diversity	Genre Level Diversity	Series Level Diversity	Series Level Novelty
1	-0.392	0.385	0.501	0.636	0.174	0.207	0.208	0.012
2	-0.486	0.477	0.632	0.446	0.250	0.154	0.276	-0.164
3	-0.149	0.474	0.297	0.241	-0.160	-0.129	-0.062	0.463
4	-0.259	0.321	0.469	0.247	-0.084	0.011	0.145	-0.082
-	-0.406	0.441	0.559	0.464	0.171	0.129	0.217	-0.056

Cid	Users Serendipity	Reconstructed Users Serendipity	Series Level Serendipity	Overlap Ratio	Effective Ratio	Top-N Spearman	User List Spearman	User List RED
1	0.543	0.718	0.048	-0.030	-0.214	0.160	-0.040	-0.135
2	-0.205	0.531	0.195	0.070	-0.230	0.190	-0.199	-0.281
3	-0.235	0.085	-0.026	-0.145	-0.152	0.142	0.274	-0.310
4	-0.511	0.045	0.275	-0.036	-0.142	0.133	0.027	-0.168
-	-0.279	0.517	0.140	0.015	-0.212	0.175	-0.098	-0.243

Table 5.7.: Correlations per cluster with $\theta_v = 0$ and $\theta_r = 3$

Table 5.7 shows that the correlations for cluster 1 with respect to the accuracy metrics are weaker than the overall correlations. The correlations for user's diversity, user's serendipity and reconstructed user's serendipity on the other hand became (much) stronger. This indi-

cates that the users in cluster 1 indeed consider the descriptive features as important. It also shows that the users in this cluster on average find diversity more important than users in the other clusters.

In cluster 2 all accuracy metrics appear to have a stronger correlation than the corresponding overall correlations. This is in line with the features that are descriptive to this cluster. The cluster consists of participants for whom accurate recommendations are most important. The differences between this cluster's correlations and the overall correlation coefficients are not as large as that of cluster 1. This is probably because cluster 2 is the largest cluster and therefore has a large impact on the overall correlations.

Cluster 3, with series level novelty as its most descriptive feature is indeed strongly correlated with this metric. Another observation is that mean absolute error and precision have only a relatively weak correlation with respect to the overall correlations for these metrics, while the correlation for area under ROC is similar.

The accuracy correlations for the fourth cluster are all weaker than the corresponding overall correlations. User's serendipity and series level serendipity on the other hand are stronger correlated to user satisfaction. In contrast to cluster 1 users in cluster 4 have an contrary correlation with respect to user's serendipity. This might be caused by the way the users rated the serendipity of the recommendation list. Users in cluster 4 probably rated a list that was really bad as serendipitous, while users in cluster 1 only considered a list to be serendipitous when is was surprisingly good.

5.5. Combinations of multiple variables

The previous two sections only focused on the relation between user satisfaction and a single other metric and did not take any combinations of metrics into account. It is possible however that combinations of metrics have a different effect on user satisfaction than each of the metrics in itself. Suppose a user gets a list of TV programmes recommended that is not really accurate, but very novel. Even though accuracy is important to the user he might appreciate the list since its novelty is high. It thus is possible that combinations of metrics enhance or abate each others effect on user satisfaction.

In statistics, multivariate analysis is concerned with the observation and analysis of more than one statistical variable at a time. Many different types of multivariate analysis exist such as multivariate analysis of variance (MANOVA), regression analysis, principal component analysis, artificial neural networks and so on [TFFT00]. Jan Telman, an expert in the field of statistics at TNO Science and Industry, was consulted to provide guidance and advice during this stage of the research.

Jan Telman advised to employ discriminant analysis since it is quite robust with respect to skewness. This is important since the observed values for the different variables are not normally distributed. The variables are tested for normality using the Jarque-Bera normality test [JB80].

Discriminant analysis identifies whether changes in the independent variables have a significant effect on the dependent variables. Furthermore it tries to identify the interactions among the independent variables and the association between dependent variables. Chapter 6 de-

scribes descriminant analysis in more detail and presents the model that resulted from the employment of discriminant analysis.

5.6. Questionnaire

When the user study was finished all participants were asked to fill in a questionnaire provided through the evaluation website. The purpose of the questionnaire was to learn from the participants, who were experienced users of a TV programme recommender system by then. The questionnaire consisted of ten questions. Some of these questions asked to indicate the importance of certain aspects to the user, while others tried to discover how the dimensions were interpreted. Furthermore the questionnaire was a means to discover whether important dimensions were missed. Detailed results of the questionnaire are listed in appendix D.

The questionnaire provided some interesting results. First of all the questionnaire confirmed many of the observations described above. Predictive and classification accuracy for example proved to be the most important dimensions to the users, while ranking accuracy is less important. This corresponds to the results of the questionnaire. Programme overlap appears not to be of great importance to users, even though they indicated in the questionnaire that viewability is. Obviously programme overlap is not an important factor when it comes to viewability and users take other aspects into consideration to determine the viewability (like the time when the TV programme is broadcasted).

Second, the questionnaire showed that users generally consider a list of recommendations diverse when the recommended TV programmes differ with respect to genre. However, the correlation between diversity as rated by the user and the genre level diversity metric (see Equation 4.6) is not as strong as series level diversity and channel level diversity.

Third, most participants indicated that they experience a recommendation list as serendipitous when it contains recommendations for TV programmes that they would not have discovered themselves. This definition however makes it really hard to measure whether a list is serendipitous and to what extent. Many users also consider a recommendation list containing TV programmes that they have not seen before serendipitous, this aspect was covered by the novelty metric however. Since the definition of serendipity differs greatly it might explain why there is no correlation between user satisfaction and serendipity.

5.7. Discussion

While the coherence between the dimensions identified in Chapter 3 and user satisfaction were determined some issues and possible limitations were identified. The remainder of this section will describe these issues.

First of all, Spearman's rank correlation measures to what extent a set of data points is strictly increasing (positive correlation) or decreasing (negative correlation). Figure 5.4 illustrates this. The left-most relation has a Spearman's rank correlation coefficient of 1, even though it is not a straight line. The relation in the middle has a correlation of -1. So, Spearman's rank correlation assumes a strong correlation when a greater diversity results in a higher (or lower) user satisfaction.

However, it is unlikely that an ever increasing diversity for example, results in a further

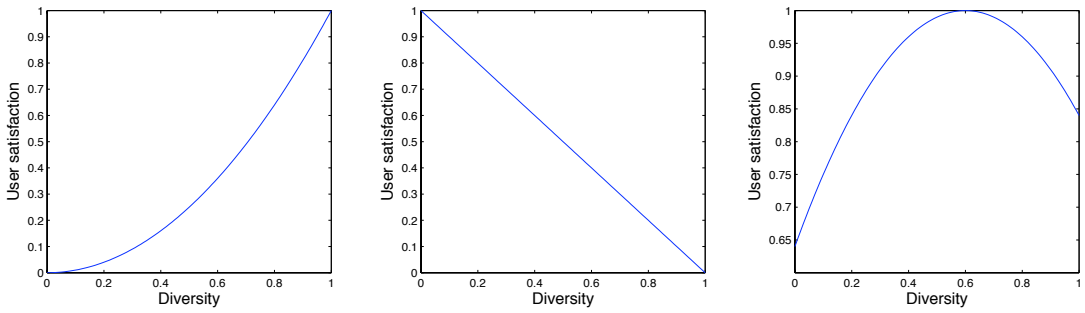

Figure 5.4.: A strictly increasing (left), strictly decreasing (middle) and a non strictly increasing / decreasing (right) relation

growth of the user satisfaction. Ziegler et. al. ([ZMKL05]) discovered that the relation between diversity and user satisfaction looks more like the right-most plot in Figure 5.4. Thus there appears to exist an optimum in the diversity of a recommendation list. Further diversification results in a decrease of the user satisfaction. The existence of such a relation results in a lower correlation coefficient, even though a strong relation might exist.

Secondly, the user base appeared not to be as diverse as anticipated. Since most of the participants were colleagues working at TNO Information and Communication Technology people working in the information and communication technology sector were overrepresented. The education level is also quite skewed since TNO employees are all highly educated. Furthermore a large number of university students and staff participated. The number of participants having an academic degree is therefore overrepresented as well.

Another issue is the translation of the user's ratings (on a scale from 1 to 5) onto a binary scale (relevant/irrelevant). The translation is done using a single threshold θ_r. Choosing this threshold is relatively arbitrary. Furthermore different users might consider differently rated items as relevant. One user might consider an item rated with 3 or more stars relevant for example, while another only considers items rated 5 stars as relevant. This issue can be solved by using user specific thresholds as suggested by Carenini [HKTR04, Car05]. Choosing this user specific threshold is also arbitrary however.

Finally, the quality of the user clustering might be suboptimal due to the lack of sufficient data for some users. For each of the dimensions the correlation to each user's satisfaction is calculated. But since some users only provided a small number of ratings their correlations might not be significant. When a large number of users is assigned to the same cluster based on poor data, this might affect the within cluster correlations.

6

Creating and Validating the Prediction Model

The ultimate goal of this research is to develop a method that is able to predict the user satisfaction achieved by an EPG/RS system, based on various off-line metrics. To achieve this a model is created based on the discriminating metrics that were identified in the previous chapter. How exactly this model is created and how it performs is the subject of this chapter.

In this chapter the following will be discussed

- How the prediction model is created (Section 6.1).
- How the validity of the model is tested (Section 6.2).
- Discussion of the results described in this chapter (Section 6.3).

6.1. Creating the model

Creating a model to predict the user satisfaction based on more than one variable requires the application of multivariate statistical methods. Multivariate methods simultaneously analyse data on several variables and incorporate knowledge on correlations among variables. The different variables are usually considered as a single element, a k-tuple consisting of $(x_1, x_2, ..., x_k)$ values [Acz93, TFFT00].

The prediction of the user satisfaction can be interpreted as a classification problem with five classes. Given a number of observed recommendation list properties $(x_1, x_2, ..., x_k)$ the goal is to classify the user's satisfaction as $1, 2, ..., 5$ stars. Statistical and machine learning approaches exist that try to solve the classification problem. On the advise of Jan Telman, an expert in the field of statistics, it is decided to employ discriminant analysis as a method to predict the user satisfaction based upon the observed list properties.

6.1.1. Discriminant analysis

Discriminant analysis is a technique for classifying a number of observations $\mathbf{x} = (x_1, x_2, ..., x_k)$ into predefined classes $\Pi_1, ..., \Pi_g$. In this research the number of groups g is five and the observations consist of the measured values for the off-line metrics. Each of the groups Π_j has a probability density function $f_j(\mathbf{x})$ that describes the chance that a measurement $\mathbf{x}$ is part of that group. In the case that $f_j(\mathbf{x})$ is known (which is rarely the case) allocating $\mathbf{x}$ to one of the groups is easy. It will be allocated to the group for which $f_j(\mathbf{x})$ has the largest value. When only the form of f_j is known, only the parameters of the function need to be estimated. This parameter estimation can be done using a data set that includes a (preferably large) number of observations, together with information on the group to which they belong. In the case that even the form of the probability density function is unknown there is an empirical approach to discriminant analysis where rules are tried to be found that discriminate between the different groups [MKB80, Krz88]. One such approach encompasses Fisher's linear discriminant function. This approach will be used in this research since the probability density functions are unknown for each of the five groups and since it is a robust method with respect to skewness (a measure for the asymmetry of the probability distribution).

A discriminant function $L = c + b_1x_1 + b_2x_2 + ... + b_kx_k$ is a linear function that maximises the among-groups variation relative to the within-groups variation [Acz93, Krz88]. The discriminant function thus is a direction in space that maximises the separation between groups and can be described by a linear function as shown in Figure 6.1. A detailed description of Fisher's discriminant function is not within the scope of this thesis. See [MKB80], [Krz88], [TFFT00] or [Acz93] for more details.

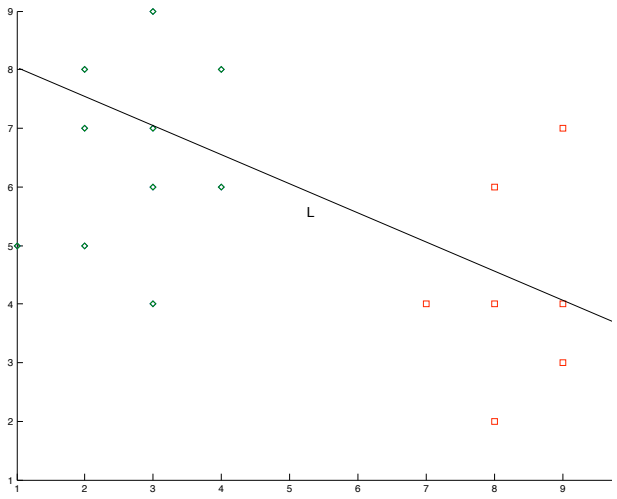

Figure 6.1.: Example of a discriminant function $L = c + b_1x_1 + b_2x_2$ when $\mathbf{x} = (x_1, x_2)$

The statistical analysis tool SPSS is used to compute Fisher's discriminant function for each of the groups $\Pi_1, ..., \Pi_g$, based on a data set that includes observations and information on the group to which the case belongs. This results in the discriminant weights $b_1, ..., b_k$ for the corresponding input variables.

Once the linear discriminant function has been calculated, an observation $\mathbf{x}$ is allocated

to a group based on its discriminant score L_i. The discriminant scores are calculated by substituting the observations $(x_1, x_2, ..., x_k)$ into each of the discriminant functions. The group that achieves the highest discriminant score with respect to $\mathbf{x}$ has the highest probability to be the group to which $\mathbf{x}$ belongs.

How these functions can be applied to predict user satisfaction will be described in the next subsection.

6.1.2. The prediction method

The method to predict user satisfaction is based on discriminant analysis, as described above. The input of the discriminant functions are the observed values for Mean Absolute Error, Area Under ROC, Precision, Channel Level Diversity, Genre Level Diversity, Series Level Diversity and Series Level Serendipity. These metrics happen to have the strongest correlation to user satisfaction, as described in Chapter 5, and can be measured without explicit user intervention. SPSS was used to determine the discriminant weights. The weights for each class (rows) and metric (columns) that were found for the various dimensions can be represented as a matrix, as shown next:

$$B = \begin{bmatrix} 7.701 & 0.241 & 13.878 & 26.448 & 17.730 & 2.825 & 0 \\ 5.045 & -0.980 & 14.526 & 29.594 & 20.600 & 2.296 & 0 \\ 4.994 & -2.057 & 17.834 & 29.576 & 21.737 & 4.868 & 0 \\ 5.011 & -1.375 & 19.419 & 28.990 & 21.660 & 7.983 & 0 \\ 4.684 & -0.333 & 19.524 & 27.532 & 18.312 & 12.206 & 0 \end{bmatrix}$$

The matrix' columns contain the weights for Mean Absolute Error, Area Under ROC, Precision, Channel Level Diversity, Genre Level Diversity, Series Level Diversity and Series Level Serendipity respectively. The last column only contains zero values, indicating that Series Level Serendipity is not a discriminating variable (even though there exists a correlation between this metric and user satisfaction). Exactly why Series Level Serendipity does not add to user satisfaction when it is combined with the other metrics requires further investigation, but it might be possible that its effect is cancelled out by the remaining metrics. Since the weights for Series Level Serendipity are all zero, it can be left out of the equation. The constant values $c_1, ..., c_5$ of Fisher's linear discriminant functions are also provided by SPSS:

$$C = \begin{bmatrix} -24.222 \\ -20.808 \\ -24.265 \\ -28.020 \\ -30.859 \end{bmatrix}$$

The discriminant scores $d_1, ..., d_5$ can now be computed as shown in Equation 6.1. In this equation $x_1, ..., x_7$ are the observed metric values for Mean Absolute Error, Area Under ROC, Precision, Channel Level Diversity, Genre Level Diversity, Series Level Diversity and Series Level Serendipity respectively.

$$\begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_5 \end{bmatrix} = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,7} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,7} \\ \vdots & \vdots & \ddots & \vdots \\ b_{5,1} & b_{5,2} & \cdots & b_{5,7} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_7 \end{bmatrix} + \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_5 \end{bmatrix} \tag{6.1}$$

As explained earlier, the predicted value for the user satisfaction, based upon the observation $\mathbf{x}$ is i when d_i is the highest discriminant score. The overall prediction method can therefore be formally defined as shown in Equation 6.2.

$$c(\mathbf{x}) = \{i \in [1,5] : d_i \geq d_j \wedge i \neq j \ \forall i,j\} \tag{6.2}$$

In the equation d_i is the result of Equation 6.1.

6.2. Validating the model

In order to verify the validity of the model its classification accuracy must be tested. When creating the model SPSS already tests the classification accuracy using the data on which the model is fitted. SPSS applies the model it creates to predict for each item to which class it belongs. These predictions are then compared to the actual classes. This method of evaluation is not really righteous, since the model is evaluated on the same data that was used to construct it and thus it could have known how to "predict" the user satisfaction.

The results of the evaluation as performed by SPPS during the model creation is shown in Table 6.1. The table displays for each class how many of its items were allocated to each of the other classes.

User Satisfaction	**Predicted**				
Actual	**1**	**2**	**3**	**4**	**5**
1	365 (77.2%)	75 (15.9%)	28 (5.9%)	5 (1.1%)	0 (0.0%)
2	42 (12.2%)	153 (44.6%)	110 (32.1%)	36 (10.5%)	2 (0.6%)
3	16 (3.9%)	86 (21.1%)	181 (44.4%)	120 (29.4%)	5 (1.2%)
4	3 (0.9%)	34 (9.7%)	99 (28.3%)	197 (56.3%)	17 (4.9%)
5	0 (0.0%)	7 (5.7%)	15 (12.3%)	73 (59.8%)	27 (22.1%)

Table 6.1.: Classification results

The table shows for example that 365 (*e.g.* 77.2%) of the recommendation lists that were rated with one star were also predicted to be rated with one star. Furthermore is shows that 120 lists that were predicted to have been rated with 4 four stars were in fact rated with three stars. The table also shows that 923 out of 1696 entries are correctly classified. This means that the model is able to predict 54.42% of the users ratings for user satisfaction correctly (as opposed to 20%, which would be achieved when the ratings were randomly predicted). However, Table 6.1 also shows that when the model misclassifies a list, the error it makes is usually small. Therefore it is interesting to measure to what extent the model can predict the user satisfaction instead of how many predictions are exactly the same as the true user's

rating.

The predictive accuracy (as opposed to classification accuracy) can be determined by calculating the mean absolute error over the predicted and true user's rating. To measure the predictive accuracy a validation methodology called cross validation is applied, since the generalisation ability of the model is important [Alp04]. The following subsection will describe K-fold cross validation in more detail.

6.2.1. K-fold cross validation

The classification evaluation as implemented by SPSS is performed on the same data set that is used to construct the model. It is therefore possible that the created model is able to predict the user satisfaction really accurately for the data on which the model is trained, while it performs much worse on yet-unseen data. To overcome this problem, cross validation can be applied.

With cross validation a part of the data set is retained when creating the model so it can be used to evaluate it later on. The data that is used to test the model is called the validation set, while the data that is used to create the model is called the training set [Alp04]. Both sets do not overlap (*i.e.* the sets are disjoint). The part of the data that is used for training and the part that is used for validation can be varied so that multiple validations can be performed.

In K-fold cross validation the data set is divided randomly into K (almost) equal-sized non-overlapping parts $X_1, ..., X_K$, called the folds. These folds are then combined into K training / validation pairs $(T_1, V_1), ..., (T_K, V_K)$ where:

$$\begin{aligned} V_1 &= X_1 & T_1 &= X_2 \cup X_3 \cup \cdots \cup X_K \\ V_2 &= X_2 & T_2 &= X_1 \cup X_3 \cup \cdots \cup X_K \\ &\vdots & &\vdots \\ V_K &= X_K & T_K &= X_1 \cup X_2 \cup \cdots \cup X_{K-1} \end{aligned}$$

For each of these pairs the model is trained on the training set and validated on the validation set. The expected performance of the model trained on all data for data that is yet unknown to the model is then said to be the mean of the K validity scores. Since the mean absolute error is used as the metric for validity the overall performance of the model is calculated as shown in Equation 6.3.

$$MAE = \frac{1}{K} \sum_{j=1}^{K} \left(\frac{1}{|V_j|} \sum_{x \in V_j} |r(x) - p_j(x)| \right) \tag{6.3}$$

In this equation the prediction for the user satisfaction for recommendation list x, denoted as $p_j(x)$ is based on the model created on training set T_j. The true user's rating for the recommendation list is denoted as $r(x)$.

6.2.2. Evaluation results

In the validation of the predictive model created in this research K is chosen to be 4 since the cost of creating the training / validation sets, training the model and the determination of the mean absolute error becomes really high for large values of K. These exorbitant costs were primarily caused by the fact that the datasets had to be split using MS Excel, the model was created using SPSS and the calculation of the accuracy using a custom build Java tool. The results of each of these steps had to be transfered to the next step by hand. Unfortunately automating the task would be even more time consuming.

With 4-fold cross validation, the data set is thus randomly subdivided into 4 subsets. On each of these subsets a model is validated that was created based on the other three subsets. The results of this 4-fold cross validation are shown in Table 6.2.

		Error size				
V	**MAE**	**#0**	**#1**	**#2**	**#3**	**#4**
1	0.75887	335 (41.0)	359 (43.9)	108 (13.2)	15 (1.8)	0 (0.0)
2	0.72093	356 (43.6)	344 (42.1)	107 (13.1)	9 (1.1)	1 (0.1)
3	0.71603	360 (44.0)	339 (41.5)	108 (13.2)	10 (1.2)	0 (0.0)
4	0.76103	338 (41.4)	353 (43.3)	107 (13.1)	18 (2.2)	0 (0.0)
Avg.	0.73922	347 (42.5)	349 (42.7)	108 (13.2)	13 (1.6)	0 (0.0)

Table 6.2.: K-fold cross validation of the prediction model

The table lists for each fold the mean absolute error and the number of times an error of size 0 to 4 occurred. The value between parentheses is the corresponding percentage of occurrences. The last row contains the average for each of the columns.

Table 6.2 shows that the prediction model on average achieves a mean absolute error of 0.73922 on data that is unknown to the model. This implies that the user satisfaction can be quite accurately predicted since the prediction method is less than one point of the mark on average.

6.3. Discussion

The attentive reader might notice that discriminant analysis was employed to create the predictive model, even though the data appeared not to be distributed normally. This is a correct observation, since discriminant analysis assumes a multivariate normal distribution. Discriminant analysis is said to be relatively robust even when there is a modest violation of this assumption [Lac75, Kle80]. This claim indeed appears to be true, since the validation of the model (as described in section 6.2) shows that the model created using discriminant analysis is quite accurate.

Another critical note might be placed by the choice of K in the K-fold cross validation. Usually K is chosen to be somewhere between 10 and 30, but since the cost of creating the training / validation sets, training the model and the determination of the mean absolute error was really high in the validation setup, K was chosen to be 4. This choice can be

justified by the fact that the observed mean absolute error differs only slightly for each of the folds.

7

Conclusions and Future Work

The main objective of this research was to determine how user satisfaction achieved by a recommender system recommending TV programmes could be predicted without requiring user participation. This chapter describes how this objective was met and presents further conclusions drawn from the research (section 7.1). Then, in section 7.2, some hints to potential future work as a result of the research are presented.

7.1. Conclusions

During this research the following research question was answered; How can the user satisfaction achieved by an EPG/RS system be predicted based on various off-line metrics? To achieve structured completion of the overall research a number of sub questions was answered. These sub questions are:

- What recommendation techniques do exist, how do they work and what are their benefits?
- What is the state-of-the-art in recommender system evaluation and what aspects are important to evaluate?
- What is user satisfaction and how can it be measured?
- How do the individual properties of a recommendation list and user satisfaction cohere?

The following paragraphs will answer these questions.

Three different prediction techniques can be distinguished in the literature: collaborative filtering, content-based filtering and hybrid forms that combine the other two techniques. One of the main issues of collaborative filtering is that it suffers from the cold start problem. Content-based filtering on the other hand suffers from overspecialisation. Hybrid prediction techniques try to overcome these problems by avoiding the limitations of each technique.

Furthermore, literature describes seven different dimensions that can be used to evaluate a recommender system: accuracy, coverage, confidence, diversity, learning rate, novelty and serendipity. Accuracy is by far the most widely used when evaluating recommender systems, although research has shown that it is not equivalent to user satisfaction.

An evaluation application to measure user satisfaction was implemented that enabled users to rate recommendation lists on a five-point Likert scale. This user study, collecting 9762 ratings of 70 users over a period of 3 months, showed that the accuracy metrics mean absolute error, precision and area under ROC have the strongest correlation with user satisfaction. Ranking accuracy metrics such as relative edit distance and Spearman's rank correlation appeared to have a weaker correlation, possibly because the order of the recommended items is less important. Other metrics that proved to have a strong correlation to user satisfaction are user's diversity, series level diversity, user's serendipity and effective overlap ratio.

Clustering the users into four clusters, based on the individual correlations between 15 different metrics and user satisfaction showed that the users in each of these clusters value distinct list properties. Users in one of the cluster were similar with respect to diversity, while two other clusters contained users that were alike regarding accuracy and serendipity respectively. The fourth cluster contained users that did not discriminate themselves with respect to any of the dimensions. The clustering proved to be valid since the within-cluster correlation coefficients showed that the correlations with the metrics that best described the clusters were indeed stronger.

The prediction method developed in this research utilises discriminant analysis using Fisher's discriminant functions. This technique was used to construct a model that classifies the observed values for Mean Absolute Error, Area Under ROC, Precision, Channel Level Diversity, Genre Level Diversity, Series Level Diversity and Series Level Serendipity into the classes "rated with 1 star", "rated with 2 stars", ..., "rated with 5 stars". The resulting model is shown in Equation 7.1.

$$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \end{bmatrix} = \begin{bmatrix} 7.701 & 0.241 & 13.878 & 26.448 & 17.730 & 2.825 \\ 5.045 & -0.980 & 14.526 & 29.594 & 20.600 & 2.296 \\ 4.994 & -2.057 & 17.834 & 29.576 & 21.737 & 4.868 \\ 5.011 & -1.375 & 19.419 & 28.990 & 21.660 & 7.983 \\ 4.684 & -0.333 & 19.524 & 27.532 & 18.312 & 12.206 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} + \begin{bmatrix} -24.222 \\ -20.808 \\ -24.265 \\ -28.020 \\ -30.859 \end{bmatrix} \tag{7.1}$$

The predicted user satisfaction for a recommendation list with observed metrics $x_1, ...x_6$ is i where d_i is the highest discriminant score.

The application of 4-fold cross validation showed that the prediction method on average achieves a mean absolute error of 0.73922 on data that is unknown to the model. This denotes that the user satisfaction achieved by a recommender system recommending TV programmes can be accurately predicted using the method developed in this research.

7.2. Future work

During the research many potentially interesting and relevant research subjects presented themselves, but to keep focused on the objectives of this research these topics had to be discarded. This section briefly discusses these subjects for possible future work.

First of all, more research into the goals of users of an EPG/RS systems need to be performed. Why do people use such a system? Is it because they want to discover content that is new to them? Do they want to spend less time in finding their favourite TV shows? Or are there other reasons for using an EPG/RS? Closely related to these goals are the tasks that users try to address when using an EPG/RS. Do they want to find *all* interesting programmes or are users satisfied when the system recommends 10 sufficiently interesting TV shows? Do users expect the recommender to prove them to be credible by recommending TV programmes that are known to them (and considered to be interesting)? It is important to understand these goals and tasks when evaluating a recommender system and even though the questionnaire addressed this subject more (ethnographic) research into it would be valuable.

Secondly, it is conceivable that more dimensions that add to user satisfaction can be identified. Although the questionnaire did not reveal any missing dimensions that play a role in the perceived user satisfaction, a more extensive research might uncover dimensions that also appear to be important. These dimensions might help to further improve the prediction method.

Another question that needs more extensive research is whether the prediction method developed in this research is also applicable to other domains. It is possible that users value other properties of the recommendation list when it contains recommendations for books or movies for example. Furthermore the prediction method needs a value for the Channel Level Diversity of the recommendation list. This metric is EPG specific, but maybe it can be left out of the equation without penalising the performance too much.

Fourth, the performance of the prediction method might be further improved when a large(r)-scale user study is performed with a more diverse user base. Finding sufficient people that wanted to participate in the user study was a tough job and therefore demanding a wide variety of users was a luxury that could not be afforded. When the user experiment could be repeated as part of an existing EPG/RS system with tens of thousands of users the resulting data could be used to develop a predictive model that is even more accurate.

Furthermore, it might be interesting to investigate whether the people that were assigned to the same cluster (as described in Section 5.4) share common demographic information such as age, gender and level of education. When this appears to be the case, it becomes possible to determine what dimensions the user would value in advance. The recommender system can then tailor its recommendations without letting its users rate the recommendation lists.

Finally, a prediction method that is able to produce a more fine grained prediction could be developed. The method developed in this research is able to predict the user satisfaction on a five-point scale, but conceivably it is also possible to refine this scale. Maybe it is even possible to predict the user satisfaction on a continuous scale.

A

Existing recommender systems

Recommender systems have changed from novelties used by a few researchers to serious business tools that are becoming widely used in various domains. Recommender systems are especially popular for recommending books, music, movies and TV programs among others. In this appendix some e-commerce sites and services using recommender systems are discussed.

A.1. Amazon.com

Probably the best known application of recommender system technology is Amazon.com. Users of Amazon.com receive recommendations in various forms. An example is the "Customers who bought" feature that recommends items that were bought by people who bought a particular item (as shown in Figure A.1).

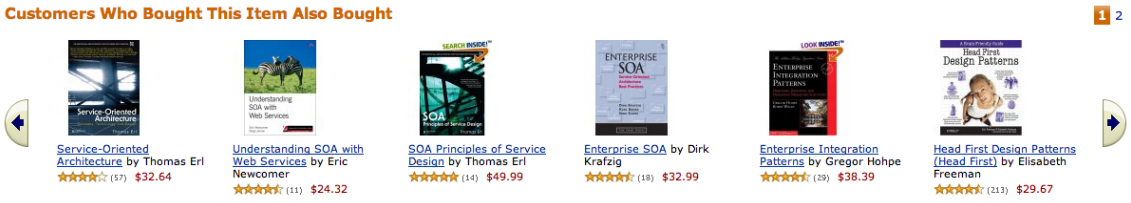

Figure A.1.: Amazon's "Customers who bought" feature

Users that have an Amazon.com account and are signed-in also receive a number of recommendations, based on their buying history (as shown in Figure A.2). The user can specify from which category he likes to receive recommendations, by clicking on it.

Amazon.com uses item-based collaborative filtering (CF) to predict what items you probably like best from the enormous set of potentially interesting items [LSY03]. Scalability is an important issue here, since Amazon.com has tens of millions of customers and products. The

Figure A.2.: Amazon's "Today's recommendations for you"

item-based collaborative filtering algorithm matches each of the user's purchased and rated items to similar items and combines those items into a recommendation list for the user. To determine what items are similar, the algorithm builds a similar-items table by finding items that customers tend to purchase together. This approach is different from most other recommender algorithms that calculate a similarity metric for each pair of items.

Amazon's algorithm can be described as follows:

```
For each item in product catalog, I1
    For each customer C who purchased I1
        For each item I2 purchased by customer C
            Record that a customer purchased I1 and I2
    For each item I2
        Compute the similarity between I1 and I2
```

The similarity between two products is computed as the angle between two M dimensional vectors P_1 and P_2, where each dimension corresponds to a user who has bought that item or not. The similarity is calculated as follows:

$$similarity(\overrightarrow{P_1}, \overrightarrow{P_2}) = \cos(\overrightarrow{P_1}, \overrightarrow{P_2}) = \frac{\overrightarrow{P_1} \bullet \overrightarrow{P_1}}{|\overrightarrow{P_1}| \cdot |\overrightarrow{P_1}|}$$

Given the similar-items table, the algorithm quickly finds items similar to each of the user's purchases and ratings. These items are then aggregated and the most popular or correlated items are actually recommended.

A.2. TiVo

Another application of recommender systems can be found in TiVo. TiVo is a combination of a digital video recorder (DVR) and a television-viewing service that is really popular in the United States. The DVR is able to record TV shows on its hard-disk in the form of MPEG streams, which can be viewed later on. Users can also pause and rewind live TV (called "time shifting") as well as recorded shows. The TiVo service provides a range of features, such as

a smart electronic program guide (EPG), recommender system (RS), home movie sharing, on-line scheduling, access to Internet radio, podcasts, weather and traffic information, et cetera.

TiVo's main benefits are that it allows viewers to watch shows at times convenient to them and facilitates means that enable users to find the shows they like in the proliferation of hundreds of TV channels.

TiVo's recommender system, called TiVo Suggestions helps users to find shows they probably like. TiVo Suggestions uses a combination of item-based collaborative filtering and Bayesian content based filtering (CBF) techniques to make recommendations. Collaborative filtering techniques base their recommendations on like-minded users (*i.e.* users who like similar programs as the user requesting the recommendations). Content-based recommenders, however, use features from the items themselves (like genre, cast, age of the show, etc.) valued by the user, to generate recommendations [AvS04].

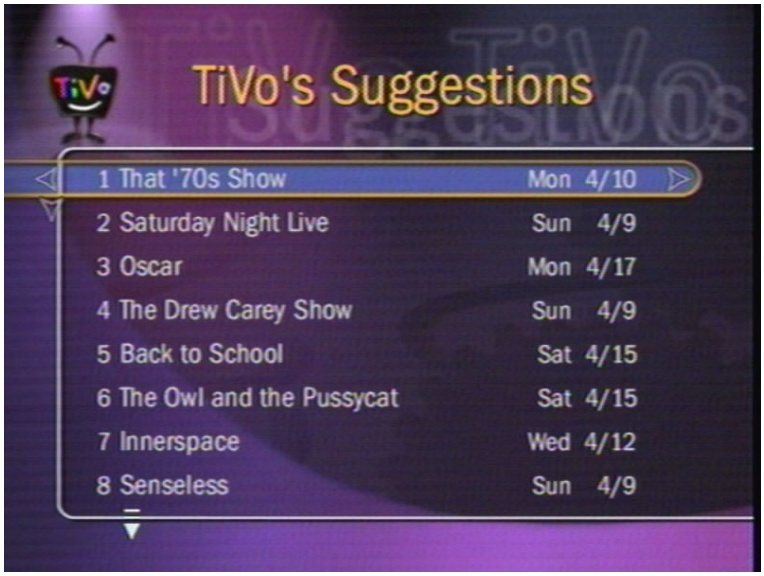

Figure A.3.: TiVo's suggestions

The process of recommendation starts with feedback provided by the user. This feedback can be either explicit or implicit. Explicit feedback in TiVo can be provided by using the "thumbs up" or "thumbs down" button. Explicit ratings range from +3 to -3. Implicit ratings are ratings derived from a user's behaviour (*i.e.* the portion of a show viewed, the show is recorded, etc.). TiVo only uses the fact that a user recorded a show as implicit rating. The user's feedback is used to build a profile of likes and dislikes.

The user's profile is periodically transmitted to TiVo's server cluster. The anonymized profiles are not permanently stored on the server due to privacy concerns and are only kept at the server during the calculation of recommendations.

Then the item-item correlations are computed at TiVo's servers. Since TiVo is based on an item-based collaborative filtering algorithm, it determines the similarity between items instead of users. The underlying idea is that items that are rated the same way in the past,

are probably similar. TiVo prepares a package of correlation pairs that are then sent to the set-top box.

Using the downloaded correlation package, TiVo predicts the user's ratings for all upcoming shows. These predictions are then combined with content-based predictions, calculated by the set-top box. The application of a content-based filter algorithm helps to overcome the "cold-start" problem (the situation in which the user did not provide enough feedback to calculate predictions). The suggestion list is constructed by combining the highest rated predictions and shows already rated by the user. The show that is on top of the suggestion list can even be recorded automatically by the DVR.

A.3. MeeVee

MeeVee is an on-line TV guide that can provide users with a personalised TV listing. The personalisation is based on a list of interests that a user can compose. Interests are not limited to TV shows and also include, among others, people, topics & hobbies, movies and sports. Once a user has composed a list of interests, MeeVee is not only able to construct a personal TV listing, but can also recommend on-line video content from hundreds of sources. The techniques that are used by MeeVee are not described, but their websites states that it is patented.

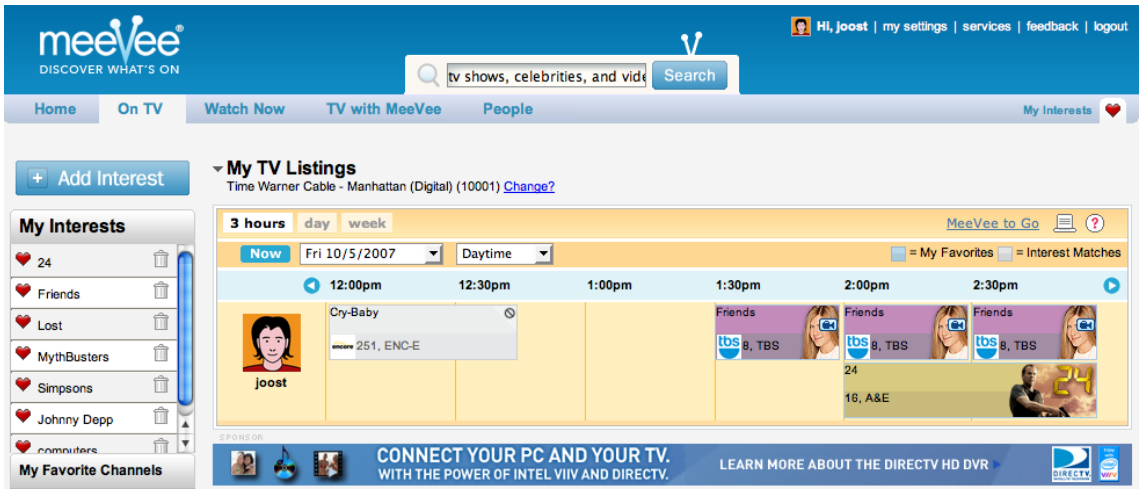

Figure A.4.: MeeVee's TV listing

A.4. Zie.nl

Zie.nl is an on-line programme guide targeted at The Netherlands. One of Zie.nl's features is a personalised cloud of TV programmes and web video snippets (as shown in Figure A.5). The cloud is personalised based on feedback on the programmes / snippets as provided by the user. To conquer the cold-start problem users that register themselves to Zie.nl are asked to provide feedback on popular Dutch actors and TV shows during the registration procedure.

Figure A.5.: TV programme recommendations in Zie.nl

A.5. Other recommender systems

Besides the applications described above there are numerous other websites and applications that employ recommender system technologies. Some of them are listed next.

- Yahoo! Movies, a movie recommendation service
- eBay, an on-line auction website
- NetFlix, an on-line movie rental service
- iTunes, Apple's audio and video player including the iTunes store
- CDNOW.com
- MovieLens, a recommender system that provides movie recommendations
- Jester, a joke recommender system
- Last.fm
- MyStands
- Pandora
- StumbleUpon

A.6. Open-source software libraries

Various open-source software libraries exist that provide developers with means to add recommendation features to applications or web sites. The following software libraries do exists:

- Duine (http://sourceforge.net/projects/duine)
- Taste (http://taste.sourceforge.net)
- Cofi (http://www.nongnu.org/cofi)
- CoFE (http://eecs.oregonstate.edu/iis/CoFE)
- Colfi (http://colfi.wz.cz)
- RACOFI (http://www.daniel-lemire.com/fr/abstracts/COLA2003.html)
- SUGGEST (http://glaros.dtc.umn.edu/gkhome/suggest/overview)
- Vogoo (http://www.vogoo-api.com)
- Consensus (http://exogen.case.edu/projects/consensus)

B

Recommender systems expertise

The design and implementation of recommender systems requires expert knowledge. This knowledge is not widespread and only few people or organisations have expertise in this field. This appendix describes some of them.

B.1. Telematica instituut

The Telematica Instituut is a Dutch public-private partnership, managed by a consortium of companies. The institute is involved in the "MultimediaN N9 Media & I-services" project that strives to develop high-quality multimedia solutions. Personalisation of multimedia content guides is one of the research areas in this project[1]. Project manager of this research programme is Dr. Mark van Setten, who obtained his doctorate with a thesis on recommender systems [vS05]. Within the Gigaport and MultimediaN research programmes, Mark developed a toolkit for building recommender systems. This toolkit, called Duine, allows several prediction techniques to be combined, which greatly increases the accuracy of recommendations. Furthermore it allows developers to add their own recommendation techniques. The toolkit has been released as an 'open source' product on SourceForge[2] in 2006.

B.2. Stoneroos

Stoneroos Interactieve Televisie is a cross platform entertainment company that designs and develops games, TV formats, applications and business solutions for broadcasting corporations and network providers. Stoneroos is involved in a project called Passepartout, together with researchers and students of the Technical University of Eindhoven (TU/e) and companies such as Philips. Within this project Stoneroos developed an electronic program guide (EPG) with recommendation functionality (see Figure B.1). One of the contributors, Dr. Lora Aroyo,

[1] http://www.telin.nl/index.cfm?language=en&id=1060&context=1063
[2] http://sourceforge.net/projects/duine

who currently holds a research position at the Vrije Universiteit Amsterdam, is an expert in the field of recommender systems. She and the other researchers of TU/e developed a framework, called SenSee, that enables easy searching in TV content. SenSee's is part of iFanzy, the EPG system that Stoneroos tends to bring to the market. iFanzy can be fully adapted to the preferences of its users.

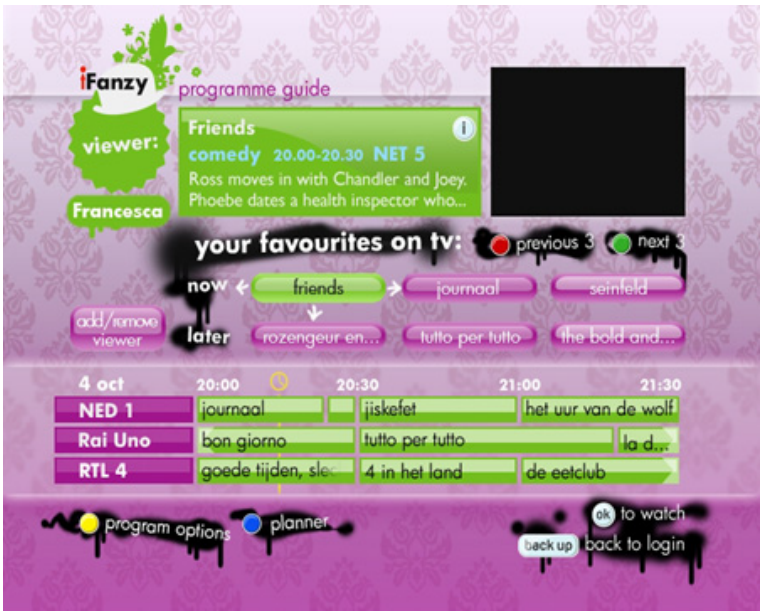

Figure B.1.: Screenshot of StoonRoos' iFanzy

B.3. Fraunhofer

The Media Interoperability Lab of Fraunhofer FOKUS [3] is a research group with expertise in the areas of interactive applications, media handling, and mobile telecommunications. This expertise is used to develop "Next Generation Media Delivery Platforms featuring live, on demand, context-aware, and personalized interactive media empowered by the Open IMS Playground". The Media Interoperability Lab's R&D activities are focused on "Metadata management & Personal Content Management" and "Movie Recommendation" among others.

Fraunhofer FOKUS has developed a recommender toolkit as stated on their website: "The Recommender Toolkit is a personalization framework developed at the Fraunhofer FOKUS. It offers high adaptability for multiple services and varying personalization approaches. To enable real-time recommendations, while still having the best possible recommendation quality, the learning algorithm works offline. Both content-based algorithms -which consider one users viewing behavior- as well as collaborative algorithms -which use community based approaches- can be used and exchanged while the system is running."[4] No further information is provided by the institute.

[3] http://www.fokus.fraunhofer.de/bereichsseiten/testbeds/nextgenmedialab/
[4] http://www.fokus.fraunhofer.de/testbeds/nextgenmedialab/IPTVAS.php

B.4. ChoiceStream

ChoiceStream[5] is a personalisation company located in Cambridge, Massachusetts. ChoiceStream offers business solutions in three different areas: online services, TV providers and mobile operators. Their products and services are used by companies such as Yahoo! (Yahoo! Movies), AOL, DirectTV, Overstock.com, MovieLink, etc. One of ChoiceStream's products, called RealRelevance Video Suite, includes TV planner functionality that can construct personalised TV listings. The algorithms and technologies used by ChoiceStream are not described on their website.

B.5. Baynote

Baynote[6] is a start up company, located in Cupertino, California, USA. The company delivers on-demand recommendation technology for websites and provides recommendation solutions for eCommerce, media, marketing and support activities. Baynote's recommender system does not collect explicit feedback and solely relies on implicit feedback/information gathered from pages visited by the user. Recommendations are delivered as a web service. The web service can provide product and content recommendations, as well as a social search results.

B.6. Minekey

Minekey[7] is another start up, located in Sillicon Valley that provides a recommendation service. Minekey's recommendation service gathers implicit user information through their surfing behaviour. The recommendation engine acquires click stream data and combines this with metadata of the content to extract the "themes of interest" for a specific user. Moreover it uses some form of collaborative filtering to derive meaningful associations and relationships amongst the various "themes of interest." These relations are then used to make recommendations to users. As a proof of concept, Minekey provides widgets that discover interesting content for users.

B.7. Other commercial companies

Other companies that deliver personalisation related products are:

- Loomia (http://www.loomia.com)
- Criteo (http://www.criteo.com)
- SourceLight (http://www.sourcelight.com)
- Collarity (http://www.collarity.com)
- MediaRiver (http://www.mediariver.com)

[5] http://www.choicestream.com
[6] http://www.baynote.com
[7] http://www.minekey.com

B.8. Universities

A lot of recommender system expertise is concentrated in universities around the world. Probably the most active research group in this field is GroupLens [8]. GroupLens consists of faculty and student researchers affiliated to the Department of Computer Science and Engineering at the University of Minnesota. They published a large number of scientific publications over the past few years, that had a great impact in the research community. Some of the researchers left the group, however, and now they start to concentrate on other topics (like online communities, mobile and ubiquitous technologies, digital libraries and local geographic information systems) as well.

In The Netherlands recommender system technology is researched at Delft University of Technology, Vrije Universiteit Amsterdam and University of Twente.

[8] http://www.grouplens.org

C

Mathematical notations

This appendix gives an overview of the mathematical notations used throughout the report.

- A = the set of users $\{a_0, \ldots, a_n\}$
- B = the set of items $\{b_0, \ldots, b_m\}$
- $r_i(b_k)$ = rating of user a_i for item b_k
- $p_i(b_k)$ = predicted rating of user a_i for item b_k
- $c_i(b_k)$ = confidence of the prediction $p_i(b_k)$
- R = the rating matrix $A \times B$ consisting of ratings $r_i(b_k)$ of all users $a_i \in A$ for items $b_k \in B$
- $B_i = \{b_k \in B : r_i(b_k) \neq \perp\}$ (*i.e.* all items rated by user a_i)
- $B_{ij} = \{b_k \in B : r_i(b_k) \neq \perp \wedge r_j(b_k) \neq \perp\}$ (*i.e.* all items rated by both users a_i and a_j)
- B^c = the set of items $\{b_0, \ldots, b_n\}$ contained in a certain cluster c (all TV programmes from a certain genre for example).
- $B_i^c = \{b_k \in B^c : r_i(b_k) \neq \perp\}$ (*i.e.* all items in a certain cluster c rated by user a_i)
- $A_k = \{a_i \in A : r_i(b_k) \neq \perp\}$ (*i.e.* all users who rated item b_k)
- $A_{kl} = \{a_i \in A : r_i(b_k) \neq \perp \wedge r_i(b_l) \neq \perp\}$ (*i.e.* all users that rated both items b_k and b_l)
- $\widetilde{A}_i$ = the neighborhood of l users $a_j \in A$ that are most similar to the active user a_i
- $|X|$ = the number of items in set X
- R_{a_i} = the ordered sequence $[b_1, ..., b_N]$ of items $b_k \in B$ rated by user a_i, sorted on ratings $r_i(b_k)$

- L_{p_i} = the ordered sequence $[b_1, ..., b_N]$ of items $b_k \in B$ representing the top-N recommendation list for user a_i
- η = sparsity coefficient of a ratings set
- σ = normalization factor

D

Questionnaire Results

After the user study was completed the participants of this study were asked to complete a short questionnaire of ten open-ended and closed-ended questions. The purpose of the questionnaire was to learn from the participants, who are likely to be experienced users of a TV programme recommender system by now. The results of this questionnaire are presented in this appendix.

Question 1: During the research a number of aspects were investigated. Give for each of the aspects an indication of its importance to the quality of the recommendation list.

Dimension	**1**	**2**	**3**	**4**	**5**	**Avg**
Accuracy	2	4	13	27	19	3.88
Ranking accuracy	9	18	21	15	2	2.74
Programme overlap	11	16	19	17	2	2.74
Viewability	5	5	16	28	11	3.54
Diversity	0	10	12	33	10	3.66
Serendipity	2	15	19	19	9	3.28

Table D.1.: Questionnaire results for question 1

Question 2: When do you experience a recommendation list as divers?

Answer	#
When the list contains many different TV shows	18
When the programmes in the list differ with respect to genre	61
Other	2

Table D.2.: Questionnaire results for question 2

Other definitions mentioned by the respondents:

- When the TV programmes are broadcasted by different TV corporations

Question 3: When do you experience a recommendation list as surprising?

Answer	#
When I have not seen most of the recommended programmes before	37
When I would not have discovered the recommended programmes myself	47
When the list fitted extremely well to my expectations	17
When the list fitted to my expectations really poorly	18
When the combination of the recommendations was nice	13
Other	1

Table D.3.: Questionnaire results for question 3

Question 4: Are there any aspects that you think are important to the quality of the recommendation list that are not listed in question 1?

Answer	#
No	59
Yes	6

Table D.4.: Questionnaire results for question 4

Aspects mentioned by the respondents are:

- No reruns of TV shows
- No irrelevant TV programmes
- Viewability
- Diversity with respect to channels
- Known / unknown ratio

Question 5: During phase one of the user study:

Answer	#
I tried to provide feedback on every TV programme, even though I had not seen it	11
I provided feedback on those series/programmes I watch regularly	32
I only provided feedback on TV programmes I actually watched	21

Table D.5.: Questionnaire results for question 5

Question 6: During phase two of the user study:

Answer	#
I quitted participating in the user study	14
I tried to provide feedback on every TV programme, even though I had not seen it	11
I provided feedback on those series/programmes I watch regularly	30
I only provided feedback on TV programmes I actually watched	10

Table D.6.: Questionnaire results for question 6

Question 7: How did you decide to participate in the user study?

Answer	#
The research seemed interesting to me	38
I wanted to win the DVD-box	9
I wanted to help Joost with his graduation	60
I liked to receive recommendations for TV programmes	11
Other	3

Table D.7.: Questionnaire results for question 7

Other motivations mentioned by the respondents:

- I think the problem is relevant
- Djoerd asked me to participate

Question 8: What would be a motivation for you for using a recommender system that recommends TV programmes?

Answer	#
There are too many TV programmes and channels, I suffer from information overload	24
I do not want to spend time finding interesting TV programmes	10
I need suggestions on what to watch since I do not know it myself	7
I would like to discover TV programmes that I am unfamiliar with	41
Other	12

Table D.8.: Questionnaire results for question 8

Other motivations mentioned by the respondents:

- I do not want to miss programmes I would be interested in
- To discover new TV programmes besides the shows I regularly watch

Question 9: Please specify to what extent you agree with the following theses.

Dimension	**1**	**2**	**3**	**4**	**5**	**Avg**
I would use an on-line service that recommends TV programmes	7	8	21	21	8	3.23
I would use an altered version of Utente	9	17	15	19	5	2.91
I would like to receive more than 10 recommendations	27	19	9	7	2	2.03

Table D.9.: Questionnaire results for question 9

Summary

Recommender systems are systems that help people to cope with the ever increasing amount of potentially interesting content. The enormous amount of available content makes it impossible to go through all the items yourself to decide whether it is useful or not and thus finding relevant items becomes more and more difficult. This problem has also become relevant for people watching television, since the introduction of digital television induced a further growth of available TV channels, and consequently a growing number of available TV shows.

Recommender systems use knowledge about a user's preferences (and those of others) to recommend them items that they are likely to enjoy. To achieve this three different prediction strategies can be utilised. The first, called content-based filtering tries to match the set of item characteristics and the set of user preferences. The more an item's characteristics match the user's preferences the better he will like the item. The second prediction strategy, called collaborative filtering, bases its predictions on ratings that other users have assigned to items. Based on these ratings either similar users or similar items can be identified, which are then used to predict a user's rating for a particular item. The third technique combines both content-based filtering and collaborative filtering and is therefore called a hybrid prediction technique.

Nowadays, many different recommender systems exist such as Amazon.com, Netflix, Last.fm, MyStrands and iTunes for example. This fosters the need for means to evaluate and compare different systems. However, recommender system evaluation has proven to be challenging since a recommender system's performance depends on, and is influenced by many factors. The data set on which a recommender system operates for example has great influence on its performance. Furthermore, the goal for which a system is evaluated may differ and therefore require different evaluation approaches. Another issue is that the quality of a system recorded by the evaluation is only a snapshot in time since it may change gradually.

Despite these challenges there exist many ways to evaluate recommender systems and research is still performed in this area. Although there exists no consensus among research on what recommender system's attributes to evaluate, accuracy is by far the most popular dimension to measure. However, some researchers believe that user satisfaction is the most important quality attribute of a recommender and that greater user satisfaction is not achieved by an ever increasing accuracy. Other dimensions for recommender system evaluation that are described in literature are coverage, confidence, diversity, learning rate, novelty and serendipity. It is believed that these dimensions contribute in some way to the user satisfaction achieved by a recommender system.

In order to investigate how these dimensions cohere with respect to user satisfaction in an electronic programme guide (EPG) context, a user study is performed. During this user study

133 people subscribed to an evaluation application specially designed and build for this purpose. The user study consisted of two phases. During the first phase users had to rate TV programmes they were familiar with or that they recently watched. This phase resulted in 36.353 programme ratings for 7.844 TV programmes. Based on this data, the recommender system that was part of the evaluation application could start generating recommendations. In phase two of the study the application displayed recommendations for tonight's TV programmes to its users. These recommendation lists were deliberately varied with respect to the accuracy, diversity, novelty and serendipity dimensions. Another dimension that was altered was programme overlap. Users were asked to provide feedback on how satisfied they were with the list. Over a period of four weeks 70 users provided 9762 ratings for the recommendation lists.

For each of the recommendation lists that were rated in the second phase of the user study, the five dimensions (accuracy, diversity, novelty and serendipity) were measured using 15 different metrics. For each of these metrics its correlation with user satisfaction was determined using Spearman's rank correlation. These correlation coefficients indicate whether there exists a relation between that metric and user satisfaction and how strong this relation is. It appeared that accuracy is indeed an important dimension in relation to user satisfaction. The accuracy metrics mean absolute error, precision and area under ROC were among the 8 strongest correlations that were found. Other metrics that had a strong correlation were user's diversity, series level diversity, user's serendipity and effective overlap ratio. This indicated that diversity, serendipity and programme overlap are important dimensions as well, although to lesser extent.

During the determination of the correlations per metric it appeared that there exist great differences among the participants of the user study. Therefore it was also investigated whether clusters of users that favour a specific dimension or combination of dimensions over another could be created. This was done based on the individual correlations between the 15 metrics and user satisfaction, using the clustering toolkit Cluto. The four clusters that were created indeed encompassed users with similar preferences with respect to the five dimensions. Users in one of the cluster were similar with respect to diversity, while two other clusters contained users that were alike regarding accuracy and serendipity respectively. This was illustrated by the fact that the within-cluster correlations for the metrics that were valued by the users in that cluster were much stronger / weaker.

Based on these results the main objective of the research (*i.e.* to develop a method to predict user satisfaction based on various off-line metrics) was pursued. In order to achieve this goal an expert in statistics was consulted. This expert advised to utilise a method from multivariate statistics, called discriminant analysis to construct a model that can be used to predict the user satisfaction. This model was created using the statistical software package SPSS. To evaluate the performance of the constructed prediction method, K-fold cross validation was employed to calculate its mean absolute error. This showed that the prediction method can predict the user satisfaction achieved by a recommender system based on the off-line metrics Mean Absolute Error, Area Under ROC, Precision, Channel Level Diversity, Genre Level Diversity, Series Level Diversity with a mean absolute error of 0.73922.

Bibliography

[Acz93] Amir D. Aczel, *Complete business statistics*, second ed., IRWIN, Boston, 1993.

[AK07] Gediminas Adomavicius and YoungOk Kwon, *New recommendation techniques for multicriteria rating systems*, IEEE Intelligent Systems **22** (2007), no. 3, 48–55.

[Alp04] Ethem Alpaydin, *Introduction to machine learning*, The MIT Press, October 2004.

[AT05] Gediminas Adomavicius and Alexander Tuzhilin, *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** (2005), no. 6, 734–749.

[AvS04] Kamal Ali and Wijnand van Stam, *Tivo: making show recommendations using a distributed collaborative filtering architecture*, KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (New York, NY, USA), ACM Press, 2004, pp. 394–401.

[BHK98] J.S. Breese, D. Heckerman, and C. Kadie, *Emperical analysis of predictive algorithms for collaborative filtering*, Proceedings of Fourteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, 1998, pp. 43–52.

[BHL07] Marcel Blattner, Alexander Hunziker, and Paolo Laureti, *When are recommender systems useful?*, September 2007.

[BKH01] J.E. Bartlett, J.W. Kotrlik, and C.C. Higgins, *Organizational research: determining appropriate sample size in survey research*, Information Technology, Learning and Performance Journal **19** (2001), no. 1, 43–50.

[Bre06] Glynis Marie Breakwell, *Research methods in psychology*, SAGE, 2006.

[BYRN99] Ricardo Baeza-Yates and Berthier Ribeiro-Neto, *Modern information retrieval*, Addison Wesley, May 1999.

[Car05] G. Carenini, *User-specific decision-theoretic accuracy metrics for collaborative filtering*, 'Beyond Personalization', Intelligent User Interfaces Conference (IUI'05) (San Diego, California, USA), January 2005.

[CdVP+07] M. Clements, A. P. de Vries, J. A. Pouwelse, J. Wang, and M. J. T. Reinders, *Evaluation of Neighbourhood Selection Methods in Decentralized Recommendation Systems*, 2007, SIGIR 2007 Workshop, pp. 38–45.

[CKP07] Jinhyung Cho, Kwiseok Kwon, and Yongtae Park, *Collaborative filtering using dual information sources*, IEEE Intelligent Systems **22** (2007), no. 3, 30–38.

[CLA+] Dan Cosley, Shyong K. Lam, Istvan Albert, Joseph A. Konstan, and John Riedl, *Is seeing believing? how recommender interfaces affect users' opinions.*

[Dam64] Fred J. Damerau, *A technique for computer detection and correction of spelling errors*, Commun. ACM **7** (1964), no. 3, 171–176.

[GRGP01] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins, *Eigentaste: A constant time collaborative filtering algorithm*, Inf. Retr. **4** (2001), no. 2, 133–151.

[Ham50] Richard W. Hamming, *Error detecting and error correcting codes*, Bell Syst. Tech. J **29** (1950), no. 2, 147–160.

[HKBR99] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl, *An algorithmic framework for performing collaborative filtering*, SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM Press, 1999, pp. 230–237.

[HKTR04] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl, *Evaluating collaborative filtering recommender systems*, ACM Trans. Inf. Syst. **22** (2004), no. 1, 5–53.

[HMAC02] C. Hayes, P. Massa, P. Avesani, and P. Cunningham, *On-line evaluation framework for recommender systems*, Workshop on Personalization and Recommendation in E-Commerce (Malaga), Springler, 2002.

[HW99] Myles Hollander and Douglas A. Wolfe, *Nonparametric statistical methods*, 2nd edition ed., Wiley, New York, 1999.

[Jar89] M. A. Jaro, *Advances in record linking methodology as applied to the 1985 census of tampa florida*, Journal of the American Statistical Society, vol. 64, 1989, pp. 1183–1210.

[JB80] Carlos M. Jarque and Anil K. Bera, *Efficient tests for normality, homoscedasticity and serial independence of regression residuals*, Economics Letters **6** (1980), no. 3, 255–259.

[Kle80] W.R. Klecka, *Discriminant analysis*, Sage Publications, Beverly Hills, California, 1980.

[Kre99] E. Kreyszig, *Advanced Engineering Mathematics*, John Wiley and Sons Inc., 1999.

[Krz88] W. J. Krzanowski (ed.), *Principles of multivariate analysis: a user's perspective*, Oxford University Press, Inc., New York, NY, USA, 1988.

[Lac75] P.A. Lachenbruch, *Discriminant analysis*, Hafner Press, New York, 1975.

[Lev66] Vladimir I. Levenshtein, *Binary codes capable of correcting deletions, insertions, and reversals*, Tech. Report 8, 1966.

[Lik32] Rensis Likert, *A technique for the measurement of attitudes*, Archives of Psychology (1932), no. 140, 1–55.

[LSY03] G. Linden, B. Smith, and J. York, *Amazon.com recommendations: item-to-item collaborative filtering*, Internet Computing, IEEE **7** (2003), no. 1, 76–80.

[MAC+02] Sean M. Mcnee, Istvan Albert, Dan Cosley, Prateep Gopalkrishnan, Shyong K. Lam, Al M. Rashid, Joseph A. Konstan, and John Ried, *On the recommending of citations for research papers*, Proceedings of the 2002 ACM conference on Computer supported cooperative work (2002), 116–125.

[McN06] Sean Michael McNee, *Meeting user information needs in recommender systems*, Ph.D. thesis, University of Minnesota, June 2006.

[MH04] Matthew R. Mclaughlin and Jonathan L. Herlocker, *A collaborative filtering algorithm and evaluation metric that accurately model the user experience*, SIGIR '04: Proceedings of the 27th annual international conference on Research and development in information retrieval (New York, NY, USA), ACM Press, 2004, pp. 329–336.

[MKB80] K. V. Mardia, J. T. Kent, and J. M. Bibby, *Multivariate analysis*, Academic Press, London, 1980.

[MLG+03] Sean McNee, Shyong Lam, Catherine Guetzlaff, Joseph Konstan, and John Riedl, *Confidence displays and training in recommender systems*, Proceedings of IFIP INTERACT03: Human-Computer Interaction (Zurich, Switzerland), IOS Press, 2003, pp. 176–183.

[MRK06] Sean M. McNee, John Riedl, and Joseph A. Konstan, *Being accurate is not enough: how accuracy metrics have hurt recommender systems*, CHI '06: CHI '06 extended abstracts on Human factors in computing systems (New York, NY, USA), ACM, 2006, pp. 1097–1101.

[RWN03] Habtom W. Ressom, Dali Wang, and Padma Natarajan, *Computational methods for identifying number of clusters in gene expression data*, Neural Networks and Computational Intelligence, 2003, pp. 233–238.

[Sal89] Gerard Salton, *Automatic text processing – the transformation, analysis, and retrieval of information by computer*, Addison–Wesley, 1989.

[SB87] Gerard Salton and Chris Buckley, *Term weighting approaches in automatic text retrieval*, Tech. report, Ithaca, NY, USA, 1987.

[SJ03] Catherine A. Sugar and Gareth M. James, *Finding the number of clusters in a dataset: An information-theoretic approach*, Journal of the American Statistical Association **98** (2003), 750–763.

[SKKR01] Badrul M. Sarwar, George Karypis, Joseph A. Konstan, and John Reidl, *Item-based collaborative filtering recommendation algorithms*, World Wide Web, 2001, pp. 285–295.

[Som00] Ian Sommerville, *Software engineering*, 6th ed., Addison Wesley, August 2000.

[SPUP02] A. Schein, A. Popescul, L. Ungar, and D. Pennock, *Methods and metrics for cold-start recommendations*, Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002), 2002, pp. 253–260.

[SPUP05] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock, *Croc: A new evaluation criterion for recommender systems: World wide web electronic commerce, security and privacy (guest editors: Mary ellen zurko and amy greenwald)*, Electronic Commerce Research **5** (2005), no. 1, 51+.

[TFFT00] Barbara G. Tabachnick, Linda S. Fidell, Linda Fidell, and Barbara Tabachnick, *Using multivariate statistics*, fourth ed., Allyn & Bacon, Boston, MA, August 2000.

[TH01] L. Terveen and W. Hill, *Beyond recommender systems: Helping people help each other*, HCI in the New Millennium (J. Carroll, ed.), Addison Wesley, 2001.

[TWH01] Robert Tibshirani, Guenther Walther, and Trevor Hastie, *Estimating the number of clusters in a data set via the gap statistic*, Journal Of The Royal Statistical Society Series B **63** (2001), no. 2, 411–423.

[vS05] Mark van Setten, *Supporting people in finding information*, Ph.D. thesis, Universiteit Twente, November 2005.

[WH06] R. J. Wieringa and J. M. G. Heerkens, *The methodological soundness of requirements engineering papers: a conceptual framework and two case studies*, Requirements engineering **11** (2006), no. 4, 295–307.

[Yao95] Y. Y. Yao, *Measuring retrieval effectiveness based on user preference of documents*, Journal of the American Society for Information Science **46** (1995), no. 2, 133–145.

[ZK07] Yi Zhang and Jonathan Koren, *Efficient bayesian hierarchical user modeling for recommendation system*, SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (New York, NY, USA), ACM, 2007, pp. 47–54.

[ZMKL05] Cai-Nicolas Ziegler, Sean M. Mcnee, Joseph A. Konstan, and Georg Lausen, *Improving recommendation lists through topic diversification*, WWW '05: Proceedings of the 14th international conference on World Wide Web (New York, NY, USA), ACM Press, 2005, pp. 22–32.

List of Figures

List of Tables