Personalized Movie Recommender

Integrating social networks in a recommender system for movies

A N N A E L I A S S O N

KTH Computer Science
and Communication

Master of Science Thesis
Stockholm, Sweden 2010

Personalized Movie Recommender

Integrating social networks in a recommender system for movies

A N N A E L I A S S O N

Master's Thesis in Media Technology (30 ECTS credits)
at the School of Media Technology
Royal Institute of Technology year 2010
Supervisor at CSC was Hannes Ebner
Examiner was Nils Enlund

TRITA-CSC-E 2010:123
ISRN-KTH/CSC/E--10/123--SE
ISSN-1653-5715

Royal Institute of Technology
School of Computer Science and Communication

KTH CSC
SE-100 44 Stockholm, Sweden

URL: www.kth.se/csc

Personalized Movie Recommender

Integrating Social Networks in a Recommender System for Movies

Abstract

This thesis project was done for Ericsson Research in Stockholm, Sweden. The purpose was to evaluate how well an existing algorithm in a recommender system predicts movie ratings and get an indication of how the users perceive the recommendations given by the system. The recommendations are computed with a revised User-based Collaborative Filtering algorithm that calculates trust amongst people in a social network to identify the most suited recommenders. The purpose has also been to use friends from a social networking site as the social network and one of the project goals was to build an application on such a site.

The prototyping activities resulted in a Facebook application for personalized movie recommendations called MovieBuddy, where a user could rate movies and choose friends that would influence their recommendations. The recommender system was evaluated from three main aspects: the accuracy of predicted ratings compared to users actual ratings, the accuracy of inferred trust values compared to users actual trust ratings and the users' opinions as gathered from a survey regarding their perception of the recommendations.

The results showed that the users rated the movies in the recommendation list higher than they did anywhere else. Still, both the inferred trust values and the predicted ratings had quite large mean absolute errors and while the users were overall positive to the recommender system, they felt that there was room for improvement when it came to the recommendations. According to the survey, users were interested in more user tasks than the application supported, like being able to filter and search for movies. Overall, they were less interested in rating movies and more interested in what the system could do for them. The results lead to some conclusions on how the recommendation algorithm could be revised and how the system could use social networking sites as a source of information and inspiration.

Personligt rekommendationssystem för film

Integrering av sociala nätvärk i ett rekommendations-system för film

Sammanfattning

Detta examensarbete gjordes på uppdrag av Ericsson Research i Stockholm, i syftet att utvärdera ett rekommendationssystem för film. Systemet använder en reviderad kollaborativ filtreringsalgoritm för att skapa rekommendationer till användare; denna algoritm tar som indata en användares sociala nätverk av vänner, räknar ut vilka relationer som bör vara de starkaste och låter dessa influera användarens rekommendationer. Ett av examensarbetets syften var att använda information om användarens vänner från en social nätverksida och ett av målen med arbetet var därför att skapa en applikationsprototyp på en sådan sida.

Ett av resultaten var att applikationen MovieBuddy skapades, på Facebook, där användare bland annat kan betygsätta filmer och välja vilka vänner som de vill ska influera deras filmrekommendationer. Rekommendationssystemet utvärderades därefter utifrån huvud-sakligen tre perspektiv: hur systemet förutspår betyg jämfört med användarnas riktiga betyg, huruvida systemet låter rätt vänner influera rekommendationerna jämfört med hur användarna själva väljer vänner, och hur upplever användarna sina rekommendationer (för att undersöka detta distribuerades en online-enkät till användarna).

Resultaten visade att användarna satte högre betyg på de rekommenderade filmerna än på andra filmer, samtidigt visade resultaten att systemet genererade ganska stora absoluta medelfel både gällande rekommendationerna och vilka vänner som valdes att influera rekommendationerna. Enkäten visade att medan användarna var positiva överlag till systemet så tyckte de att det fanns utrymme för förbättringar i rekommendationerna. Enligt enkäten så var användarna intresserade av fler funktioner än applikationen stödde, så som att kunna filtrera och söka efter filmer. De var generellt sett mindre intresserade av själva betygsättandet av filmer och mer intresserade av vad systemet kunde ge till dem. Resultaten har lett till förslag på hur rekommendationsalgoritmen skulle kunna revideras och hur sociala nätverkssidor kan användas vid fortsatt utveckling av rekommendationssystemet.

Acknowledgements

I would like to thank everyone that has contributed to the making of this thesis project, which is the final chapter of my Master education in Media Technology at the Royal Institute of Technology in Stockholm, Sweden.

Thanks to my supervisors Mattias Lidström at Ericsson and Hannes Ebner at KTH for their guidance and advises during the process of the project. I would like to give special thanks to Jonas Björk and Simon Moritz at Ericsson, who have helped with the recommender system and taken the time to answer questions and solve many application related issues. Lastly, I would like to thank Victor Schelin for patiently answering any and all questions regarding PHP and application development.

Anna Eliasson

Stockholm, October 2009

Table of Content

1 Project Introduction

This report is the result of a Master Thesis project in Media Technology at the Royal Institute of Technology in Stockholm, Sweden. The project was provided by Ericsson Research AB and this chapter gives an introduction to the area researched in this project and a few related projects as well as states its purpose, goals and research questions. It also contains a description of the report's intended target group, relevant concepts and definitions as well as some of the project's limitations.

1.1 Background

1.1.1 Ericsson

Ericsson is a leading provider of telecommunications equipment and related services to mobile and fixed network operators' worldwide. It is one of Sweden's largest companies with more than 78000 employees around the world and customers in 175 countries. Ericsson has been in the telecommunications market for more than a hundred years and has business areas covering everything from technology research to networks development and system maintenance. More than 40% of all mobile traffic goes through Ericsson networks and they support networks which together serve more than one billion subscribers. (Ericsson 2008).

Through R&D investments as well as new areas such as *IPTV*, Ericsson is continuously making new advancements on the multimedia market. TV solutions such as the newly acquired Tandberg Television have established Ericsson in the TV space. Ericsson's interest in the multimedia market has been the basis for this thesis project, which was done in for the Knowledge Discovery project at the department of Server Layers Technologies at Ericsson Research.

The Knowledge Discovery Systems group conducts research in areas such as *Data mining* and *Recommender Systems* and the last of these is the focus of this project. Recommender systems are a popular enterprise on the web today and used to recommend all kinds of content, in many cases entertainment-based such as movies, books and music, with the purpose of providing the users with more personal experiences.

1.1.2 Related Thesis Projects

This thesis is to some extent an expansion on a project that examined the possibilities of integrating *Social Networks* in a recommendation system for movies at Ericsson and that resulted in an algorithm for predicting recommendations (Shahan 2008). The material reused from that thesis is restricted to the specific algorithm (denoted as algorithm *2.5* in this thesis). The algorithm was during that time tested on a smaller set of users, using one of the Jester jokes datasets[1] and has for the first time in this project been tested using a dataset of movies.

1.1.3 Project Purpose and Goal

This thesis project consists of three main parts. One is a general study of and theory behind social networks and recommender systems, the second is the development of a prototype application on a social networking site and the third is the evaluation of a recommender algorithm that has been developed at Ericsson Research. Three research questions were chosen for the project:

- *How well does the recommender system predict ratings?*
- *Can the existing algorithm be revised in order to optimize the recommendations?*
- *Is a recommender system based on social networks valuable?*

The literature study was done to provide insight into the history and world of recommender systems and social networks. As a step in evaluating the algorithm, Ericsson wished to have an application for movie recommendations built on a social networking site, where users could interact with the recommender system. The purpose of the prototyping activities was to gather the data necessary to evaluate the accuracy of the algorithm as well as gather insight into the users' perspectives on the recommender system. One hypothesis in this project has been that the algorithm returns better recommendations than if random movies were returned. The goal of the project has been to suggest changes which can further the development of the recommender system.

[1] The Jester Online Joke Recommender System developed at Berkley University of California in the United States has gathered millions of ratings on a dataset of a 100 jokes, freely available for download and use at http://eigentaste.berkeley.edu/dataset/.

1.1.4 Delimitations

To balance the theoretical and the practical part of this thesis project, the programming has been limited to the client side of the recommender system; meaning the application. The suggested changes have not been implemented in the recommender system.

There exist several different filtering techniques for generating recommendations. While the one used in the recommender system has been explained in this report, this project does not include an extensive study of these techniques. Their pros and cons have been examined and analysed in another thesis project that also included a study of benchmark evaluation metrics (Björk 2008). Additionally, scalability issues which are an important part of recommender systems performance have not been evaluated in this project. There are also several ways of mining information from different sources in order to create user profiles which are relevant to recommender systems and while these are mentioned in this thesis, how they could be implemented are not addressed.

1.2 Project Methodology

To meet the project goals a literature study was done and an application was built on a social networking site as a tool with which to conduct user tests. The methods used during this thesis project are described in this chapter.

1.2.1 General Approach

The project began with a theoretical phase where previous thesis projects about recommender systems were studied along with relevant theory. As one of the project requirements was building an application prototype on a social networking site, the initial phase also included a brief study of the sites available and their developer documentation and guides.

The majority of time was spent on the prototyping activities, which were initiated early. During the design process, the user tests were planned and a literature search was initiated. This was followed by a literature study that was conducted in parallel with the implementation process. Some of the literature regarding how to evaluate recommender

system was used to revise the design of the user tests before finishing the implementation process. The user tests were conducted during a two week period during which the main findings of the literature study were summed up. After analysing the data and literature, suggestions were made relating to the recommendation process and possible future work.

1.2.2 Literature Search

The literature study was mainly done with materials from the library at the Royal Institute of Technology in Stockholm and the ACM and IEEE Xplore digital libraries. Additionally, internal reports relating to the subject fields were provided at Ericsson.

The main concepts used in the literature search were various combinations of "recommender systems", "social networks", "social network analysis", "data mining" and "trust in social networks". Several articles were also found through the reference lists of previous thesis reports written at Ericsson in the subject fields of *Recommender Systems*, *Social Networks* and *Collaborative Filtering* (Shahan 2008, Björk 2008).

Facebook's own developer site and web tutorials from w3schools.com were used during the implementation process, as well as course literature from courses at KTH in the fields of *Program Development* and *Internet Programming*. Furthermore, the book "*Interaction Design – beyond human-computer interaction*" (Preece et al. 2007) was used during the design process to provide a general perspective on usability and design[2].

1.2.3 Evaluation Process, Reliability and Validity

The application was built as a test bed for the algorithm and was the main resource for gathering data and user feedback. Two user tests were conducted, the first one was to detect any bugs or problems in the application and the second one – the main user test – was to evaluate the performance of the recommendation system. A survey was included in the main user test in order to get the users perspective on the application in general and recommendations in particular.

[2] Main literature in DH2620 Human-Computer Interaction, Introductory Course, at KTH.

The study of the recommender system aimed to be qualitative, meaning that the goal has been to reach a deeper understanding of the system rather than to analyse it statistically (Bell 2005, p.17). Some aspects of quantitative analysis have however been used in order to analyse the results of the user tests. Since the market for recommender systems is evolving and users might become more exposed to them, there is no guarantee that a repeated user test would generate the same responses. This relates to the reliability[3] of the results but has not been considered an issue since the purpose in this project has been to get a user point of view on the recommendations in the users and the system's current state. In order to validate the points on which the data was evaluated, they were discussed with the project supervisor at Ericsson.

A *hypothetico-deductive model* was partly used to evaluate the recommender algorithm. According to this model a hypothesis is first formulated, which can later be either falsified or verified by observing the consequences that it generates (Hansson 2003, p. 48). As previously stated in *1.1.3 Project purpose and goal*, a hypothesis was that the recommendation algorithm used today performs better than if random movies were selected and shown as recommendations. A falsification approach was taken, and if randomly predicted ratings proved to be more accurate than the ones predicted by the recommendation algorithm, the hypothesis would be falsified.

1.3 Report Structure

This report is thought mainly to be read by Master students interested in recommender systems as well as by Ericsson employees working at the Service Layer Technology department.

The second chapter describes the main results of the theoretical study and provide the reader with information on how a user-based collaborative filtering system works in general and how the algorithm used in the user-based Ericsson recommender system works in particular. It also provides a description of the application design process. Chapter three describes one of the main findings of the literature study; an evaluation technique for recommender

[3] A measure of the extent to which an experiment will generate the same results if it is repeated on different occasions (Bell 2005, p. 117).

systems which is thought to be of particular interest to the readers at Ericsson. The fourth chapter describes the prototyping activities and depicts the final application on Facebook, whereas the fifth describes the results of the following user tests set to evaluate the recommender system and application. The following two chapters then sum up the most important conclusions from the user tests and the project as a whole and suggest future work to Ericsson.

1.3.1 Concepts and Definitions

In this report, the user for which a recommendation is being made is denoted the active user *u*. The term "target user" is sometimes used interchangeably with "active user". I_u refers to all items that have been rated by user *u*, $R_{u,i}$ denotes user *u*'s rating on item *i* and $\overline{R}_u$ is user *u*'s average rating. The term *"Web-based social networks"* is used to describe a set of people that are connected in the same social network through some sort of common interaction on the web while *"Social networking sites"* is used to describe sites of social character such as Facebook and MySpace, where users have their own profile and network of friends

2 Recommender Systems and Social Networks

This chapter describes the material gathered during the literature study. The first section provides an overview of recommender systems and the filtering technique used in the recommender system at Ericsson, and should be of especial interest to the user who is unfamiliar with recommender systems. The second section describes Social Network Analysis and how social networks can be integrated in a recommendation process; it aims to provide the reader with understanding of how the algorithm evaluated in this thesis came about; the algorithm is shown in the end of *section 2.2.4*. The last part of the chapter then describes the initial prototyping activities to integrate the recommender system on a social networking site and the design process of the application.

2.1 Introduction to Recommender Systems

In 1966, approximately 20 million words of technical information were recorded every day and after the introduction of the World Wide Web, the number has grown exponentially (McNee 2006, p. 1). Recommender systems are essentially guides meant to help users find relevant information on the web. For instance; systems like Google act as recommenders by enabling the user to sort through large amounts of information. Used by more than 30 % of all internet users, it could be said to be the most popular recommender system on the web (Alexa, 2009a).

2.1.1 The 3 Recommendation Methods

The first recommender system, Tapestry, was created in 1992 to filter e-mails and other documents arriving in a continuous stream. Users of the system recorded their reactions (such as "interesting", "not interesting") when they read a document and these were then used to filter out bad or uninteresting e-mails (Resnick et al. 1997, p.56, Goldberg et al. 1992, p. 61). The creators behind Tapestry coined the phrase *collaborative filtering (CF)*, referring to the collaborative nature of the filtering process. Since then, collaborative filtering (also known as *automated collaborative filtering*) has sometimes been used synonymously to recommender systems (Resnick et al. 1997, p.56).

Today, two other methods for recommending content have been developed: *Content-based* and *Knowledge-based* filtering. Several processes that combine the three in various ways have also been created. Content-based filtering operates by finding information and metadata about items and then compares that information to what a user seems to be looking for. The last type of recommender, knowledge-based, uses patterns or associations between items to create recommendations – for instance buying a lamp might generate recommendations for buying light bulbs. (McNee 2006, p. 14, 57).

2.1.2 Collaborative Filtering

In Tapestry, if a user knew that one of his co-workers always commented on interesting documents he could receive a heads up whenever a new document had been commented on (Goldberg et al. 1992, p. 62-63). Or as Resnick & Varian (1997, p. 56) describe it: Collaborative filtering emulates the natural social process of asking someone for advice for example on what to read or watch. Today collaborative filtering is used in many different web applications and online stores like Yahoo! Music, Amazon for books and Netflix for movies (McNee 2006, p. 2).

One of the most popular collaborative filtering algorithms is *User-based Collaborative Filtering*, also known as *Resnick's algorithm*. It operates by first calculating similarity values between users in a system, then finding the k most similar. These k users constitute a so called *neighbourhood*. The recommender system then uses the neighbourhood's ratings to compute and return a list of n recommendations. In 2001, *Item-based Collaborative Filtering* was proposed as an alternative to User-based CF, which instead of utilizing similarity between users calculates similarity values between items. (McNee 2006, p. 39, 43). *Figure 1* shows a user-based collaborative filtering recommendation process. The matrix on the left is the ratings matrix where user ratings are stored, as explained in the next section.

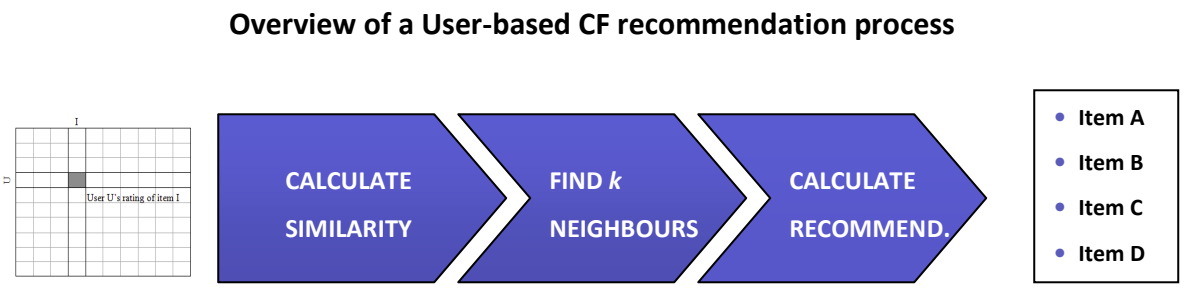


Figure 1) Conceptual User-based CF recommendation process.

2.1.2.1 The Ratings Matrix

Before computing similarity or calculating recommendations, a recommender system needs information about a user – more specifically a user's preferences – in order to generate recommendations. These are stored in a ratings database, also known as a *rating matrix* (McNee 2006, p. 33) where the columns represents the items in the database and the rows correspond to the users in the system.

In collaborative filtering, every position in the matrix denotes one user's rating for one item. The opinions of all users stored in the matrix are used to create *predictions* for items that the active user has not yet rated, and consequently recommendations for items that the active user might be interested in. The matrix is used in different ways depending on what type of algorithm is used. When creating recommendations in a content-based recommender system only the active user's row is of interest. It acts as a preference list which is compared to information which has been gathered externally about the items themselves. (McNee 2006, p. 35, 55). *Figure 2* shows a basic ratings matrix.

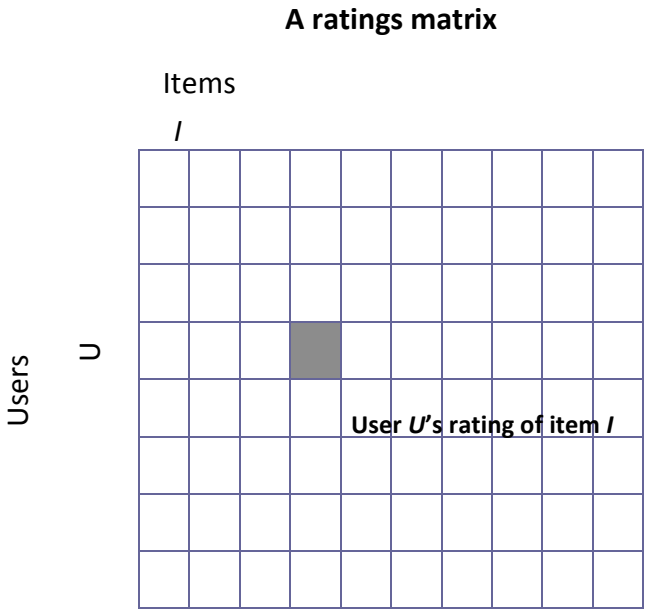


Figure 2) Basic ratings matrix.

2.1.2.2 Calculating Similarity

Users tend to trust people in a given context with whom they share common views (Golbeck 2005, p. 117). This is the reason why similarity values are computed between users in a recommender system.

Pearson Correlation coefficient is the most popular measure used to calculate similarities between users. For two users *u* and *v* it considers all items rated by both users (denoted here by I_u and I_v) and divides the covariance of the two variables with the product of their standard deviation. For instance, if users *u* and *v* have rated item *i* 3.5 and 4.5 respectively and their average ratings are both 3, their covariance will be computed by $(3.5-3)(4.5-3)$.

$$corr(u,v) = \frac{\sum_{i \in I_u \cap I_v} (R_{u,i} - \overline{R_u})(R_{v,i} - \overline{R_v})}{\sqrt{\sum_{i \in I_u \cap I_v} (R_{u,i} - \overline{R_u})^2 \sum_{i \in I_u \cap I_v} (R_{v,i} - \overline{R_v})^2}} \quad (2.1)$$

$R_{u,i}$ is user *u*'s rating on item *i*, and $\overline{R_u}$ is user *u*'s average rating as given by

$$\overline{R}_u = \frac{1}{|I_u|} \sum_{i \in I_u} R(u,i) \tag{2.2}$$

Another commonly used similarity measure is *Vector similarity* (Liu & Yang 2008, p. 85), also known as *cosine correlation*, where a user's ratings represent the user in a high-dimensional vector space. The cosine angle between two vectors in this space indicates how close the users are. Just like with Pearson, the correlation coefficient ranges between -1 and 1 where two users with perfect similarity will have a value of 1. Other well known similarity measures are *Spearman's* ρ and *Kendell's Tau* (Herlocker et al. 2004, p.30).

The Ericsson recommender system has replaced Pearson correlation with a trust measure that corresponds to the similarity between users. This is explained further in *section 2.2 Social Networks*.

2.1.2.3 Finding Neighbours

Finding neighbours is typically done by sorting out the *k* users with the highest similarity value. Research has shown that including neighbours with low similarity can deteriorate the quality of the recommendations, which suggests that it might be better to set a threshold over which the *k* nearest must be in order to be used (McNee 2006, p. 41). Such a limit would make for less computation as well.

Another factor that can be considered when choosing neighbours is the number of co-rated items that the similarity measure has been based upon. A higher number co-rated items makes for a more reliable similarity value, for example a value based on three co-rated items is probably less reliable than a value based on 50 items. It might therefore be a good idea to lessen similarity values which are based on fewer items (McNee 2006, p. 41).

2.1.2.4 Calculating Recommendations

Recommendations are generally made only for items which the active user has not rated. For every such item, a predicted rating is computed based on the ratings from every user in the neighbourhood on that particular item. This implies that only neighbours who have rated the

item are considered. The predicted rating for item *i* is computed by

$$P_{u,i} = \overline{R}_u + \frac{\sum_{v \in U_i} corr(u,v)(R_{v,i} - \overline{R}_v)}{\sum_{v \in U_i} |corr(u,v)|} \quad (2.3)$$

Where U_i is the set of all users who have rated item *i* and $|corr(u,v)|$ is the absolute value of the correlations. For instance, if user *v* and *z* have rated an item *i* 5 and 2 respectively and their average ratings are both 3 and their correlation values with user *u* are 0.8 and 0.6 respectively; the predicted rating is calculated by

$$P_{u,i} = \overline{R}_u + \frac{(0.8)(5-3)(0.6)(2-3)}{|(0.8)(0.6)|} \quad (2.4)$$

The sum of all users' opinions is added to the active user's average rating. A positive result means that the users combined gave the item in question a higher rating than usual. If the result is negative instead, the predicted rating will be lower than the target user u's average ratings, indicating that the item is not as good as an average item. A list of *n* items with the highest predicted ratings is then returned to the user.

2.2 Social Networks

The study of social networks started long before the introduction of the internet and web-based social networking sites such as Facebook and MySpace. In the early 1930's the study of *sociometry* was developed, which was a method for evaluating emotional relationships between people in a social network such as friends or enemies, liking or disliking (Wasserman & Faust 1994, p. 77-78).

Aiming to understand the different properties of a social network is the field of *Social Network Analysis* (*SNA*) which has emerged from social science disciplines such as sociology and social psychology. While these areas to some extent focus on the individuals in a network when trying to understand behaviour, SNA instead operates from the understanding that an individual's *relationships* are the most influential factors on his or her

behaviour. (Wasserman & Faust 1994, p 10). In order to analyse the relationships in a social network, two different mathematical representations have generally been used: the *sociomatrix*, which is a two-way matrix where the rows typically contain the sending actors and the columns the receiving ones, and the *sociogram*, which is a graph where the nodes represent people and the lines the interactions between them. (Wasserman & Faust 1994, p. 77-78).

2.2.1 Properties of a Social Network

Normally, research on a social network is not done using a real network (meaning one that has occurred naturally) but on automatically generated networks (Golbeck 2005, p. 57). There have also been several advances in developing ways to mine or extract social networks from the web using semantic technologies (Matsuo et al. 2006). While these techniques are interesting for instance when seeking to infer trust between actors on the web, they have been decided with Ericsson to be outside the scope of this thesis and will therefore not be discussed further here. The main reason for generating a network instead of waiting for one to develop naturally is that automatic generation is quicker (Golbeck 2005, p. 57).

In order to generate a network automatically it is necessary to know the characteristics of a social network, such as the fact that it has the same properties as a *small world network*. This for instance means that the fraction of connections amongst neighbours of an active user compared to the total number of edges is high. This quality is called *connectance* and basically indicates how strongly actors in a neighbourhood are connected, which in a small world network is expected to be much higher than in a randomly generated network. Also, the average shortest path length for a small world network grows logarithmically with the size of the graph. This must be regarded for computational purposes. (Golbeck 2005, p. 57, 87).

Given a social graph, relationships between nodes can have different properties depending on what type of social network that the graph represents. For instance, some social networking sites enable users to describe their relationships with labels like "sister", "brother" or "classmate". These constitute *non-directional relations* because there is no difference between the line from node *A* to node *B* and the line from node *B* to node *A* (Wasserman & Faust 1994, 70-77). On Facebook, users can send friend requests to other

users. When a receiving user has accepted this request, a non-directional relationship is formed between these users. Mining such a social network therefore results in a graph with information about the existence of users and links.

It is safe to assume that a user's social network contains relationships of different strengths, both the user's best friends and acquaintances. Some sites allow numerical ratings to describe the level of closeness between two actors (Golbeck 2005, p. 40-41) which implies that actor *A* can rate the relationship with actor *B* higher than the other way around. A relationship which might differ depending on which of the actors you ask is called a *directional relation* and the graph is a *directed graph* (Wasserman & Faust 1994, 70-77). Information about how strongly a user values a relationship is personal and sensitive and is as such rarely shared with third-party applications. Because such values can only be used internally, a third-party system must infer this trust information in some other way. Inferring such information is one of the big issues for researchers who are trying to create a web of trust on the internet (Golbeck 2009, p. 3). How this is handled in the Ericsson recommender system is explained further in *section 2.2.3 Inferring trust to replace similarity*.

When automatically generating a directed graph, connectance is computed with twice as many links, as every two nodes can be connected by two edges (Golbeck 2005, p. 57). Complexity of the graph can be added depending on what you wish to study, for instance, visualizing both directed and non-directed lines in the same graph.

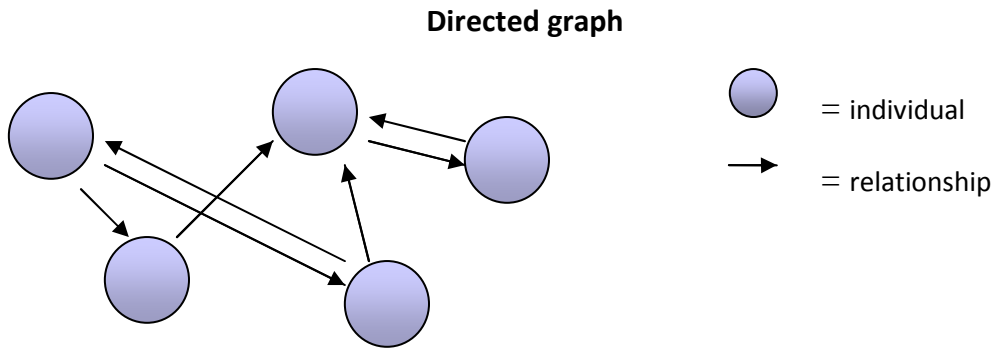


Figure 3) Graph with directed relationships.

2.2.2 The Notion of Trust

Many interactions on the web are done between people who do not know each other. Computing with social trust involves validating people and content and thereby creating a sense of trust between such people. Golbeck describes trust as "*Alice trusts Bob if she commits to an action based on a belief that Bob's future actions will lead to a good outcome*." and Ljung & Wahlforss calls trust "[an] attitude towards the future. By trusting one *places a bet on the future* and thus acts as if this future only contained a certain possibility of action."[4] Social trust is by these definitions a personal opinion rather than an objective measure.

The concept of using trust as a measure has been researched frequently in the context of peer-to-peer networks and rating systems for online communities (Ziegler & Lausen). These spaces often contain large quantities of content which is produced by the users themselves thus making it increasingly important for the users to be able to separate trustworthy content or users from the untrustworthy ones (O'Donovan 2009, p. 213). Computing with trust is relevant to the context of recommender systems because users prefer to receive recommendations from parties that they trust; both from users whom they know and trust as well as systems which they trust and understand (Golbeck 2005, p. 118). This is the main reason why friends should make good recommenders.

As recommender systems handle personal and sometimes sensitive information it is important to note that in order for users to be able to feel trust for a system it must be secure. Security and reliability issues will however not be discussed further in this thesis other than to say that they are important issues to handle.

2.2.3 Inferring Trust to Replace Similarity

The algorithm which is under evaluation in this thesis replaces the user-based similarity measure in a collaborative filtering algorithm as explained in *section 2.1.2.2* with a social network-based correlation measure. It basically considers which relationships that exists in a social network and from that infers how much each user should trust the others. If a pair of users is connected through many mutual friends, they are assumed to trust each other a great

[4] Emphasis in original.

deal. Every user in the system is associated with a matrix containing these trust values as shown in *figure 4*. How the trust is computed is explained in the next section.

A Basic Relations Table

Active User	Neighbour	Trust
User A	User B	0.7
User A	User C	0.4

Figure 4) Matrix with trust value between the active user and the neighbours.

2.2.4 Computing with Trust

There have been several efforts made to compute trust values in social networks. One straightforward way that has been implemented in *binary systems* (where the users can choose either to trust someone or not) is shown below. Where d is the maximum depth from the active user which is considered and n is the shortest path from the user in question to the active user. (Golbeck 2005, p.112).

$$\frac{d-n+1}{d} \tag{2.5}$$

This inference process considers only the depth of which a node is from the active user which means that all users with the depth of one are trusted completely.

The Ericsson algorithm calculates a default trust value for each node in the network in a similar way – considering the maximum depth and a nodes distance from the user, with $d-n$. The maximum depth is set to three (the reason for this is explained below) which means that all default trust values will range between 0 and 2. When calculated like this, all users on the same depth will have the same trust value which is not a nuanced representation of reality. With regards to movies for instance, it seems unlikely that all of the active user's friends are equally suited as recommenders and should therefore be given different trust values.

In order to compute more diverse trust values, the algorithm takes into account the number of mutual *cliques* two users in the same social network are a part of. A clique is a maximal complete *sub graph* (meaning that all nodes are connected to each other by two edges) containing three or more nodes. (Wasserman & Faust 1994, p. 254). This means that a large clique represents a large number of users who know each other. The definition of a clique that was used in the thesis report where the algorithm was first published was; if the active user has two or more neighbours which he or she trusts and if they have a relation in *either direction of the graph*, the graph forms a clique.[5]

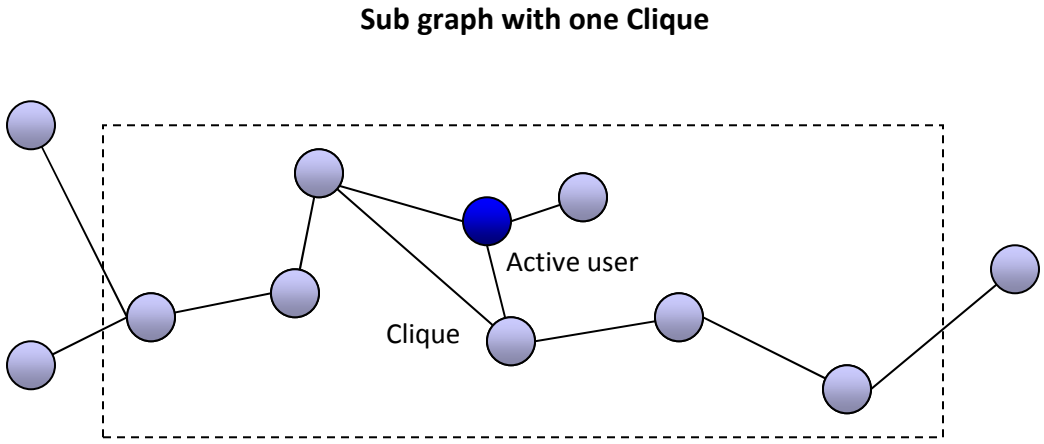


Figure 5) Sub graph with maximum depth three to the active user

Finding cliques has proven to be computationally expensive (NP-problem[6]). To save time and memory, a sub graph including only users with a maximum distance of three from the active user is extracted from the entire social graph. This means that nodes with a distance greater than three will not have a similarity value with the active user. Such a sub graph is shown in *figure 5*. The nodes that do not have a direct relationship with the active user but are connected through other users are called *transitive friends* and the links *transitive relationships*. The trust value between the active user and a friend of a friend will always be smaller than for a direct friend. A higher number of mutual friends between the active user and the tran-

[5] Shahan, M. (2008). *Personalized TV with Recommendations – Integrating Social Networks*. Master's thesis
[6] http://mathworld.wolfram.com/NP-Problem.html

sitive friend will however increase the trust value.

The cliques which are found are first broken up into smaller cliques of size three. The cliques that do not include the active user are then discarded from further computation. Meaning if four cliques {0,1,2}{0,1,3}{0,2,3} and {1,2,3} are found, where user 0 is the active user, only the first three are considered when calculating trust values. As previously mentioned; the general idea is that the more cliques that one user shares with the active user, the stronger the connection is between them.

As the whole graph then consists of only size three cliques where the active user is one node, the number of cliques that *user v* shares with the *active user u* can be calculated by extracting 1 from the number of edges connecting *user v* outwards. This is used when inferring the relationship value between them with

$$corr(u,v) = \frac{D_{u,v} + 1 - \left(\frac{c}{m}\right) - (1-c)(\overline{D}_u)}{D_u + 1} \tag{2.5}$$

where $D_{u,v}$ is the default trust value between the two users, m is the number of mutual friends, c is a constant determining how important the number of mutual friends should be and $\overline{D}_u$ is the mean value of all neighbour ranks. The inferred values are between 0 and 1.

Wasserman & Faust (1994, p. 256) suggest that a clique might have to strict a definition to be used for social network analysis, as the absence of a single line disqualifies a group. The implications of using clique-count to infer trust values between people in a social network are discussed further in the following section. This (4.5) is however the algorithm used in the recommender system to compute trust values between friends in a social network.

2.3 Integrating the Recommender System on a Social Networking site

2.3.1 Friends as Recommenders

Golbeck (2005, p. 33) has stated that there are many ways in which trust can be broken down, for example you might trust a friend to make good recommendations for action movies but not for thrillers. This means that a just because a user trusts his or her best friends in general, the user does not have to trust them as movie recommenders.

Computing cliques (which the system does to infer trust values, as described in *Chapter 2.2.4 Computing with trust*) does however not imply that the user's best friends are found, but rather that the social butterflies in the social network are. For instance, if two best friends have completely different social networks and only a small number of mutual friends, they will not be recognized as the strongest relationship due to how the correlation algorithm works. One of the goals of the application was therefore to see how well the algorithm infers trust values in a social network of friends. This was evaluated by asking user's to give explicit ratings of their friends and then comparing these to the inferred trust values.

2.3.2 Choice of Platform

Googles social networking site Orkut was considered as a possible platform on which to develop the application. It reaches 0.6% of the world's internet users with the largest user base in India and Brazil (Alexa 2009c). The main reason why Orkut was considered was that it has a flexible application platform – OpenSocial. Developed by Google, OpenSocial allows any application to be incorporated on any and all social networking sites that support the OpenSocial APIs, such as MySpace, Flixter and Hi5 to name a few (OpenSocial 2009). This would mean that relationship data and user information could be mined from several sites at once.

Facebook was eventually chosen as platform for the application because of its popularity in Sweden (where Orkut is yet to be established) and its' well developed PHP documentation. Facebook.com is the most popular social networking site in the world and reaches more than 20% of the world's internet population (Alexa 2009d). The Facebook platform was

released in 2006 and is according to the Facebook Developer Wiki now sporting more than half a million developers (Facebook 2009a). The process of starting to develop a Facebook application is very well documented and several programming languages including PHP, JavaScript and Python have client libraries for Facebook's API.

2.3.3 Design Process

Due to the project's time constraints, the design was not subjected to any user tests before launch and the design process was based a great deal on previous experience with user interface and interactive design as well as advice mainly from the book *Interaction Design* by Preece, Rogers & Sharp.

According to Preece et al. (2007, p. 536), it is recommended that *low-fidelity prototyping*, such as cardboard cut outs and sketches, is used during the construction of a product. These tend to give the developers a higher flexibility when it comes to possible changes, as the concepts are low in time and monetary cost. The requirements on functionality in the application however meant that a *high-fidelity prototype* had to be created, which is a prototype much like the finished product, preferably built in the same language as the real product would be. The advantage of the *high-fidelity prototype* is an application that is fully interactive and therefore could be used for exploration and tests.

In this project, the high-fidelity prototype served its purpose well – data could be gathered as well as some insights into applications development on the Facebook platform. The down side of skipping a low-fidelity development is the risk of losing flexibility when it comes to modifications and redesign. If further development is done based on the prototype, it would for instance be recommended to evaluate the user perspectives which have not yet been taken into account.

2.3.3.1 List of Requirements

A simple requirements list, as shown below, was written together with Ericsson as a first step in the design process. This was done after reaching general understanding of what was expected from the application and the data it would generate. As described by Preece et al. (2007, p. 476), identifying requirements is a necessary step towards the design process.

Requirements as such can come in many different forms of abstraction, but should always aim to describe what the product should do and how it should act in the clearest way possible. With this in mind, the outline was used to discuss the application further at Ericsson, to reach a joint understanding of the overall functionality of the application.

Requirements list

Write an application where:

- Friends who also have the application are listed
- Movies are listed
- Recommendations are shown
- Random movies are shown
- Users can rate movies
- Users can choose which friends to have as recommenders
- Users can set trust values to friends
- Users can rate recommendations

Information that the recommendation system needs in order to create a predicted rating:

- A social graph
- All movie ratings from a user
- All movie ratings from a user's friends

2.3.3.2 Related Applications

As inspiration when designing the application, several other recommender systems and applications on the web were studied. The purpose of this was to get ideas on what could be done in the prototype as well as what not to do. After studying these applications, a first design suggestion was created, followed by an assessment of how this could be implemented in PHP.

FilmTrust

FilmTrust is a website created as a part of a research study at the University of Maryland, which combines social networks, movie ratings and reviews. Users are encouraged to rate how much they trust their friends movie taste on a scale from 1-10 (1 is low, 10 is high) and movies on a scale of 0-4, which is then used to calculate predicted ratings for other movies.

Rate movies in FilmTrust

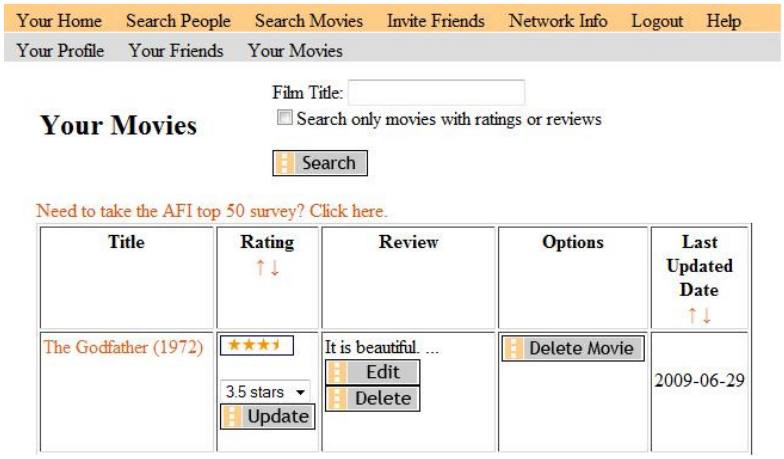


Figure 6) Print screen of the FilmTrust website (http://trust.mindswap.org/)

but the website covers the basic functionality described in the requirements list above and more. A user can see a friend's movie ratings and reviews by visiting their profile or by searching for a movie. Research however indicates that users tend to rate towards predicted ratings that are provided by the system (O'Donovan et al. 2008), which suggests that it would be better not to visualize the predicted ratings. Golbeck (2006, p. 119), one of the founders of FilmTrust, states that users can be sensitive when it comes to revealing what trust values they assign their friends, and because the system is reliant on the accuracy of this information it should be kept personal and invisible from other users.

MovieLens

MovieLens is another research project, at the University of Minnesota in the United States, which is based on the same filtering technique as FilmTrust is namely *collaborative filtering*

(MovieLens 2009, Golbeck 2005 p. 116). *Figure 7* illustrates how the recommendations are shown on the MovieLens website. Blue stars indicate a rating which has been set by the user, and the red is the predicted rating calculated by the recommender system for a movie which the user has not yet rated.

MovieLens recommendations

Figure 7) Print screen, recommendations from MovieLens (http://www.movielens.org)

The MovieLens website offers some additional functionality. The users have the option of letting the system know if they like or dislike the recommendation or some of the tags associated with the movie in question. That way the system can learn more about the users preferences, which can be taken into account in future recommendations. Group recommendations can also be generated for a user and friends of his or her choice, which can help users who want to watch a movie together. MovieLens users can help improve the movie list by flagging movies which for some reason do not belong in the system – it might have a duplicate or a broken link to the Internet Movie Database (IMDb.com). The users can read additional information about the movie directly on the site, by following a link to IMDb.com or by clicking the letter & star-symbol, which represents an existing thread in the MovieLens forum.

Flixter

Flixter.com is a social movie site and recommender system. It is available on Facebook and is currently one of the most popular applications with over 17 million monthly users (AppData, 2009). The application has several options for displaying and rating movies as

well as friend compatibility tests, movie quizzes, trailers and theatre listings (for U.S. residents only). Flixter is also available on other social networking sites such as MySpace and Bebo as well as a mobile application for iPhone and Android [7].

The Flixter application on Facebook

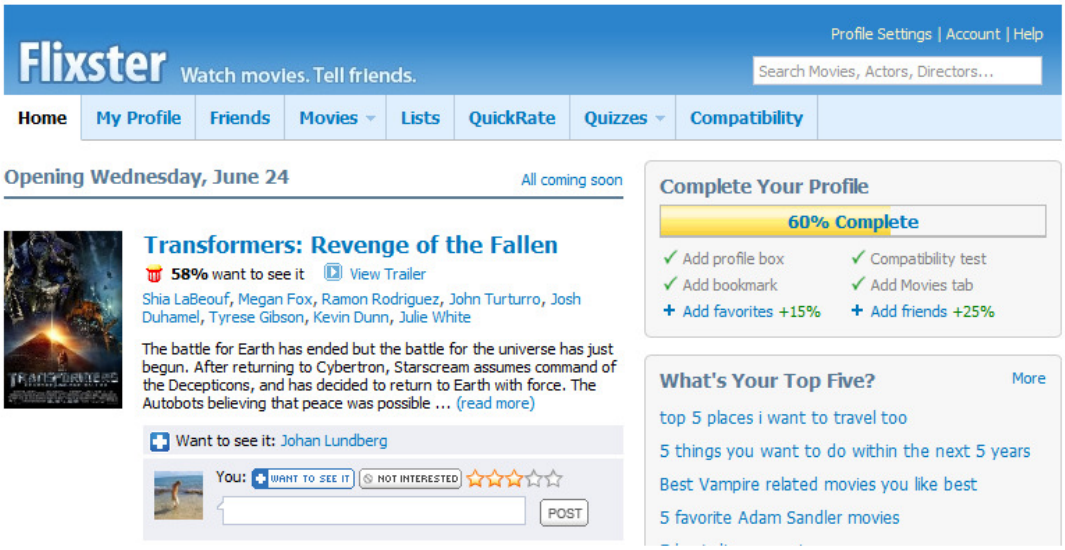


Figure 8) Print screen, Flixter on Facebook. (http://apps.facebook.com/flixster/)

As seen in *figure 8*, a user is presented with several rating options as well as explicit information about how friends have rated movies. Such visible ratings might have implications on how users choose to rate movies. Ljung & Wahlforss (2008, p. 30) state that people in general have a tendency to change how they present themselves depending on what the situation is and who their audience is. If users decide to rate movies based on how they want others to perceive them the consequence might be that ratings which do not really reflect the user's taste are inserted into their ratings matrix, giving the system a distorted view of the user, possibly resulting in bad recommendations for the user. There is on the other hand a social aspect of being able to see a friend's ratings, much like being able to view friends News Feeds, photos and profiles (which are all Facebook features).

[7] http://www.android.com/market/#app=flixster

Jinni

Another application that has been viewed as inspiration was the movie recommender web-site Jinni.com. The site has been publicly available in beta-version since the 7th of October 2009 and while many of their functionalities cannot be used yet they have received several awards for their technology (Jinni 2009a). Movies can be searched by content where Jinni analyses movie plots, mood and more, or they can be browsed by mood, plot, genre, time/period, place and other tags.

Results are shown as images which are sized by relevance but can be ordered for instance by duration or rank. The results can be further adjusted on terms like "Little known" to "Well known" or "Light" to "Serious" by dragging a slider. The site aims to sort movies by different characteristics which has been done by first letting film professionals manually tag movies with as many aspects as possible and then having their system continue the process manually by learning from the available tags (Jinni 2009c).

Recommendations from Jinni

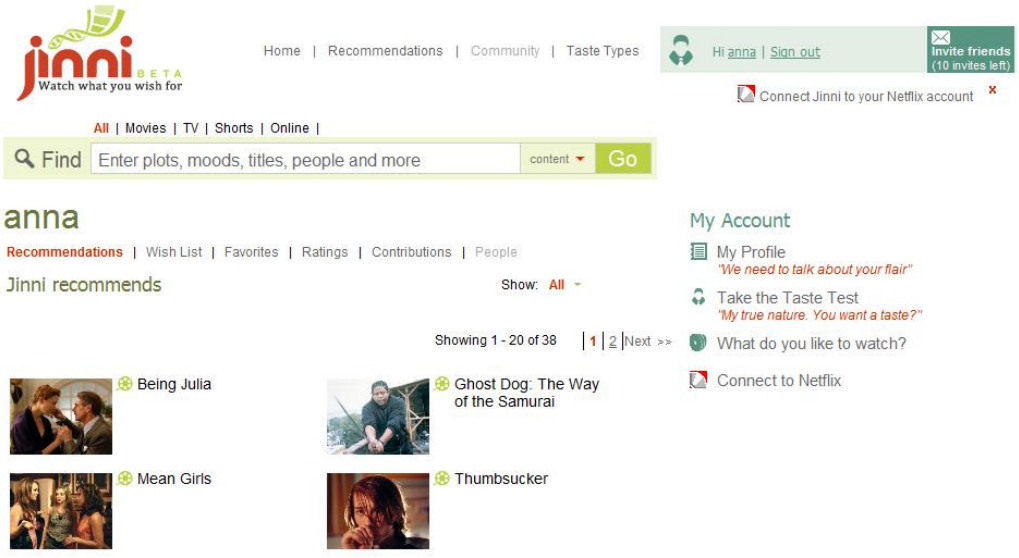


Figure 9) Print screen showing Recommendations from Jinni (http://www.jinni.com)

It provides links to TV-listings, DVD-rentals or online streaming. Recommendations can be given to a user from the system, similar users or explicit recommendations from friends (Jinni 2009b). Recommendations can also be given to a group of users.

3 Evaluating Recommender Systems

This chapter provides an introduction to *Human Recommender Interaction theory (HRI)*. The HRI process model takes a user-based approach to the development of recommender systems and this chapter describes how it can be used to evaluate recommender systems. This chapter should be considered a pre-study of the subject and its aim has been to illustrate the possibilities of HRI for future reference. The last part of this chapter describes the limited extent to which HRI has been incorporated in this project.

3.1 Human Recommender Interaction Theory

Evaluating a recommender algorithm is usually done by measuring its accuracy with different types of *accuracy metrics* (Herlocker et al. 2004, p. 6). McNee (2006, p. 127) suggests that evaluating a recommender algorithm is not only a question of measuring how accurately it predicts ratings, it is equally if not more important to understand the user's intentions when asking for a recommendation. Evaluating the user's perspective and recognizing that the user has a purpose when utilizing a recommender system beyond what accuracy can fulfil has become the focus of some research devoted to recommender systems (Herlocker et al. 2004, McNee 2006).

One of the preconceptions with recommender systems has been that the only requirement a user has when using the system is to get a good set of recommendations as tailored to his or her taste (McNee 2006, p. 5). McNee argues that developers must realize that there are other contextual requirements which might be relevant for the user. In the context of movie recommenders these can be only movies that are available on DVD or in a particular store – and understanding these user tasks are important when creating a *useful* recommender system. *Human Recommender Interaction theory* (HRI) has been developed in recent years in order to add a user perspective to the evaluation process of recommender systems and recommender algorithms.

3.2 The HRI Process Model

The evaluation process in this project was partly based on the classic idea of measuring

algorithm accuracy and done with one metric (*mean absolute error*). While the results of this evaluation provided insight into how well the algorithm performed, it did not consider the user perspective. To do this, a survey was given to the users, which is described in *section 3.4*.

The HRI Process Model, shown in *figure 10*, describes how a recommender system should be evaluated using HRI (McNee 2006, p. 149). It contains the following four steps: 1) Recognizing the users and the user tasks, 2) analysing and translating the user tasks into HRI language, 3) deciding which metrics are best to evaluate the HRI tasks and 4) choose the best algorithm for any of the described HRI tasks.

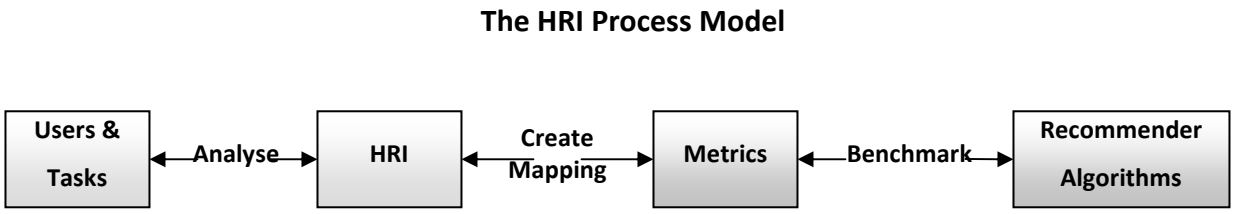


Figure 10) The HRI Analytic Process Model (McNee 2006, p. 149).

3.2.1 Recommendation List

One of the issues regarding accuracy is that it is measured on an item to item basis. While a user is presented with a list of recommendations (as normally done in a recommender system) the impact that the list as a whole has on a user is not evaluated. For instance, an individual item (like the movie "the Matrix") might be a good recommendation, but an entire list containing the same movie in different editions and its sequels might not be all that appreciated. This problem is known as a *similarity hole*. It is most often encountered in systems using *Item-Item Collaborative Filtering* (McNee et al. 2006, p. 1098). Understanding the user's perspective including the fact that a user views a list of items rather than individual items is however an issue for all types of recommender systems (McNee 2006, p. 6) and HRI therefore evaluates a recommender system based on the *recommendation list* it produces.

3.2.2 Users and User Tasks

A vital step in HRI analysis is getting a clear perception of the target audience, as it both

helps in recognizing different types of users and then enables mapping between use cases and specific algorithms. According to Ericsson, the recommender system should not yet target a particular audience. It might therefore be worth to note that while a movie recommender in general should be of interest to a wide variety of people, the distribution channel might narrow the scope of users to the system. For instance, the social movie site Flixter.com that have a movie application on Facebook (much like the one built in this project)[8] has an overrepresentation of 18-34 year olds (Alexa 2009b). Distributing the recommender system through a Facebook application could affect the audience and another distribution channel might reach a different audience all together.

Herlocker et al (2004, p. 9) stated that there are many different user tasks that a recommender system could choose to support, however, only a few are usually implemented. The most common is *Find Good Items*, which basically means creating a list of recommended items and return it to the active user, either with or without predicted ratings. HRI recognizes a set of different user tasks. It is worth noting that Human Recommender Interaction theory was developed from research on recommender systems in digital libraries, therefore several of the user tasks described in HRI might not be significant to movie recommenders. This is described further and discussed in *section 3.3 Applying HRI to a Movie Recommender* and *Chapter 5 User Tests and Results*.

3.2.3 HRI Aspects

The second step in the HRI process model is translating the user tasks into HRI language in order to simplify and structuralize the interactions between user and system. In HRI terminology the words in this language are referred to as *Aspects*. These are developed based on digital libraries, as noted in the section above, which should be kept in mind when applying them to other domains.

Describing a user's perception of the recommendations given in a recommender system are eight different aspects: *Correctness,* measuring how relevant the user thought a recommendation was, *Transparency,* how well the user understood why an item was recommended, *Saliency,* how much the recommended item stands out, *Serendipity,* how unexpected a recommenda-

[8] Chapter 2 contains a description and discussion of Flixter.com and their Facebook application.

tion was, *Quantity,* relating to how many recommendations were given, *Usefulness,* how well does a recommendation help the user in his or her search for a relevant item, *Spread*, how many of the total amount of items that are recommended and *Usability*, measuring how well the user understands the system interface and process. (McNee 2006, p. 134).

Another eight aspects describe the user's overall perception of the recommender system: *Personalization,* how personal the recommendations were perceived to be, *Boldness,* how strongly items are recommended, *Adaptability,* how much recommendations change when a user changes something in his or her profile, *Freshness,* how new the items feel, *Risk Taking/Aversion,* recommendations of items which the user finds obscure, *Affirmation,* recommendation of items which the user finds safe, *Pigeonholing,* how similar are the recommendations to each other, and *Trust*, how sure is the user that the recommendations will be relevant and that the system is secure. (McNee 2006, p. 139).

McNee (2006, p. 8) states that there in general is a gap between users and recommender system where contextual information from the user is lost together with information explaining the systems actions to the user and the purpose of HRI is filling in this gap. In some research, *transparency* in recommender systems has been utilized to try and minimize this gap. Transparency refers to how much of the recommendation process is explained to the user, a transparent system can be considered to emulate the social process of asking a friend for a recommendation much more correctly than a closed recommendation process does. Research has suggested that users feel more confident with systems that generate transparent recommendations (Sinha & Swearingen 2002, p. 831).

An example of a system where the recommendation process is shown is PeerChooser (O'Donovan 2009, p. 241). It is an interactive interface which visualizes a user's neighbourhood and movie preferences and lets the user fine-tune recommendations by dragging neighbours closer or further away as shown in *figure 11*. O'Donovan et al. (2008, p. 1088) proved that the visualization could help generate better accuracy as well as improve user experience.

PeerChooser: Visual Interactive Recommender

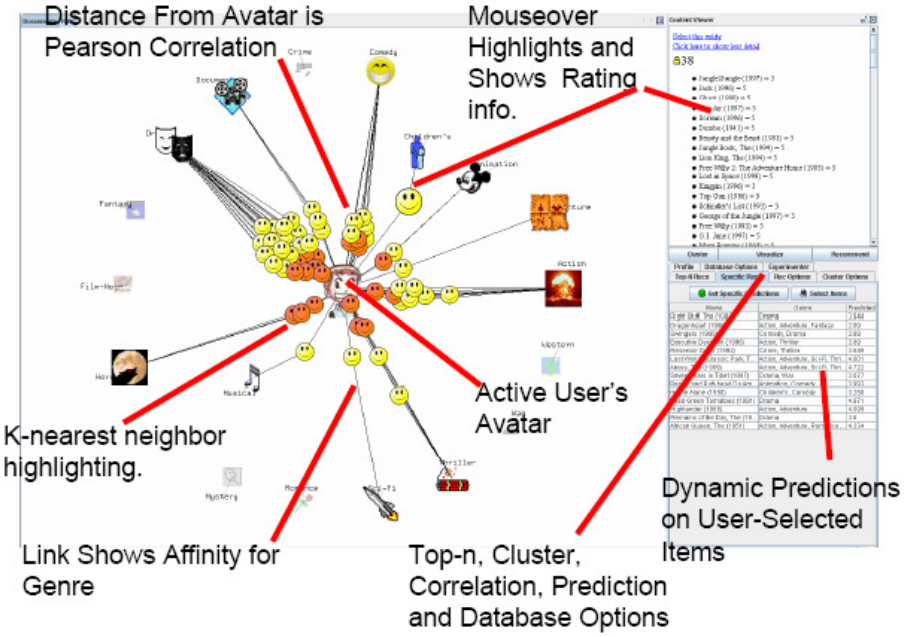


Figure 11) Screenshot of PeerChooser's interface (O'Donovan et al. 2008)

Transparency is only one of the aspects of the recommendation process evaluated in HRI.

A user task might have implications on the design of a recommender, which is why there exist another set of aspects to translate the important elements of user tasks. *Concreteness of Task* suggests that a user might not know what he or she is looking for and a possible design implication of this is enabling a user to browse through existing recommendation lists. *Task Comprising* regards a recommender's ability to translate a user's language and return relevant information, which is important as a user might be in the mood for something particular but is unable to express it clearly. *Appropriateness* determines if the recommender as a whole gives appropriate recommendations for a certain user. This is connected to *"Expectations of Recommender Usefulness"* which for a novice user might refer to an intuitive system design, whereas an experienced user might primarily have expectations on the system's ability to help solve a task. Lastly, *"Recommender Importance in Meeting Need"* is when a recommender system describes what type of information a user can expect to find through the system and what is excluded, which for a movie recommender can be trailers or full cast list.

To support all of these user tasks, a recommender system must be quite versatile. Depending on what type of system one wants to create, some of these aspects will be more important than others. In this project, basically only Herlocker's *Find Good Items* has been explicitly taken into account during the application design process. To evaluate the importance of other aspects, a survey was included in the user test of the application as described further down.

3.2.4 Metrics

The third step of the HRI process model is choosing metrics on which a given algorithm should be measured, for example against benchmark or other algorithms. As previously stated, the most common property to evaluate in recommender systems is accuracy, which measures how close a predicted rating for an item is to a user's actual rating. There exists several different metrics which can be used to measure accuracy; mean absolute error (*MAE*), *Precision/Recall* and *ROC Curves* to name a few (Herlocker et al. 2004, p. 19-25). According to Herlocker et al. (2004, p. 6) there seems to be a limit at 0,73 (on a five-point rating scale) for how accurate predictions an algorithm can generate. It is not possible to have less than a certain amount of errors since users themselves vary in their ratings over time.

McNee (2006, p. 169) presented a set of new metrics based on previous research on recommender systems. For instance a popularity metric, which calculates the popularity of a research paper *j* by dividing the number of citations for the given paper with the number of years ago the paper was published (rounding up so that 1 year includes papers published sometime between today and up to a year ago), and a personalization metric to measure how similar recommendation lists are between different users.

A table of how different metrics and aspects have been shown to correspond is depicted in *figure 12* (McNee 2006, p. 196). A mapping between metrics and aspects suggests how an algorithm should perform on the different metrics to suit a certain user situation. For instance; a certain user finds *transparency* to be the most important aspect. According to McNee (2006, p. 196), transparency corresponds to high scores on the popularity and

authority metrics and a low score on the personalization metric. A suitable algorithm for the user in question would score accordingly.

Mapping metrics to aspects

	Popularity	Authority	Personalization	Boldness
Correctness				
Transparency	+	+	-	
Saliency			+	+
Serendipity	-	-	+	
Quantity				
Usefulness				
Spread			+	
Usability				
Personalization	-	-	+	
Boldness	-	-		+
Adaptability			+	
Freshness				
Risk		-		+
Affirmation		+		
Pigeonholing			-	
Trust				-

Figure 12) How four metrics are mapped to different Aspects (McNee 2006, p. 196).

3.3 Applying HRI to a Movie Recommender

Influenced by beliefs from the area of Human Computer Interaction, Herlocker et al. states that an evaluation process must start with understanding the user tasks which the system will serve (Herlocker et al. 2004, p. 11) and in order to get a better understanding of the users, a survey was included in the user test of the application. The survey also aimed to identify important user tasks and a list of user tasks (seen in *figure 13*) was created inspired by the aspects described in *Chapter 3.2.3 HRI Aspects*. The list presented here should only serve as a

suggestion of user tasks which a movie recommender system could choose to support. Descriptions of the user tasks follow below. The purpose of defining these user tasks is to illustrate how they could be incorporated in a recommender system.

User tasks in a movie recommender

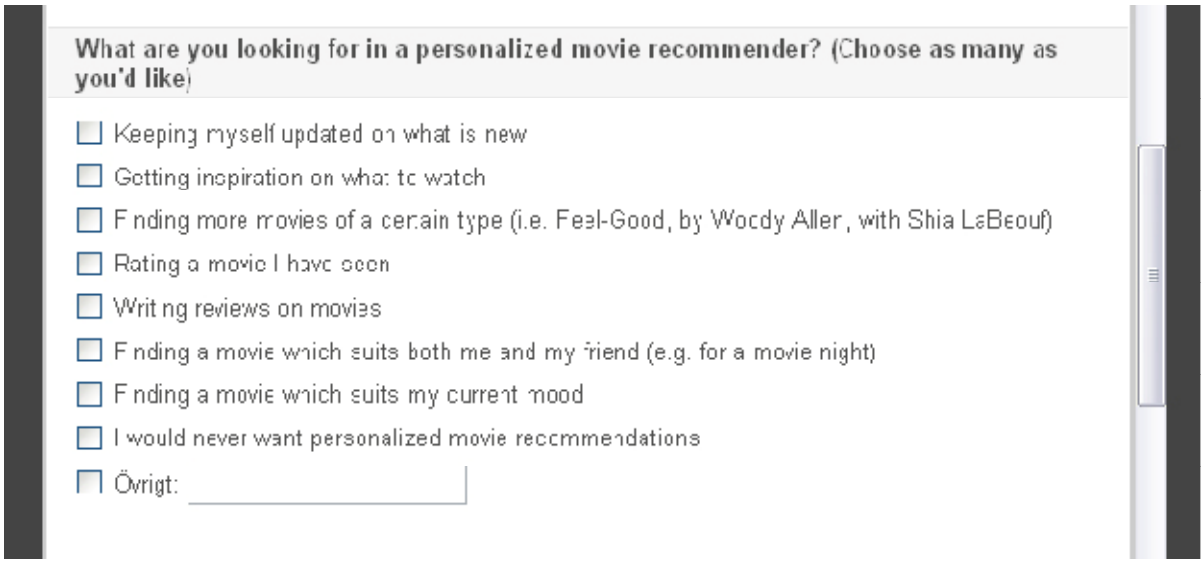


Figure 13) Survey question regarding which user tasks the system should support.

Getting inspiration on what to watch is based on McNee's *"Fill out reference lists"* (2006, p. 156). To support this user task, the system could return a list of recommended items that the user should be interested in based on the user's history. McNee (2006, p. 157) proposed that this task could have variations like finding novel items or items in a specific genre and the input for the recommender should be all the items which the user has already seen, or those of a certain genre or context and the output is a small set of movies which the user should consider. This could be supported in the system by implementing different filtering functions for instance, thus also supporting the user task of *Finding more movies of a certain type*. *Finding a movie which suits my current mood* is inspired by Jinni.com, a recommender website for movies that has made it possible to filter and search movies based on mood (more on Jinni in next chapter). This could be a variation of *Finding more movies of a certain type*, where the input is probably a small set of movies on the given theme, and the output a small set of recommended items.

A related task which was not included in the survey but might be important to consider is *"Find starting point"*, which is about finding items in a new genre which the user has had few or none encounters with previously, maybe *film noir* has caught the user's interest recently. The input should be the user's previous encounters with the genre and the output a set of items which are well known in the genre or that has been nominated to and winners of established movie awards and festivals. If the user has seen one such movie, maybe this should also be presented amongst the recommended items as to strengthen the user's perception of the genre or show the user that he or she has already come across an item like the one requested. The output can be a much less novel list than *"Finding more movies of a certain type"* as the user does not have an extensive knowledge about the type of item asked for. (McNee 2006, p. 157).

Keeping myself updated on what is new is based on McNee's *"Maintain awareness"* (2006, p. 157). It refers to wanting to get more information on what is new in movies. Variations of this task are filtering out movies from specific directors or from movie festivals and so on. The simplest way of supporting this could be through filtering on the newest additions in the database instead of based on personal preference. If the movie database is large, the user might wish to only keep up with what is new in a certain area and as such the input could be the user's current state of knowledge.

Finding a movie which suits both me and my friend might be interesting for a social movie recommender system to support. Implemented in the MovieLens and Jinni systems, co-recommendations still seem to be a quite novel feature to incorporate in a recommender system. The input could preferably be two or more user profiles and the output a list of items which neither of them have seen. Adding an option as to include movies which one or several of the participants have seen might be useful since it being a new movie might not be a requirement for them to want to watch it.

4 Application Prototype

This chapter describes the application that was built on Facebook in order to test the recommendation system and depicts different aspects of the application. It also points out some of the possibilities and limitations surrounding the application and its integration on a social networking site. Due to agreements with Ericsson, the code is not included in this rapport.

4.1 Application Purpose

The application as such was intended as a prototype. This means that it is limited in its functionality and design but works so that the users can interact with it and investigate its suitability (Preece et al. 2007, p. 530). The application was build mainly as a tool with which to evaluate the performance of the recommender system.

Previous performance tests had been done with a Jester jokes dataset and simulated social graphs which focused mainly on measuring algorithm scalability issues. Evaluating the algorithm's performance in the context of movies was done for the first time in this project, where the goals were to see how accurately the system predicted ratings by comparing these to the users' actual ratings and get an understanding of how the users' perceived their recommendations.

4.2 Final Prototype

The application was written in PHP 5 and developed in version 1.7.1 of CodeIgniter (CI). The latter is a framework used to develop web-applications[9] based on *Model-View-Controller* (*MVC*) architecture to separate the business logic from input and presentation. The application holds one model where all communication between the application and the recommender system is written, as well as nine controllers (each corresponding to one view) to invite, rate and select friends, rate and show movies, handle rated movies and show recommendations. The Controller in CI loads the files needed to process a request, including

[9] http://codeigniter.com

the Model and other resources like core libraries and helpers and the Views are then rendered and sent to the web browser to be shown (CodeIgniter 2009). The figure below shows how the framework structures the communication internally.

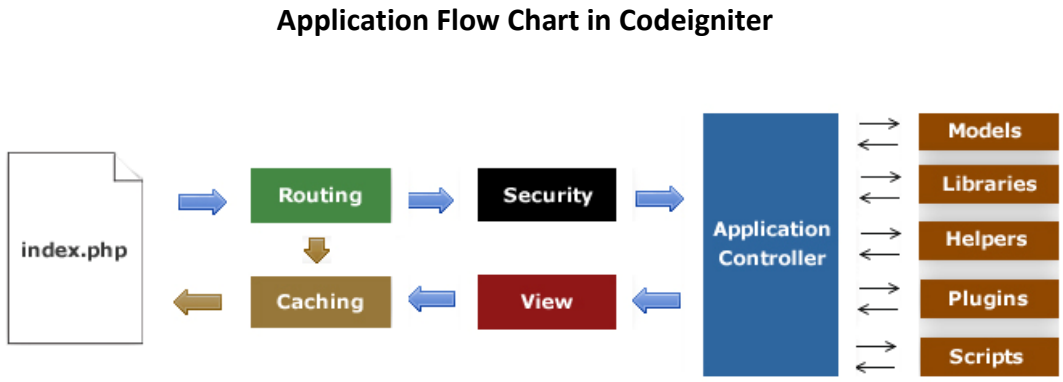


Figure 14) CI Application flowchart (http://codeigniter.com/user_guide/overview/appflow.html).

All information sent from the application to the system is stored in the recommender systems database and in a local database located on a different web-server used by the application. The application and the recommender system are located on different parts of the same server at Ericsson, separated from the Facebook servers as depict in the picture below. To ensure secure transfer, WinSCP was used to transfer the application files to its server. When a user adds the application, information about the user's facebook id is sent to the system. The application also sends information about the users' ratings and their friends' facebook ids and the recommender system then builds the social network using the knowledge of a user's relationships and computes predicted ratings for a user based on its knowledge of how the friends have rated.

The application communicates with the recommender system through *Representational State Transfer (REST)*, meaning that the application sends requests to a server that than returns a response. The application makes REST-requests to the recommender system through URLs to get lists of movies and to set data in the system. The application as such could be used on a separate website outside of Facebook, the Facebook specific functions would then have to

be replaced and users would have to be assigned a different identifier.

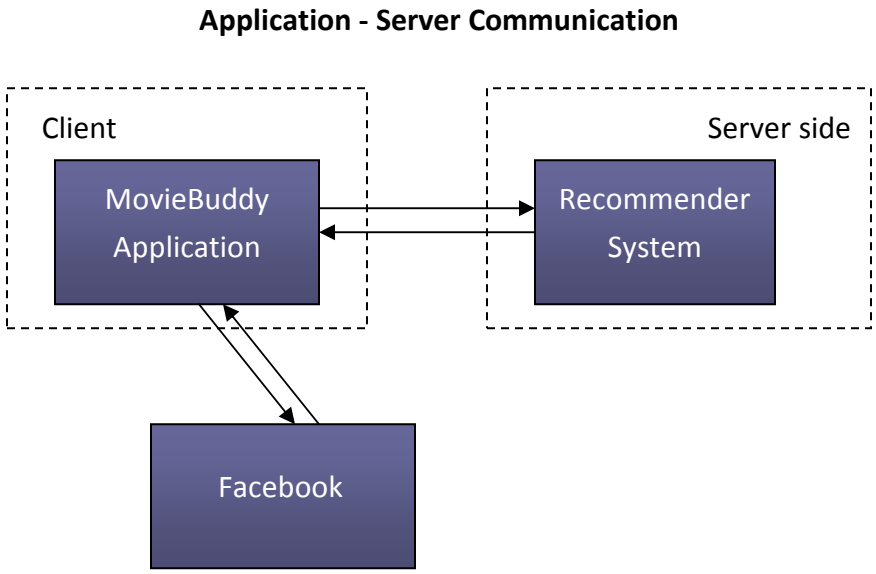


Figure 15) Application-server communications.

MySQL Query Browser was used to view the information stored in the systems database and phpMyAdmin to handle the local database. By storing the gathered ratings and information about the users trust, these could be compared to the systems predicted ratings for the same items and the inferred trust values for the same people in the social network. The results of these comparisons are described in the next chapter: *Chapter 5 User Tests and Results*.

4.2.1 Rate Movies

According to Golbeck (2005, p. 74) the average web user prefers a rating scale like 1-10 over one like 0-1. This has been considered in the application, where the scale for rating friends is from 1 to 10 and the scale for rating movies ranges from 1, 1.5, 2, to 5.

The "Quick Rate" view shows random items from the database, which contains 400 movies. Movies which the user has rated are displayed in the view "Rated Movies". If a user rates a movie in any of these views, the rating is sent back to the system and incorporated in computing recommendations. Based on the previous discussion regarding how people

choose to present themselves, a user's movie ratings will not be visible to anyone except the active user, as this is believed to result in more honest ratings than visible ratings would.

Rating a movie in the final application

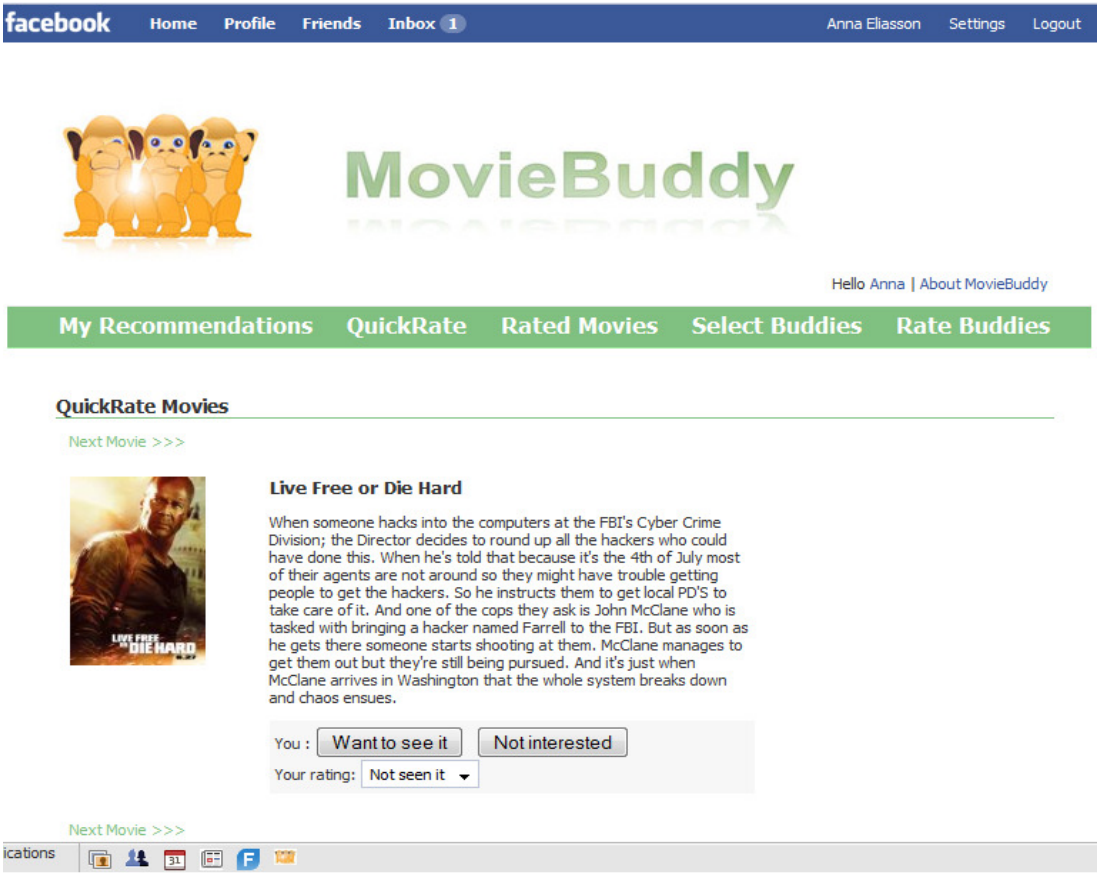


Figure 16) A print screen of the QuickRate view (http://apps.facebook.com/amovierecommender).

4.2.2 Select and Rate Friends

The friend selection is done in two steps; first, the user selects the friends that he or she wants as recommenders out of all the friends who have the application and second, the user gives the chosen recommenders trust ratings. A Facebook user must add the application before information about the user and the user's friends can be accessed and saved in the recommender system. Every time a user rates a friend, the system computes a trust value for

their relationship based on all the previously chosen friends in the user's social network. Information about a user's trust ratings of friends are not shown to anyone in the application other that the actual user in order to protect user privacy.

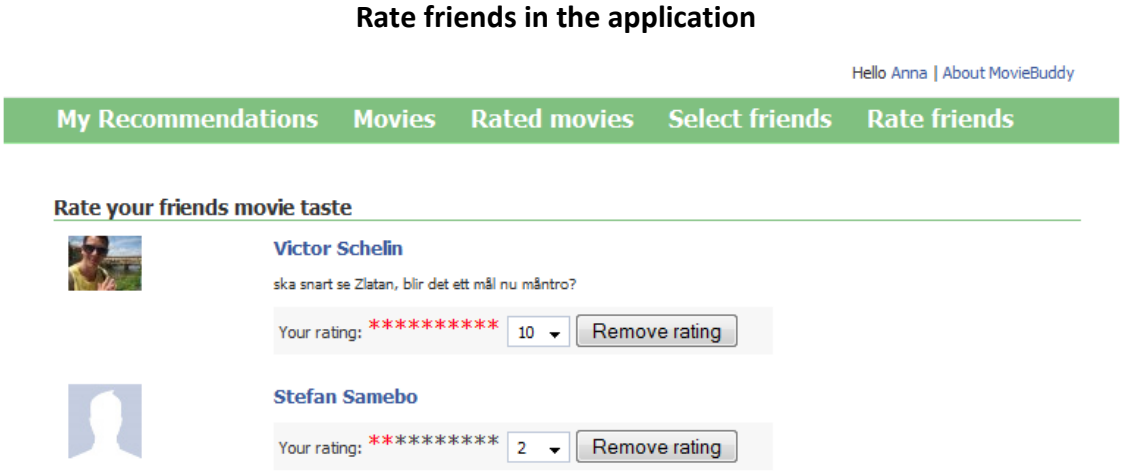


Figure 17) The Rate friends function in the final application.

4.2.3 Recommended Movies

As previously discussed in this chapter, users tend to rate towards ratings given by the system. It is important to the reliability of the evaluation that such influences are minimized and predicted ratings will therefore not be showed for the recommended items. When a user requests recommendations, the system considers all ratings provided by the user's movie buddies and computes a recommendation list. If some of the user's friends continue to rate movies, these new ratings are included in the systems computations the next time the active user hits "My Recommendations".

Given a list of recommended movies, there are four scenarios in which a user can act:

- The user has not seen the movie and does not wish to see it
- The user has not seen the movie and wishes to see it
- The user has seen the movie and rates it
- The user has not seen the movie and does not know whether or not it is worth seeing

The first of these two scenarios have been supported by a "Want to see"-choice and a "Not interested"-choice and evaluating these aspects have given an indication on how well the recommendation list was liked. The third scenario have been the basis for evaluation on how well the predicted ratings mirrored the users' actual ratings, by comparing the predicted rating and the user's actual rating. The feedback from "Want to see" and "Not interested" have been saved separately and not incorporated in recommendation computations. Numerical ratings from the "My Recommendations" view were however sent back to the system to be incorporated in future computations of recommendations. All feedback on recommended movies were saved in a separate table containing predicted ratings and actual ratings.

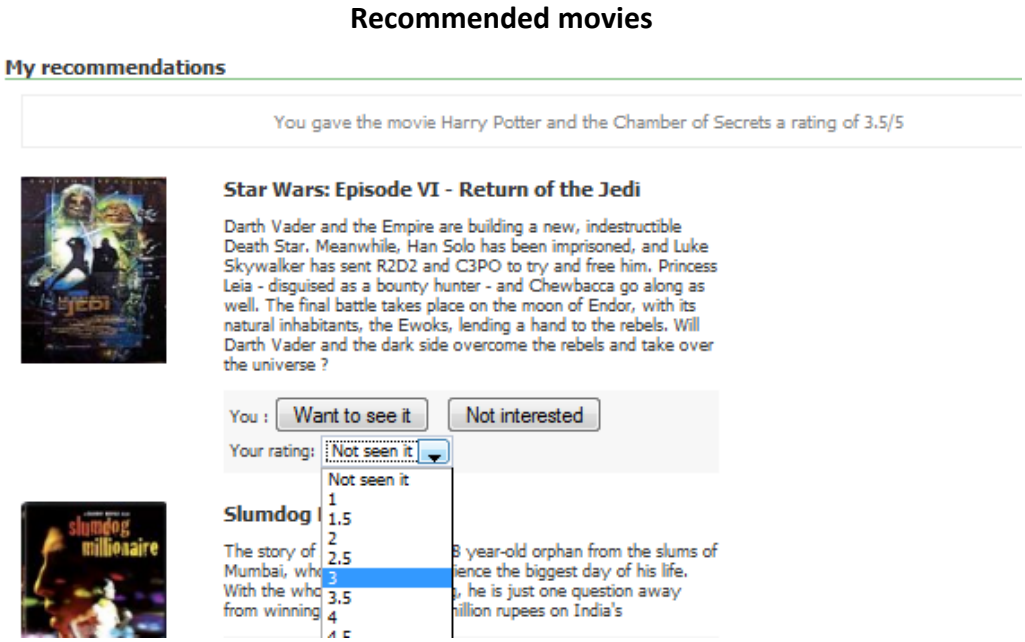


Figure 18) A list of recommended movies as predicted by the system.

4.2.4 Limitations

Developing an application on Facebook has certain characteristics. The most difficult one to navigate was the debugging process. When an error occurs, Facebook often show error messages such as: "This application is under development, come back later" which does not reveal much to the developer about what has gone wrong. To get information like a user's profile information and News Feed, the Facebook Markup Language (FBML) must be used.

A FBML-tag like fb:if-is-app-user provides an easy way of checking if the user has the application for instance before displaying information. The site also requires JavaScript to be transformed into Facebooks own version of JavaScript, FBJS. The basic syntax and functionalities are the same and use guides can be found on the Facebook Developer Wiki (Facebook, 2009b). Worth noting is also that applications cannot be hosted on Facebook, and the host server must be equipped with the Facebook client library and MySQL.

While the user tests included both English- and Swedish speaking users, the application has only been made available in English. Research has shown that users prefer to interact with systems in their own language which might want to be considered in further development (McNee 2006, p.120). With regards to user preferences, research has also shown that users have an easier time trusting recommender systems that incorporate some transparency with regards to why a certain recommendation was given (O'Donovan 2009, Golbeck 2005, p. 118).

Sinha & Swearingen (2002, p. 830) suggest that the importance of transparency for a user is dependent on the cost of a bad recommendation. Konstan et al. (1997, p. 78) preformed a cost-benefit analysis for recommendations in a set of different domains including movies. The analysis showed that recommending movies is a low-cost domain; while the value of consuming a desirable item might be high, the cost of consuming an undesirable item is basically only the money spent on the item (approximately €5 for a stream in Sweden) and the time spent before turning it off. Compared to recommending a restaurant for instance which can be far more costly. Additionally, the value of succeeding in keeping from recommending an undesirable movie is quite high, while failing to recommend a desirable movie will have low cost since there are many other desirable movies that can be recommended instead. In a high-cost domain, it might be more important to explain to the user why the system creates a certain recommender than it is in a low-cost domain such as movies.

In this project the choice was made not to put emphasis on transparency issues and reveal any information to the user about the recommendation process, other than to loosely

describe what the application aims to do[10]. This is however one of the user aspects which might be worth examining further in future work.

4.2.5 Possibilities

The possibilities of gathering information from social networking sites have not been examined in depth in this project. For instance, some social networks save user information such as favourite movies and TV-shows. On Facebook this information is made available to third-party applications (Facebook 2009d). This information might open up new possibilities for user profiling which should be investigated further.

Also, users of one social networking site are sometimes active on other similar sites as well. Implementing MovieBuddy on other sites could be helpful during further evaluation of the recommender system because it could expand the social network on which it is being tested. If the social network grows it will provide better opportunities for research like creating a web of trust. If a user adds the application on MySpace for instance, his or her social network could be mapped to the user's friends on Facebook. Golbeck (2005, p 22) has stated that users might wish to keep the information about their business live separate from their social life. A user that has connected the recommender system to several different platforms could always be given the choice to include friends from all social networks or to keep the networks separated. While an application in the context of movies might not be an interesting feature on a networking site devoted to creating and maintaining business connections like LinkedIn, it might be if it were for other domains such as literature or restaurants.

The average Facebook user has 120 friends (Facebook 2009c). If all of these friends were to add the application, their friends and their friends as well (creating a depth of three) the social network could contain more than 1.7 million people (120^3). While this scenario might not be likely yet, it might be an important aspect to consider with regards to future scalability evaluations.

[10] http://apps.facebook.com/amovierecommender/index.php/welcome

5 User Tests and Results

In order to answer the thesis project's questions at issue, data was gathered from user tests on the recommender system and a survey covering the users' opinions. This chapter contains the result and analysis of these tests. The initial user test is first described, followed by the findings from the main user test and the results from the survey.

5.1 Initial User Test

An initial user test was conducted with the main goal of noticing any bugs or problems in the application that could affect the outcome of the main user test. The chosen users were all fourth or fifth year students at the Royal Institute of Technology, one woman and three men in the ages from 24 to 27. The idea was to choose a set of users that matched the demographic of the participants in the actual user test as this is a recommended approach for any pilot test (Bell 2004, p. 149).

The initial users were chosen also because they were all familiar with usability and interaction design. They were as such thought to be able to detect a majority of the errors or problems in little time, as they most likely would know where to look at what to look for.

The tests lead to several changes in the presentation of movies in the "QuickRate" and "My Recommendations" views. Two of the users also felt that a way to invite friends was missing which was later implemented. Other desired functionalities (as described in *Appendix A*) that were suggested but not implemented were: searching for specific movies, enabling explicit recommendations to friends and making friends movie ratings visible.

5.2 Main User Test

Instructions for the user test were integrated in the applications menu as shown in *Appendix B*. The users were asked to first rate eleven movies, making them eligible as recommended items to friends. They were then asked to select any number of friends as recommenders and give all the chosen friends trust values denoting how much their opinions would matter in the recommendation process. The friends that they could choose from were all of their

friends on Facebook who also had added the MovieBuddy application. The next step was to look through a list of 20 recommended movies and rate the ones they had seen or click "Want to see it" or "Not interested" on the ones they knew about but had not seen.

The last step of the user test was to fill in a survey with the purpose of generating more insight into how the users felt about the recommendations and the recommender system in general.

5.2.1 User Demographic

The application was distributed to 185 individuals (the developer's social network on Facebook and a list of Ericsson employees) and of these 41 added the application. From these there was a decline in users who actually decided to participate in the user test, bringing the total to 34 active users. Out of the participants, little over 32% were female and close to 68% male, compared to the distribution of survey respondents where 40% were female and 60% male – indicating that women were somewhat overrepresented in the survey in comparison to the application tests.

5.2.2 Movie Dataset and Distribution of Ratings

The system's movie database consisted of 400 movies. To compose the list, three different top-rated movie lists from the Internet Movie Database (IMDb.com) were fused together: *"Most Searched-for '2000s' Titles in last 7 Days"*, *"All-Time Worldwide Box office"* and *"Top Rated '2000s' Titles"*. The goal was to create a list of well-known movies that could generate many ratings from the users.

A total of 1060 ratings were registered in the system. *Figure 19* illustrates the spread of all ratings excluding "Want to see" and "Not interested". It shows that users tend to rate movies higher rather than lower. The users average rating was computed to 3,2.

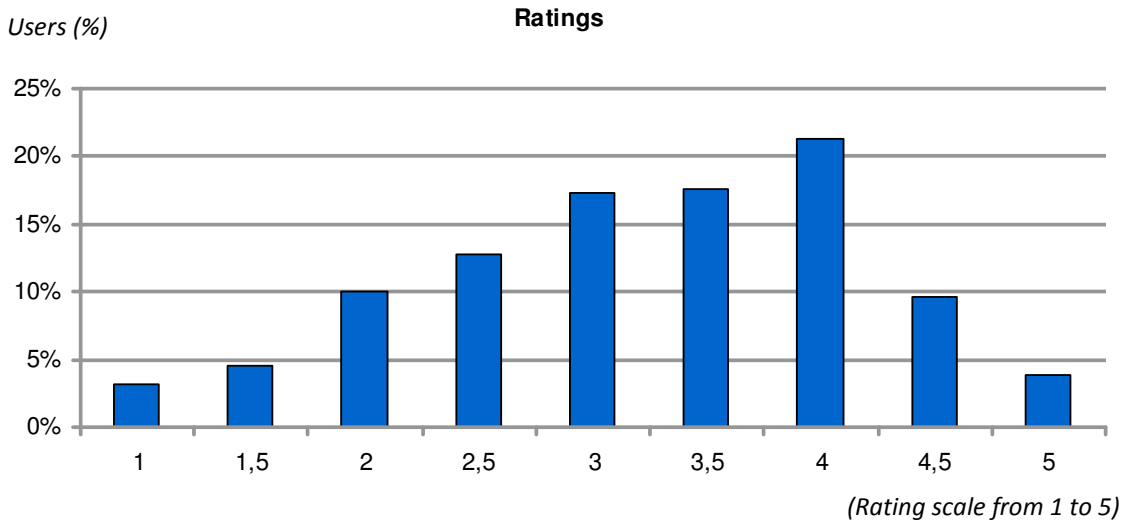


Figure 19) Distribution of user ratings.

5.2.3 Distribution of Ratings for Recommended Movies

Some users rated more movies in the application than others did. As previously mentioned, the users were asked to rate movies in their recommendation lists. Close to half of the users chose to follow through with this. These users stood for 79% of the total ratings registered in the system, which corresponds to an average of 49 movies each, while the users who did not rate movies from their recommendation lists created a total of 21% of all ratings in the system, which corresponds to an average of 13 rated movies each. This could imply that the users who rated movies in their recommendation lists were similar to each other for instance through a shared interest in movies, recommender systems in general or the MovieBuddy application in particular.

The average rating for the users who rated movies in their recommendation lists was 3,3 (0,1 points over the average that included all users). *Figure 20* depicts the ratings from the recommendations lists. -2 and -1 in the figure represent "Want to see" and "Not interested". The latter of these can be viewed as a higher rating than average and the first as a lower rating than average. The results then indicate that 73% of the ratings were over average, compared to all ratings registered in the system where 52% were over the users' average. This means that the users rated the movies in the recommendation lists higher than they rated movies in the application in general.

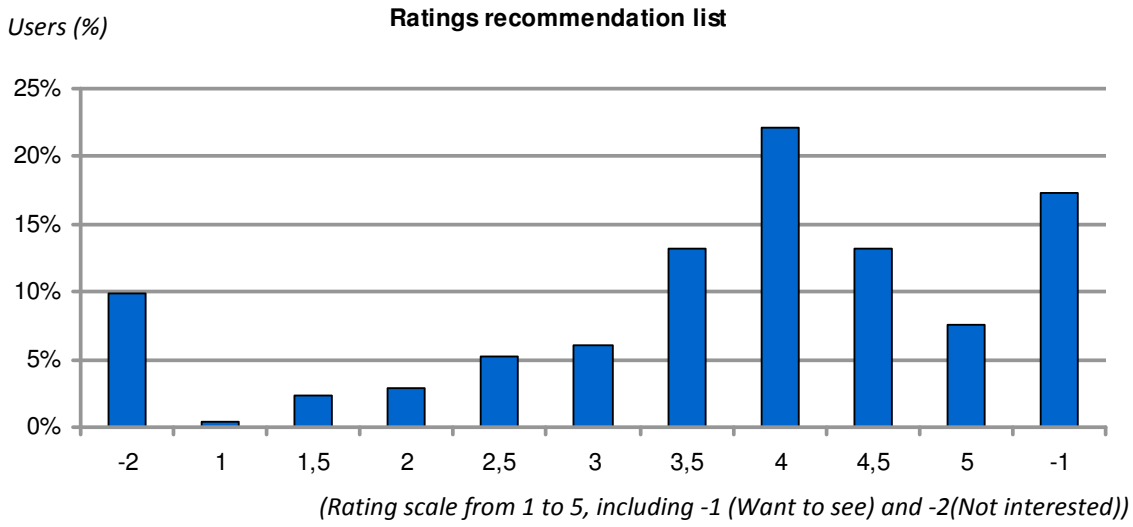


Figure 20) Distribution of ratings of recommended items.

5.2.4 Accuracy of Predicted Ratings

As illustrated in *figure 20*, less than 6% of all recommended movies diverged more than one point under the users' average. 10% were rated with "Not interested", which suggest that they were not suitable as recommendations. The premise has been that the users' actual ratings are correct and that the predicted ratings should mirror these as accurately as possible. The predicted ratings for the recommended items in *figure 21* below ranged between 3 and 4,5. "Not interested" was viewed as an actual rating of 1. This means that if a user noted that he or she was not interested in a recommended item, a low prediction would be considered accurate.

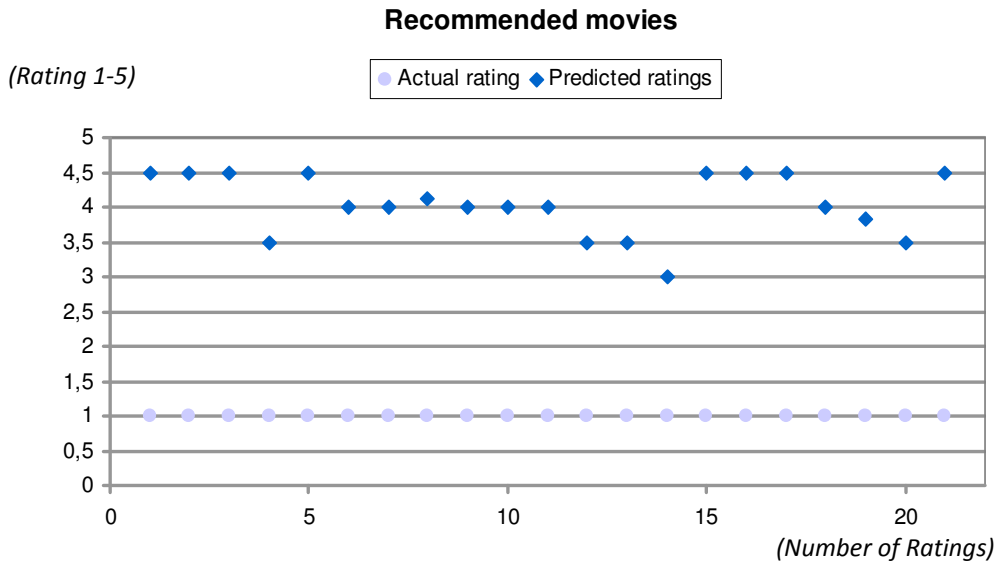


Figure 21) The predicted ratings for all recommended items which were denoted "Not interested" by the users.

21% of the movies got a rating higher than one point over the users' average rating. Additionally, 17% of the recommended items were denoted as "Want to see" which indicate that they were appropriate as recommendations. These ranged between 3,5 and 4,5 as depict in *figure 22*. "Want to see" was viewed as an actual rating of 5, which meant that a high prediction would be considered accurate.

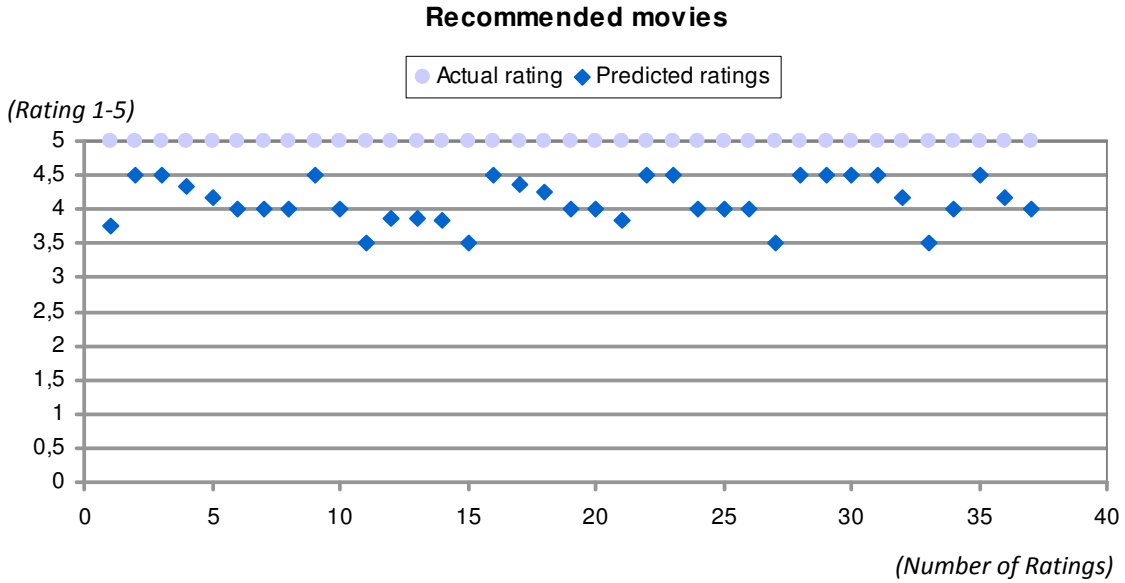


Figure 22) The predicted ratings for all recommended items which were denoted "Want to see" (rating 5) by the users.

To further measure the accuracy of the recommendations, all actual ratings for recommended items were compared to the predicted ratings. A total of 213 ratings were registered from the recommendations lists which translate to 20% of the total amount of ratings. "Want to see" and "Not interested" were again transformed to ratings 5 and 1 respectively in order to be quantifiable. The mean absolute error between predicted and actual ratings was calculated to approximately 0,93 points with

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|r_i - p_i| \tag{5.1}$$

Where r_i is the actual rating for item i, p_i is the predicted rating and n is the number of items compared. This means that for a predicted rating 4, the user would rate the item somewhere between 3 and 5. *Figure 23* shows the difference between predicted ratings and actual ratings for all recommended items which have been rated by the users. Each point in the figure represents the difference for one item.

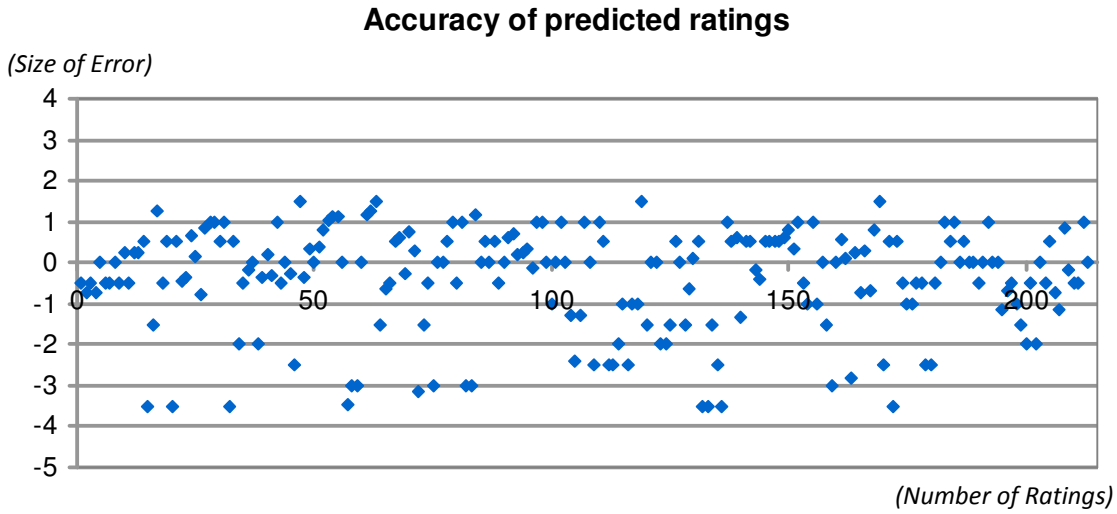


Figure 23) Difference between predicted ratings and actual ratings.

As illustrated in the figure, the majority of differences (47%) were negative which mean that the predicted rating were larger than the actual rating. In 39% of the cases, the actual rating was higher than the predicted and in 14% there was no difference. The maximum positive error was 1,5 points and the percentage of errors with a deviation of more than +1 point was 5%. The maximum negative error was -3,5 and 26% of the errors deviated more than -1.

This suggests that the predicted ratings tend to be higher than the users actual ratings more often than the other way around and when the actual rating is higher, the difference is less noticeable.

In order to see if the mean absolute error has changed over time, it was computed for the first 100 and 40 ratings respectively. The first showed a MAE of 0,88 and the latter 0,83. These suggest that the MAE has not changed significantly over time. To appreciate the statistical significance of the results, the confidence interval was computed for all of the mean absolute errors. Since the examined population was rather small, the t-distribution[11] was used to account for the uncertainty. The true mean was calculated as

$$\overline{X} - (t)\frac{S_x}{\sqrt{n}} \leq \mu_x \leq \overline{X} + (t)\frac{S_x}{\sqrt{n}} \tag{5.2}$$

Where $\overline{X}$ was the mean absolute error, *t* the *critical t* as computed with 213 data points and a confidence level at 95%, S_x the sample standard deviation of 0,89 and *n* the number of data points. The confidence interval was calculated to $0{,}93 \pm 0{,}12$. This suggest that there is a 95% likelihood that the true mean absolute error for the entire population will be somewhere between 0,81 and 1,05. The confidence interval for the first 100 ratings was found to be about $0{,}88 \pm 0{,}18$ and $0{,}83 \pm 0{,}29$ for the first 40 ratings, meaning an interval of 0,7-1,06 and 0,54-1,12 respectively. These indicate that while the MAE has increased, the maximum error has decreased.

No systematic errors seemed to be generated by the algorithm. Based on the results, the recommendations were however more accurate for some users than others. 38% of the users had a MAE of more than 1. Of these, a third had rated less than 4 recommended movies leaving a total of four users who had rated between 10 and 22 recommended movies each. Discarding all users who had rated less than 10 recommended movies gave a total of 3 users with a MAE close to 1 and 5 of 0,76 or less. Breaking down the data to this degree leaves to little information to draw any definite conclusions regarding why the accuracy was different between users. With this in mind, it was noted that for three users, the inferred trust values

[11] Vännman, K. (2002). Matematisk statestik. 2ed. Lund: Student literature.

corresponded well with their actual trust ratings and two out of these three had the lowest mean absolute errors (0,41 and 0,67, respectively). Nine out of the ten users with lower similarity between actual trust and inferred trust had higher mean absolute errors (0,72 and over). This could suggest that predicting ratings based on inferred trust values that do not correspond well to the actual trust could decrease the quality of recommendations. Again, there is too little data to draw such conclusions with certainty and these groups could preferably be evaluated in the future with more users in the system and more recommended items rated. A compilation of figures comparing the different users' average ratings with predicted and actual ratings is included in *Appendix C*.

The inferred trust values – while they differ between users – tend to be the same for all direct relationships that a given user has. For instance, one user gave friend *a* trust rating 5 and friend *b* rating 10, but the system gave them both a 1 in trust. Another user had six direct relationships ranging from 5 to 8 and they were all the inferred to 0,67. This indicates that the algorithm fails to personalize the trust ratings.

5.2.5 Distribution of Connections

62% of the users, which corresponds to 21 users, added friends as movie recommenders and created 110 connections altogether. 55% of these were direct connections and 45% transitive relationships inferred by the system. As depict in *figure 24*, most users chose between 4 and 6 friends as recommenders.

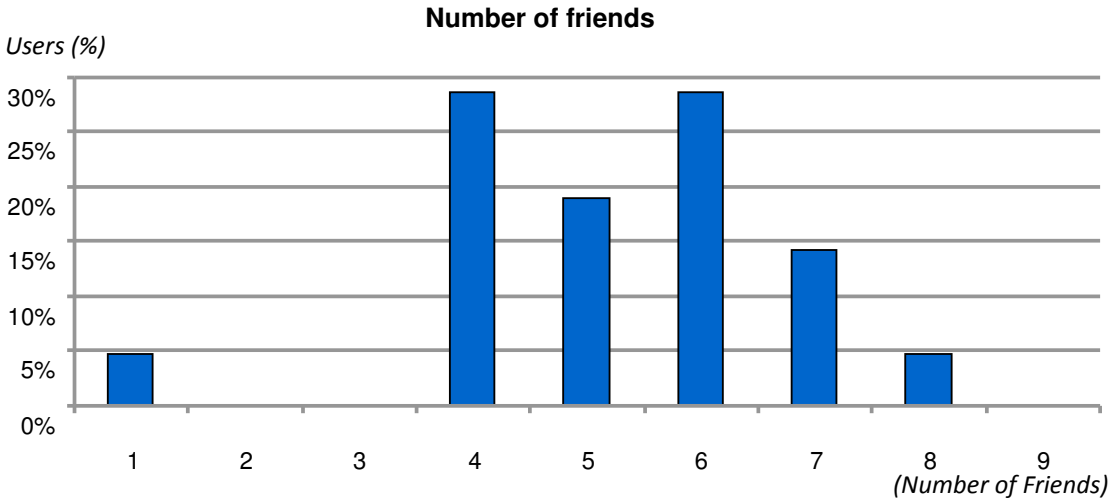


Figure 24) Number of friends the users chose as recommenders including the transitive relationships that the system found.

5.2.6 Accuracy and Distribution of Trust Ratings

To measure the accuracy of the inferred trust values, the users' actual trust values were compared to the inferred values for direct relationships (transitive relationships were excluded). In order to compare the values, the actual trust ratings were transformed from 1-10 to scale 0-1.

61% of the inferred values showed an absolute difference of 0,2 or lower. This means that in the case of a user noting the value 6, the system inferred a value between 4 and 8 in the in the majority of cases. In 33% of the cases, the difference was higher than 0,25. The mean absolute error for all direct trust values was found to be 0,22 on a scale of 0-1. This means that if a user gave a friend a trust value 6, the system would on average infer a value between approximately 4 and 8.

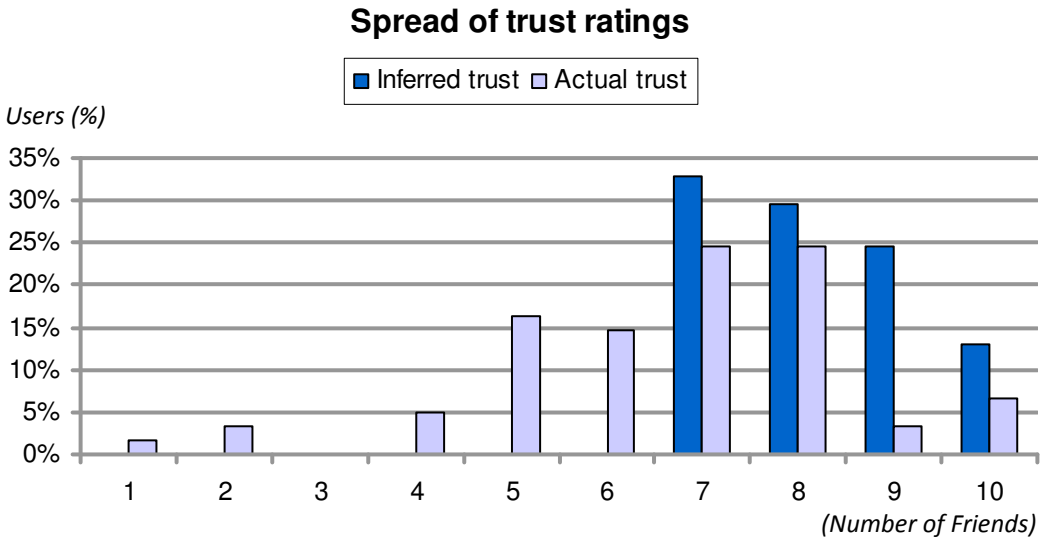


Figure 25) Spread of trust ratings, inferred and actual.

Figure 25 shows the spread of inferred as well as actual trust ratings. To visualize the inferred trust ratings in comparison to the actual trust ratings they were translated to the scale 1-10. The transitive relationships have been excluded as they denote relationships that the users have not rated. As seen in *figure 25*, 100% of the inferred ratings are between 7 and 10 compared to 59% of the actual ratings. Since the premise has been that the users' actual ratings are correct, this indicates that the system has viewed direct relationships as stronger than they have been in actuality. As with the movie ratings, users tend to give higher ratings rather than lower. As previously described, the users filter out the friends they want as movie recommenders and rate only the chosen ones. This might be the reason why the majority (74%) of all actual trust ratings are 6 or higher. If the users instead had rated all of their friends who had the application, the ratings might have looked different.

Figure 26 shows the spread of both the transitive and the direct trust values computed by the system. As denoted by the scale, some of the transitive relationships were set to zero. 71% of all transitive relationships had a value of 3 or 4 and 87% of all direct relationships were set between 7 and 9.

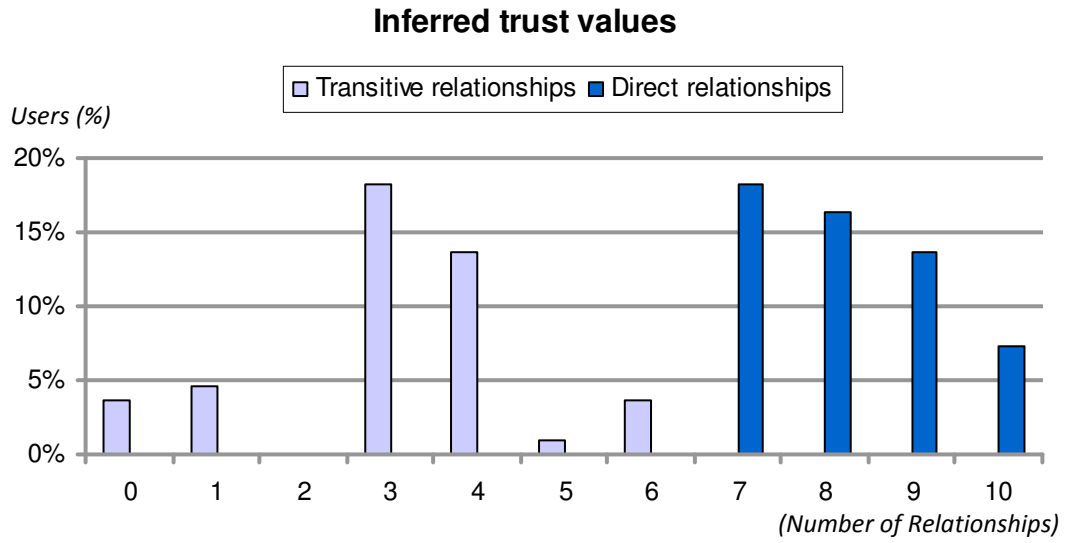


Figure 26) All the trust values inferred by the system.

A confidence interval was computed for the mean absolute error. Since the examined population was rather small, the t-distribution was used to account for the uncertainty. There were 61 direct relationships to consider and the confidence level was set at 95%. The confidence interval was found at $0{,}22 \pm 0{,}04$. This suggests that there is a 95% confidence that the true mean will be somewhere between 0,18 and 0,26. To appreciate if the accuracy has changed with the number of relationships in the system, the confidence interval was computed for the first 40 and the first 20 direct relationships. The first showed a mean absolute error of 0,21 and a confidence interval between 0,16 and 0,25. The latter showed a mean absolute error of 0,26 and an interval from 0,19 to 0,33. These indicate that the average mean has been somewhat stable.

In order to appreciate how the system would act if the recommendations were randomized, the predicted ratings were replaced by randomized ratings. A mathematical randomization function generated random numbers between 0 and 5 which in turn gave an mean absolute error of 1,89. The difference for random accuracy is illustrated along with the current algorithm's difference in the figure on the left below, larger difference means lesser accuracy. The figure on the right shows a concrete example; if a user *a* has given a movie rating 3, the algorithm will create on average a prediction almost between 2 and 4, whereas a random

"prediction" will range between 1 and 5.

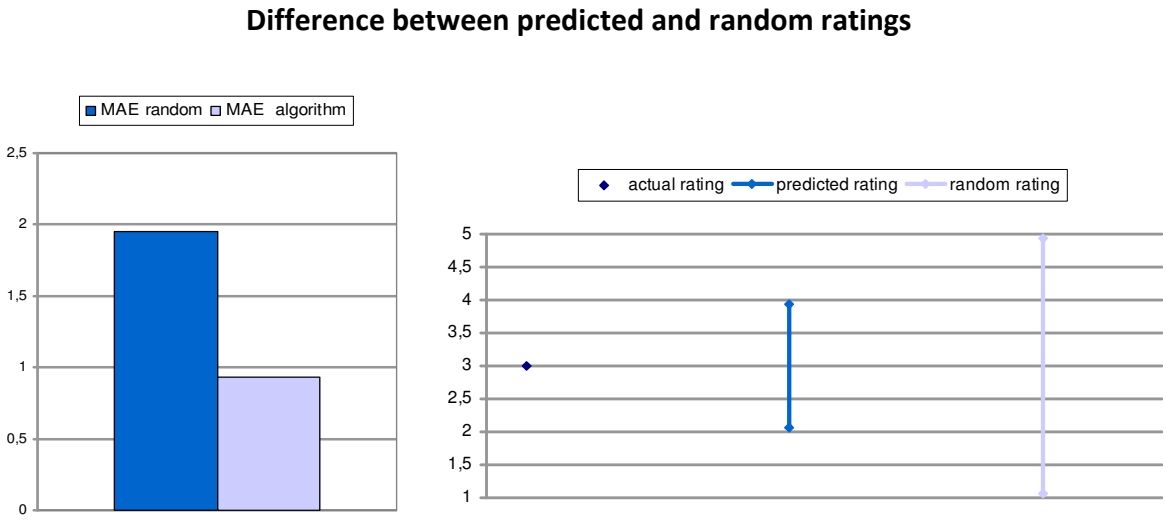


Figure 27) The figure on the left shows the difference in accuracy between the algorithm and random predictions. The figure on the right illustrates an example of how accurately the algorithm can predict a rating compared to a random function could.

5.3 Survey

10 of the user test participants chose to answer the online survey (included in *Appendix D*). This meant a response rate close to 30%. Several factors might have contributed to this low participation; the users were not given any incitement other than goodwill to finish the survey which was the last step in the user test and they might have felt that they had participated enough by using the application. First and foremost however, many technical issues which were not anticipated arose in the recommender system. As programming the system server side both lay outside the scope of the thesis and this developer's programming ability, these issues had to be met by Ericsson employees with limited time at their disposal. As a result, there was several weeks delay before the user tests could be initiated and they could therefore not be conducted during as long a period as planned.

Since the number of respondents was quite low, the results of the survey cannot be viewed as a representation of an entire population but rather as indications on what the users thought. It should be considered that the people who answered might have been similar in

type, they may for instance have had a shared interest in recommender systems, and that the answers therefore reflect a certain kind of user.

5.3.1 Demographic

The first two questions were regarding gender and age and were asked to get an understanding of who the users were. According to the answers, 40% of the respondents were women and 60% men. Of these, 30% were in the ages 16-24 and 70% in the ages 25-30.

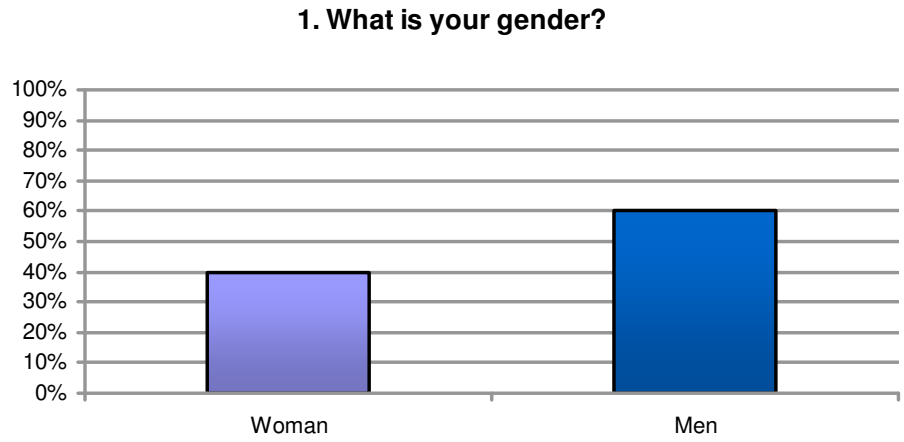


Figure 28) Gender distribution of the survey respondents.

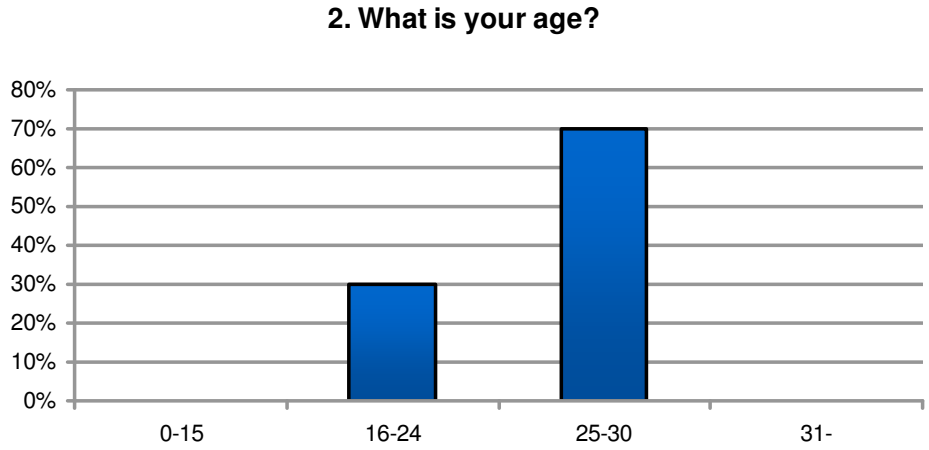


Figure 29) Age distribution of the survey respondents.

5.3.2 Familiarity with the Movies in the Application

Question three was regarding how familiar the users were with the movies shown in the application. On a scale of 1 (not at all familiar) and 5 (very familiar), 80% answered 4 or 5 and 20% answered 3. Question four asked about their familiarity with the movies shown in their respective recommendation list. 60% stated a 4 in familiarity, 30% noted a 3 and 10% said 2. As previously mentioned, 27% of the recommended movies were rated either "Want to see" or "Not interested". The other 73% were movies which the user had already seen. These results suggest quite a high overall familiarity with the recommended movies.

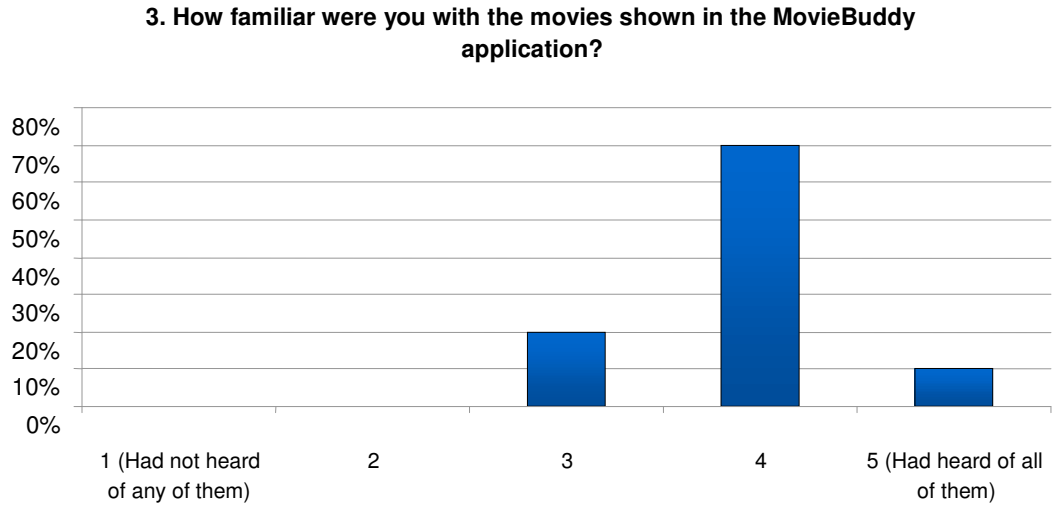


Figure 30) Chart over answers to question number 3 regarding overall familiarity with the movies shown in the entire application.

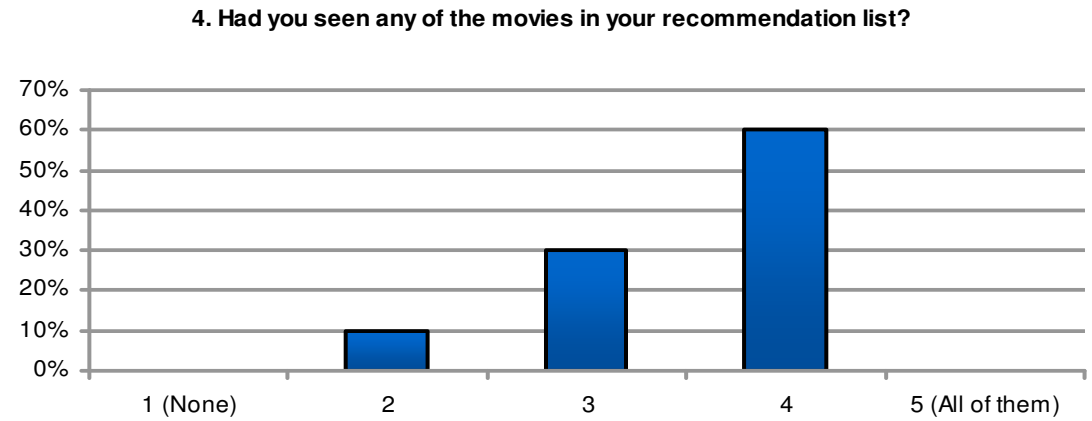


Figure 31) Chart over answers to question number 4, familiarity with the movies shown only in the recommendation list.

5.3.3 User Perception of the Recommendation Lists

Question five asked the users how much they liked their list of recommended movies on a scale of 1 to 5, where 1 was "Not at all" and 5 was "Really appreciated it". According to the survey, 40% of the respondents rated the list 4 or 5, which indicate that they appreciated the list. 20% gave the list a rating 2 and 30% rated it 3. Question six asked if the recommendations had got them interested in any new movies, to which 90% stated that it did and 10% that it did not.

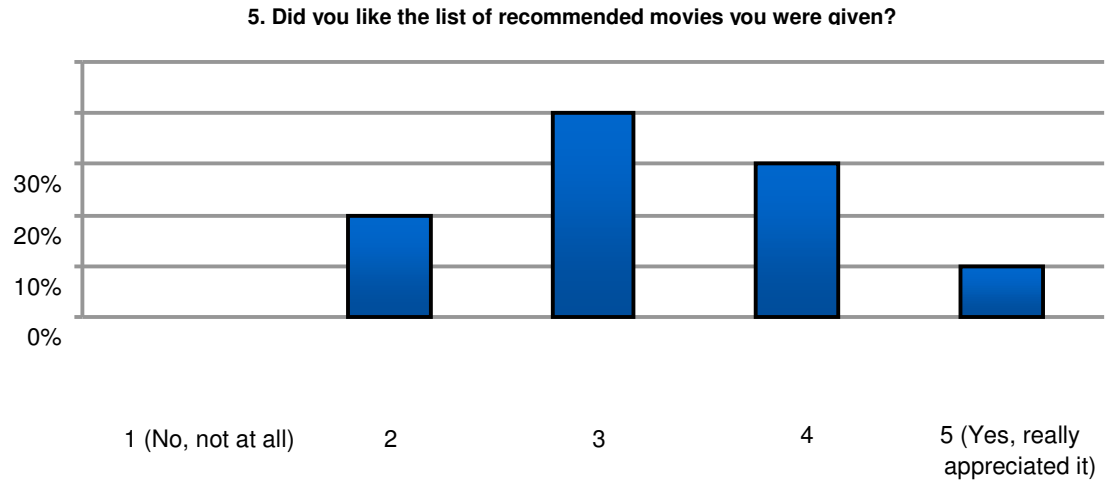


Figure 32) Chart illustrating the answers to question number 5, weather or not the respondents liked their lists of recommendations.

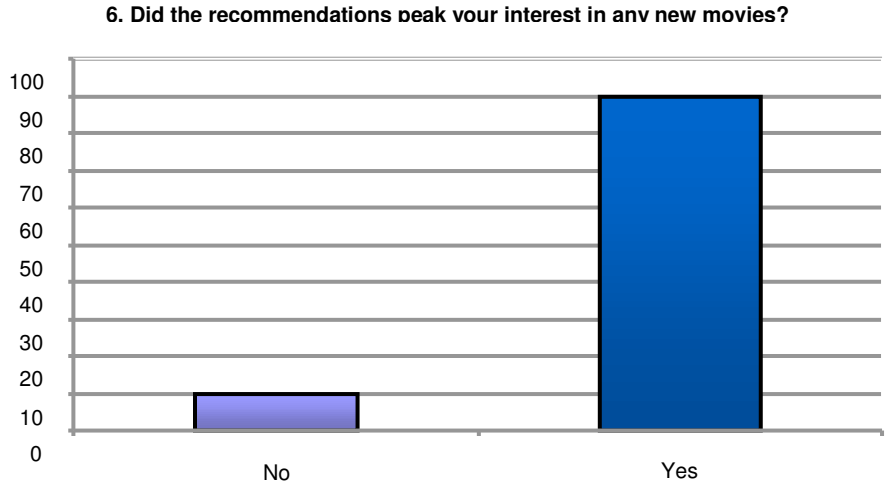


Figure 33) Chart visualizing the answers to question number 6, which asked the respondent if the recommendations peaked his or her interest in any new movies.

Questions 9 and 10 were about recommendation list characteristics. The questions were how novel and bold the user thought the recommendations were on a scale of 1 to 5. High novelty means unexpected recommendations; for instance movies that they would never expect a friend to recommend. Bold is the opposite of safe; for instance something very dissimilar to the type of movies that the users themselves had rated in the application. 30% of the respondents thought the recommended items were more novel than boring while 20% had the opposite view. The rest of the participants, 30%, gave the recommendations 3 out of 5 on novelty. 20% of the respondents thought the recommended items were more bold than safe, 10% were of the opposite opinion and 70% rated 3 out of 5.

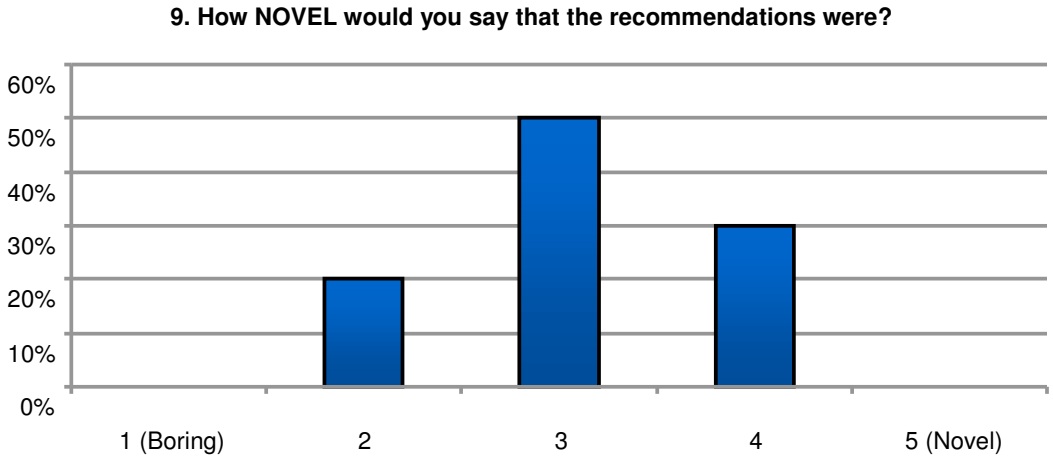


Figure 34) Novelty of recommendation list.

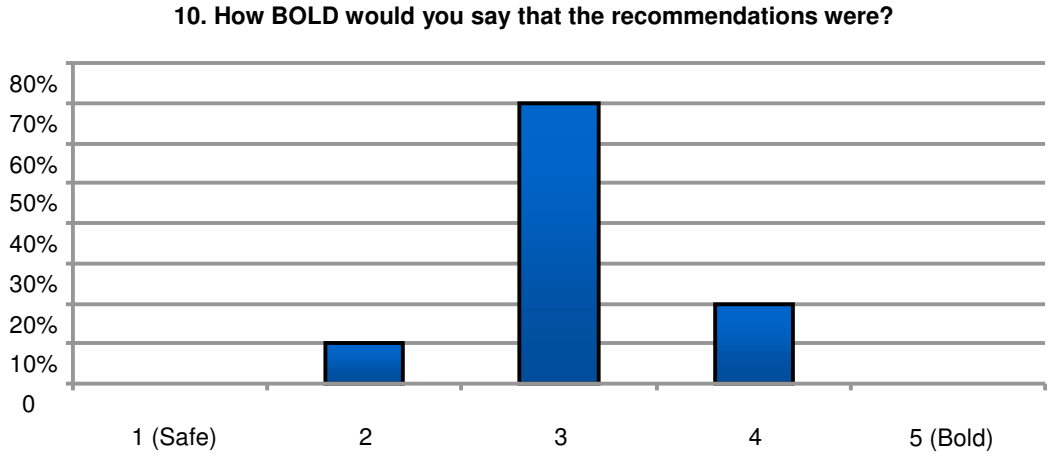


Figure 35) Boldness of recommendation list.

The fact that the majority of participants rated 3 on both novelty and boldness might suggest that these types of characteristics were not apparent for the recommendations or that these concepts were less intuitive than the other questions.

Question number 8 asked the respondents about their overall perception of the recommendation list. There was a 20% decline in answers to this question which contained both good and bad critic. Someone suggested that the list was too long and that 10 movies would have been enough (it displayed 20 movies). Another note was that the list took too long to load. This tendency was noticed during development and had a lot to do with the server that held the application at Ericsson.

One respondent wrote regarding the list: *"It was very good. I have an account at Lovefilm.se and their recommendations list is very flawed, especially compared to this one. And here I had only graded around 20 movies compared to over 200 at Lovefilm, so either the algorithm is really good, or I have friends with really good taste in movies."* Another respondent wrote that while the application did not peak his interest in any new movies, he did find some movies that he had seen but forgotten about – *"[which] is always nice"*. A suggestion similar to one previously made during the initial user test was to make it possible to search for and sort amongst movies in the application.

5.3.4 Regarding User Tasks

With Human-Recommender Interaction theory in mind, some of the questions were aimed at trying to identify the users and the user tasks they felt a recommender system for movies should support. The above answers have given some indications towards both who the users are and some of the tasks that they wished the application would support. Question 7 gave the users a list of different user tasks to choose from, and asked what they were looking for in a personal movie recommender. The users were allowed to choose as many alternatives as they pleased, *figure 36* illustrates the answers.

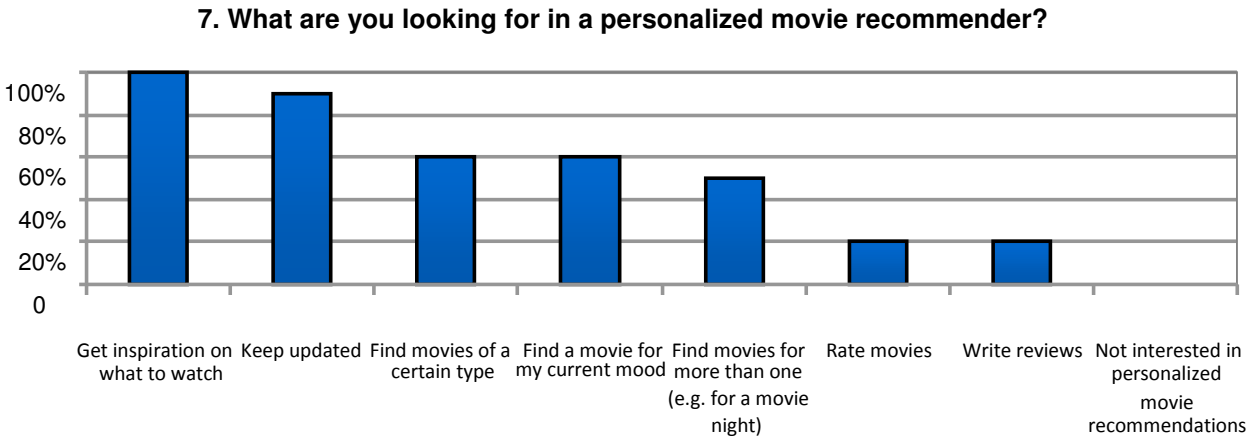


Figure 36) Chart illustrating the answers to question number 7, describing what user tasks the users find most important in a movie recommender.

The most popular user tasks were *"Getting inspiration on what to watch"* that was chosen by 100% of the users and *"Keeping updated on what is new"*. The user tasks are explained in more detail in *Chapter 3.3*. The MovieBuddy application sought to support the first of these user tasks. The results indicate that the users are interested in more features than the ones implemented so far.

5.3.5 Overall Perception of the Application

The final question in the survey was regarding the users overall perception of the MovieBuddy application. There was a 30% decline in the answers to this question. Some suggestions were made towards design and appearance. One user wished that more information were made available such as cast lists, length of movie and so on, especially for movies which the user had not seen. The information displayed for each movie was restricted to image and plot information if these were available. Another user commented on the fact that the length of information varied quite a lot between movies, some with no plot information and others with quite long texts.

The main points are summarized below. Swedish answers have been translated and some of the language rectified in places to form coherent sentences here in order to simplify reading.

The original answers are included in *Appendix C*.

11. What was your overall perception of the MovieBuddy application?

"Good idea, especially is you know the taste of your friends well"

"Pretty nice, with some work and some more functions it will be a really good app!"

"Was kind of surprised of the movies I got recommended. I'm not into action movies usually..."

"It was pretty good. It was a bit slow at times, which was a bit annoying. The recommendations list was very impressive however, making the app well worth using. But as expected in the beginning of rating movies, you mostly get movies you have already seen and liked, but hopefully this will change as you rate more and more of the movies already seen."

"I think it would be more useful if I had more friends to rate."

"Good idea, well thought out. It could be even sharper and give even more suited recommendations"

6 Conclusions and Discussion

The purpose of this thesis project was to evaluate how well an algorithm in a recommender system predicts movie ratings, how the user's perceived the recommendations and answer whether or not it should be revised. The goal was also to build an application on a social networking site. The project was based on three main questions: *How well does the recommender system predict ratings? Can the existing algorithm be revised in order to optimize the recommendations? And is a recommender system based on social networks valuable?* Based on the results, some factors have been identified as key for the development of the recommender system. These are described in this chapter.

6.1 Conclusions Summary

- *Users liked getting recommendations from friends.* The users were overall positive to using friends as recommenders and rated the movies in the recommendation list higher than they did anywhere else. The recommendation list did get the users interested in new movies, but the users still felt that there was room for improvement when it came to the recommendations.

- *The trust inference process is not personalized enough.* The actual trust ratings are more diverse that the inferred trust ratings are – which tend to be the same for all direct relationships. Accuracy of the predicted ratings can get better. Using clique count to find the best movie recommenders in a social network is not optimal. It finds the social butterflies and might be more valuable in a social network where no one knows who to trust. Explicit trust ratings are easy to use instead in a social network of friends.

- *The recommender system in general and the algorithm in particular can improve.* By using the users actual trust ratings when predicting ratings and by recognizing that there are different types of users that want different things out of their recommender system the system can improve. The algorithms should then preferably be measured on more metrics (inspired by HRI metrics) than accuracy.

- *By using the prototyped Facebook application MovieBuddy, further evaluations can be done live with*

users and different algorithms. Social networking sites also hold potential for profiling users and, if the application was connected to several sites, extracting and building larger social networks. These sites are also easily accessible test beds.

6.2 Main Issues

6.2.1 Users and User Tasks

A question which was not included in the survey was how much contact the users had had previously with recommender systems. This relates to what type of users the system can expect to support. It is useful to know ones users since inexperienced and experienced ones may have different requirements. An inexperienced user might for instance have the same high requirements on *Trust* that an experienced user has on *Personalization*.[12] Identifying the users and user tasks that the developers want their recommender system to support could lead to better recommendations and a better defined recommender system.

The answers to the question regarding the user's overall perception of the MovieBuddy application indicated that most of them though that the idea of personalized movie recommendations was quite novel, suggesting that they have had little experience with similar recommender systems. While no efforts further than this were made to try and categorize the users, the answers did indicate that the majority of users recognized most movies in the application. Since the movies were gathered from three lists of popular movies, the level of familiarity was expected to be quite high. Had the results instead indicated low overall familiarity, the conclusion could have been that the user type in question did not watch that much film. The results have instead indicated a user type that watches a lot of popular movies.

According to the survey, the two user tasks which were least popular were *Rating movies* and *Writing reviews*. These are basically the two tasks were the users give back to the system. Perhaps a system that mines information about its users from their social network profiles could be useful as a complement in a system where explicit user input is sparse. The most

[12] *HRI Aspects*, described in *chapter 3*.

important user tasks according to the users were *Getting inspiration on what to watch*, *Keeping updated on what is new, Finding movies of a certain type or for a current mood* and *Finding movies for more than one (like for movie night)*. These are all different types of search behaviours. The first two could be supported by a search function or scrolling through lists of movies in different categories (new movies for the second one) while the next two requires filtering functions based on movie properties and the last one requires that more than one profile can be cross referenced in the system.

While the instructions for the main user test clearly stated that the user should give feedback on the recommendations list by rating the movies he or she had seen, wanted to see or was not interested in, only 39% of the users chose to follow through with this. It is unlikely that all of the users who did not rate movies from their recommendations lists were unfamiliar with all of their recommended movies. The reason why so many chose not to give feedback on the recommendation list could be one of several; while the purpose of rating movies from the QuickRate view was to get better recommendations, the purpose of rating recommendations was to help in an evaluation process. A user may have felt that it was more rewarding to see how good recommendations he or she could get, than to help some evaluation. Also, the users were asked to rate 15 movies which might have lead to them feeling feed up when it came to rating the recommendation list.

The same idea could be applied to the issue of low participation in the survey. Again, this lead to results where their answers could not be quantifiable, but only give indications of how the users felt. Had the application been up and running as planned, the number of respondents would most likely have increased. It would however have been a good idea to give the users more incentive; one possibility could have been to give them movie tickets or something similar.

The behaviours described above indicate that a user feels that the purpose for visiting a recommender system is not for them to give explicit input to the system, but for the system to give explicit output to them. Based on Human Recommender Interaction theory, users' requirements and tasks are different depending on the context for which recommendations are being made. Thus, the system might need to support different aspects depending on what domain it is making recommendations for and use different metrics to evaluate system

performance. This is worth recognizing before applying the recommender system to other domains.

An important step to take is deciding what type of recommender system should be developed; more user tasks than the ones suggested in this project could be identified and so far, no particular demographic has been chosen for the system. The existing algorithms could then also be measured with the many different metrics found in Human Recommender Interaction theory.

6.2.2 Friends as Recommenders

According to the results, the trust values were a bit of the users actual trust values with a MAE of 0,22 on a ten-point rating scale from 0 to 1. The results showed that the system tend to, for a given user, reward the same trust value to all direct relationships while the transitive are more diverse. These factors add up to the conclusion that the trust inference part of the system is faulty and puts more emphasis on how far from the active user another user is and not enough of the differences between the ones on the same distance. Computing cliques like the algorithm does, rewards people that are tightly intertwined. It should be considered that the best movie recommenders in a user's social network are not the most connected ones. It might instead be acquaintances or old friends from childhood that do not know any of your other friends. (It should also be said that the user's closest friend is not necessarily the one with the most connections in the network).

One possible way of revising the inferred trust values is to consider the actual trust values that the users assign their friends – either by replacing the inferred values or weighing the inferred values against them to devalue or increase the trust. The trust ratings could be normalized to account for any differences between the users' individual rating scales. The advantage of using inferred values is the possibility of including transitive relationships when computing recommendations. Using trust values in general saves the worry about sparse ratings. One could otherwise use a benchmark CF algorithm to compute similarity values for people in the social network. The active user's actual trust ratings could then be used to weigh the similarity value.

When users are asked to choose friends as movie recommenders, they will not necessarily choose the users that will generate the best recommendations. This depends on the active user's knowledge of his or her friends' movie tastes. A user *a*, that is interested in movies and has friends with the same interest might choose friends with similar taste, while another user *b* who the only one that watches movies in her circle of friends might not have any friends with similar movie taste. Both of these users will probably be interested in a movie recommender and will probably require good recommendations to keep using it. In order to address this issue, the recommender system could provide additional recommendations based on for instance similar users, where friends are not key, besides the friend based recommendations. This is for instance done in the Jinni application.

A problem that occurred several times during the main user test was that users could not get any recommendations because they had rated the same movies that their friends had. This is especially a problem for users who themselves have rated many movies but have friends who have not and this might be another reason why the system should provide additional types of recommendations. Furthermore, it could be worth noting that the user preference and number of movies is an issue in a small system; given a dataset of 400 movies and a user that only likes action movies, if 10% are action movies and the user rates all of these, there will be no movies left that can be recommended.

6.2.3 Social Networks

While the choice was made not to disclose people's ratings in the application, social networking sites are driven by transparency between actors; where wall posts, photographs and profile information are made available for others to see and interact with at the active user's discretion. The results held some indications that that the most intriguing aspects of an application like MovieBuddy could be viewing how friends rated movies and what recommendations that friends would generate. The majority of users also chose to add 4 to 6 friends which may suggest that they care for this process. Using friends to create recommendations brings the possibility of adding a new social level to the recommendation process that has not been considered in typical recommender systems. The popularity of these sites indicates that there is a demand for such interactions which could be incorporated in a recommender system by for instance letting the active user choose to whom his or her

ratings should be made visible and enabling direct recommendations, or tipping, between users (much like wall posts on Facebook).

Currently the recommender system does not build a profile for the active user outside of calculating his or her average rating and inferring trust values in the user's social network. There is however a lot of information available to third-party actors on social networking sites that could be used to build a user profile in a recommender system; such as favourite movies and TV-series. Also, if more than one social network is connected through the recommender system, for instance by adding MovieBuddy on several platforms, both knowledge of the users and the social network generated in the system could expand even further.

6.2.4 Accuracy

According to the results, the algorithm predicts ratings that are closer to the users' actual ratings than random "predicted" ratings are which means that the hypothesis has not been falsified by the results. Meanwhile, the mean absolute error of 0,93 points is quite large. As previously mentioned, research has indicated that there seems to be a limit at 0,73 (on a five-point rating scale) for how accurately an algorithm can predict ratings, and while the Ericsson algorithm generates a larger error, it is still in the same ballpark.

It has previously been noted that users tend to rate towards visible predicted ratings in a system, so if these were to be made visible the MAE could possibly be lowered. On the other hand, the fact that predictions are not shown to the user allows for some error. For instance; if the system predicted a rating 4,5 for an item that the user considers a 3,5 he or she might feel that the prediction was bad even if the user's average rating is 3 (which still suggests that he thought the item was better than average). When the predictions are not shown, the user will never know if the system predicted the item 3,5 or 5. The concern is instead if the recommendations are *good* or *bad*.

The survey indicated that almost everyone felt that their recommendations list had peaked their interest in at least one new movie and the gathered data showed that the users had rated the movies in the recommendation list higher than they did in the rest of the

application, which indicates that the recommended movies were better than average. Half of the users however stated that they were less than thrilled with their lists (gave it 3 or less out of 5) and the overall impression of the survey was that there were room for improvement with regards to the recommendations. So, even if the recommendations are successful in getting the users interested in new movies and the individual items are considered better than average, it does not mean that they will appreciate the list of recommendations as a whole.

6.2.5 Prototype, Content and Design

The application is fully functional and should be considered a useful tool with which different features and algorithms could be tested in future development. A risk in launching an application prototype that looks much like a fully functional product is however that the users might not be so lenient when it comes to possible functionality problems than arise during use. If they find the layout or general performance irritating for some reason, their experience with the entire recommender process can be tainted which can influence their perception of the recommendations negatively. In order to manage this, the initial user test was conducted.

It should be said that some errors was probably left undetected due to the lack of a proper user test, but the overly critical attitude of the participants in the initial user test helped detect some issues. Due to a shortage of time, the "invite friends" function was implemented after the main user test had begun which could have affected the amount of participants negatively. Several other functionalities were suggested in the initial user test but were chosen not be considered in this project, like a search function and more sorting options. One of the users also wrote that being able to see a friend's ratings would be the main reason why she would want to use the application.

Since the items with the highest predicted ratings are displayed in the application, the predicted ratings accuracy is quite important in order to ensure good recommendations. 20 movies were displayed in the application and the majority happened to have predicted ratings that were higher than the users' average rating. If the choice had been to restrict the number of displayed movies to 40 instead, the number of movies with predictions lower

than the users' average could have been higher. While the accuracy of the predictions might have stayed the same, the users' perceptions of the recommendations might not have. If items get predicted ratings lower than the active user's average, the questions arises whether or not the items can be considered recommendations at all.

The survey contained suggestions of incorporating a link to IMDb, which is a common feature in recommender systems. Currently, the information displayed on each movie is its title, an image and a plot description. This could be extended to leading roles, director and trailer links depending on how one wants to present the system. The plot information was uneven at times with regards to length – some movies had no information while others had quite a lot – which annoyed some of the users. These may be deciding factors on whether or not the user finds the system appealing, they should preferably be revised. Either in the system by manually changing the information in the database or by finding it somewhere else, or in the application, by hiding plot information after a certain length.

6.2.6 Marketing

Identifying individuals that appear in many social constellations can be very useful for marketing purposes. They should be able to reach large number of people with their opinions and ideas and for a company it could be valuable to target such individuals and have them as spokes persons for products or services, or spreading the word on the new movie recommendation application MovieBuddy. These are however not aspects which have been examined in this project, but they could be worth mentioning for future reference.

6.3 Further Development

While I believe that the idea that friends are innately good movies recommenders should be rejected, using friends as recommenders show promise. There seems to be an interest from users to have friends as movie recommenders and the first question to answer is what role in the recommender process friends should be allowed to play.

The survey indicated that the users have an interest in a more diverse recommender system than the MovieBuddy application supports. Some of the user tasks that the users are

interested in demand that the items themselves are categorized by tags like genre, filmmaker, actors and more; making versatile filtering possible. Some of the user tasks that they showed an interest in allows for the possibility of recommending items that the user has already seen; if recommending items to more than one person or for a certain type of movies of which the user has little knowledge. Mapping different algorithms to certain user types is made possible by the metrics found in Human Recommender Interaction theory.

6.3.1 Combining Trust

Since the trust inference process proved to give users on the same distance from the active user very similar trust values, one alternative that could be considered is combining the inferred trust values with the actual trust values. According to the results, the transitive relationships ranged basically between 0 and 6 (when transformed to a scale of 1-10) while most of the actual ratings were higher than that. The transitive relationships could be left as computed while the direct relationships are combined with the actual trust ratings to get a mean and thereby consider the difference in trust amongst friends more than it has so far.

The actual trust between friends in a social network could otherwise be used to create transitive relationships. If all trust values for a user is stored in a matrix (as currently done), the system can iterate over the active users social network to find all transitive friends and their actual trust values.

Example trust matrices

Active user	User	Trust
user a	user b	0,7
user a	user c	0,9
user a	user d	0,2
user a	user e	
user a	user f	

Active user	User	Trust
user b	user e	0,9
user b	user f	0,4

Figure 37) Example trust matrices

I would suggest computing a user's average trust rating and then only finding transitive friends through those friends who have the same trust value as the average or over. Much

like items with a predicted rating under the active user's average rating are not suited as recommendations, friends who have low trust values might not be reliable enough to recommend friends in the context. In the example graph below (where the average trust value is $\overline{T}_g = 5$) only two friends, user *j* and user *k*, then are considered to lead to transitive friends. The level to which the direct friend is trusted and the depth on which he or she is from the active user should also matter. The premise here is basically that the less a user is trusted in the context and the further away he or she is from the active user, the less that person's opinions should matter in the prediction process.

Social graph of friends

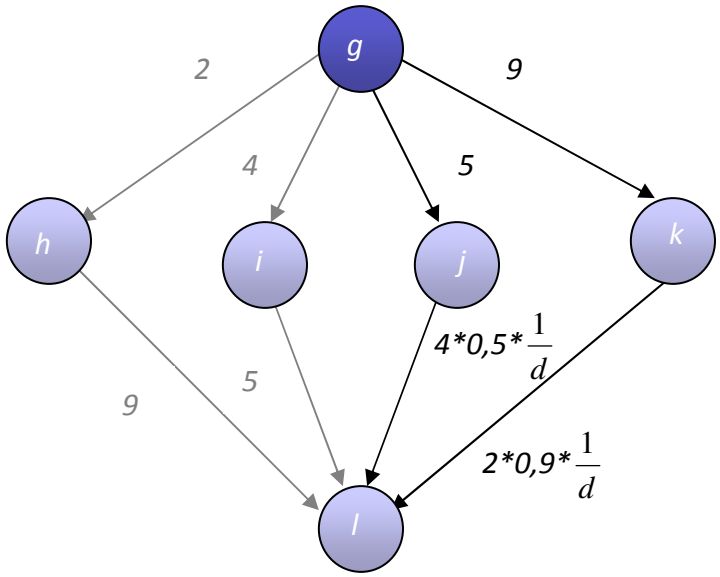


Figure 38) Social graph of friends

Pseudo code

```
for each friend in list
        if trust(a,b) ≥ average trust for user a
                get friends of this friend b
                trust = trust(a,b) * trust(b,e) * (1/d)
                if trust > 2
                        add to list(trust, current user)
```

The mean value of the two direct friends' opinions gives the transitive trust value (note that the trust ratings are on the scale 1-10).

6.3.2 Considering Uncertainties and Profiling Users

A reason why the depth is important to consider is that it comes with a level of uncertainty. For instance: the active user likes mostly action movies and comedies and gives a friend a high trust value based on their common interest in action movies. That friend then gives a high rating to another friend due to their interest in science fiction. If the system only considered the users actual trust the transitive relationship would be calculated as quite strong – not at all considering that the active user does not like science fiction at all.

The system does not consider any profile information about a user, like what type of movies the user rates the highest or lowest. If users could be compared on their profiles then transitive relationships could be discarded when the users do not match each other; this would decrease the uncertainty with depth and render a high devaluing factor like $\frac{1}{d}$ excessive. In order to avoid perfect trust in transitive relationships, a devaluing factor of 0,9 could be used instead in every step. This way, transitive relationships could become stronger than direct relationships which could result in a social network of friends optimized towards generating good movie recommendations.

Using friends as recommenders will not be an exact science. Replacing the inferred trust values with the actual trust ratings will generate recommendations that depend more or less entirely on how good or bad taste the user's friends have, with

$$P_{u,i} = \overline{R}_u + \frac{\sum_{v \in U_i} T_{(u,v)} (R_{v,i} - \overline{R}_v)}{\sum_{v \in U_i} \left| T_{(u,v)} \right|} \tag{6.1}$$

In order to account for differences between individual rating scales, the actual trust ratings should be normalized. I believe that the users will understand that the recommendations are an effect of this and the trust values they have given their friends. To enhance the social

aspects further, the users' ratings could be made visible to friends at the user's discretion. This could also help users get a better idea of their friends' movie tastes. Some recommender systems for movies choose to display ratings from IMDb as well as the systems predicted ratings. Based on the knowledge that users tend to rate towards what the system has rated, one has to weigh the value for a user to see how a movie has been rated towards minimizing the influence on the users. I would suggest keeping the predicted ratings invisible but a movie's rating on IMDb could be displayed (for instance in numbers as not to conflict with the users star ratings). This would give the users some additional appreciation of the film. Another social feature could be allowing direct recommendations to be made between friends.

Some of the transitive relationships in the system had a trust value lower than 0,1 and since low similarity values have shown to decrease the quality of recommendations in benchmark CF systems, a minimum weight should be added which all correlations values must be over – independent of how trust is computed.

Adjusting other aspects of the algorithm might improve the predict ratings. Instead of using the inferred trust ratings as calculated by (4.5), similarity values could be computed for all users that have the application in addition to the explicit trust ratings that the active user sets. An average value could for instance be computed between the actual trust and the computed similarity like in (8.1) but with *corr(u,v)* as computed by (4.1). This would adjust the end value towards the user's opinion. Based on the literature study, whenever a similarity value is used in a recommendation process it could be weighed with a factor to devalue the ones that have been computed using few co-rated items. For instance a factor of $n/20$ where n is the number of co-rated items. This number (1/20) could increase in a larger system. An issue to consider is that while a high number of co-rated items increase the security in the similarity measure, it becomes more likely that there are few items left that can be recommended by that user. If similarity values were computed for the friends, the system could display to the active user a list of suggestions of friends that the system thinks would be good as recommenders. This could strengthen the quality of the recommendations and help the users who do not have a good perception of their friends movie taste.

7 Future Work

This thesis project has examined the possibilities and limitations surrounding integrating a social network in a recommender system. The focus has been on creating a tool with which the system could be tested on real users and evaluating its success by measuring prediction accuracy and gathering the users' views on the recommendations. This chapter contains suggestions on fields of study for future work.

7.1 Human Recommender Interaction & User Tests

This project has included a brief summary of Human Recommender Interaction theory which provides insight into the users' perspectives on recommender systems much like Human-Computer Interaction does for computer design and evaluation. HRI also covers interesting mappings between the characteristics of recommendation lists and recommender algorithms. It is a field of study that could be useful to investigate further.

As previously described, the movies in the dataset were chosen as they were well recognizable. Close to 90% of all movies in the movie list were from 1990 or younger and there were as many movies from this year (2009) as there were movies predating 1990. I would suggest that further testing contains a larger data set with more diverse movies. This could show how good friends are as recommenders with a more obscure set of movies. I would also suggest further tests on the accuracy of the predictions, where items with the lowest predicted ratings were shown as recommendations. Since the ones with the highest predicted ratings were shown, the accuracy has not been tested on as many movies with low predictions.

7.2 User Profiling & Combining Platforms

The recommender system currently does not consider user preferences with regards to specific genres, actors, directors or such when computing recommendations. Such aspects of a user profile could be used to personalize recommendations further. As previously stated, some of the information kept on social networking sites such as Facebook could be used to build these profiles. The Jinni system has chosen to create movie preference tests with the

purpose of pin pointing the users' preferences.

Integrating the recommender system on other social networking sites could generate a larger user base and give an indication on the user interest for movie recommendations (or recommendations in other domains) on different types of platforms. The flexibility of OpenSocial can also make it interesting to consider for future work. Another research area which might be of interest is mining both social networks and user profiles from the internet using semantic web technologies. The combination of trust inference algorithms and such networks seem to be becoming more popular in recommender research.

8 References

ALEXA (2009a). *Google.com.* (Electronic). Available: <http://www.alexa.com/siteinfo/google.com> [2009-08-13]

ALEXA (2009b). *Flixter.com.* (Electronic). Available: <http://www.alexa.com/siteinfo/flixster.com> [2009-08-25]

ALEXA (2009c). *Orkut.com.* (Electronic). Available: <http://www.alexa.com/siteinfo/orkut.com> [2009-08-13]

ALEXA (2009d). *Facebook.com.* (Electronic). Available: <http://www.alexa.com/siteinfo/facebook.com> [2009-08-13]

APPDATA (2009). *Flixter (Movies).* (Electronic). Available: <http://www.appdata.com/> [2009-06-29]

BELL, J. (2005). *Introduktion till forskningmetodik*. 4ed. Lund: Student litteratur.

BJÖRK, J. (2008). *Personalized TV and Content Recommender – collaborative filtering in recommender systems*. Master of Science Thesis. LTH Engineering Physics Department of Mathematics. Luleå, Sweden

CODEIGNITER (2009). *Application Flow Chart.* (Electronic). Available: <http://codeigniter.com/user_guide/overview/appflow.html> [2009-11-15]

ERICSSON (2008). *Ericsson Annual Report 2008*. (Electronic). Available: <http://www.ericsson.com/ericsson/investors/financial_reports/2008/annual08/sites/default/files/downloads/Complete_Annual_Report_2008_EN.pdf > [2009-09-17]

FACEBOOK (2009a). *Facebook Developer Wiki.* (Electronic). Available: <http://wiki.developers.facebook.com/index.php/Main_Page> [2009-08-31]

FACEBOOK (2009b). *FBJS*. (Electronic). Available: <http://www.wiki.developers.facebook.com/index.php/FBJS> [2009-09-15]

FACEBOOK (2009c). *Facebook Statistics.* (Electronic). Available: <http://www.facebook.com/press/info.php?statistics> [2009-09-16]

FACEBOOK (2009d). *User profile info.* (Electronic). Available: <http://wiki.developers.facebook.com/index.php/Users.getInfo> [2009-09-21]

GOLBECK, J. (2005).*Computing and Applying Trust in Web-based Social Networks*. Ph. D.

Dissertation at the University of Maryland, USA.

GOLBECK, J. (2006). *FilmTrust: Movie Recommendations from Semantic Web-based Social Networks*. 3rd IEEE Consumer Communications and Networking Conference, Vol. 2, p. 1314 – 1315.

GOLBECK, J. (2008). *Weaving a Web of Trust*. Science, Vol. 321, No. 5896, p. 1640-1641, September 2008.

GOLDBERG, D., NICHOLS, D., OKI, B. & TERRY, D. (1992). *Using Collaborative Filtering to weave an Information Tapestry*. Communications of the ACM, Vol. 35, No. 12, p. 61-70, December 1992.

HANSSON, S. O. (2003). *Konsten att vara vetenskaplig*. Kursbunt examensarbete i Medieteknik. Stockholm: Filosofienheten, KTH.

JINNI (2009a). *Press and Awards*. (Electronic). Available: <http://www.jinni.com/press.html> [2009-09-16]

JINNI (2009b). *Watch our Demo*. (Electronic). Available: <http://www.jinni.com/signin.html> [2009-09-16]

JINNI (2009c). *The Movie Genome*. (Electronic). Available: <http://www.jinni.com/movie-genome.html> [2009-09-16]

KONSTAN, J. et al. (1997). *Applying Collaborative Filtering to Usenet News*. Communications of the ACM, Vol. 40, No. 3, p. 77-87, March 1997.

LIU, N. & YANG, Q. (2008). *EigenRank: A Ranking-Oriented Approach to Collaborative Filtering*. Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval, p. 83-90, Singapore.

MARSH & BRIGGS (2009). *Capturing Trust in Social Web Applications*. Computing with Social Trust. P. X-Y. Golbeck, J. (ed.). CITY:FÖRLAG

MCNEE, S. (2006). *Meeting User Information Needs in Recommender Systems*. Dissertation at the University of Minnesota, USA.

MCNEE, S., RIEDL, J. & KONSTAN, J. (2006) *Being Accurate is not enough: How accuracy metrics have hurt recommender systems*. Conference on Human Factors in Computing Systems, CHI '06 extended abstracts on Human factors in computing systems, p. 1097-1101, April 2006, Montréal, Québec, Canada.

MOVIELENS (2009). *About MovieLens*. (Electronic resource). Available:

<http://www.movielens.org/info?action=about> [2009-06-29]

O'DONOVAN, J. et al. (2008). *PeerChooser: Visual Interactive Recommendation.* CHI '08: Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, p. 1085 – 1088.

O'DONOVAN, J. (2009). *Capturing Trust in Social Web Applications.* Computing with Social Trust, p. X-Y. Golbeck, J. (ed.). CITY:FÖRLAG

OPENSOCIAL (2009). *OpenSocial Foundation FAQ.* (Electronic). <http://www.opensocial.org/page/opensocial-foundation-faq> [2009-08-31]

PREECE, J., SHARP, H. & ROGERS, Y. (2007). *Interaction Design – beyond human-computer interaction.* 2nd Ed. West Sussex: John Wiley & Sons, Inc.

RESNICK, P. & VARIAN, H. (1997). *Recommender systems.* Communications of the ACM, Vol. 40, No. 3, p. 56-58, March 1997.

SHAHAN, M. (2008). *Personalized TV with Recommendations – Integrating Social Networks.* Master of Science Thesis. KTH Information and Communication Technology. Stockholm, Sweden.

SINHA, R. & SWEARINGEN, K. (2002). *The Role of Transparency in Recommender Systems.* Conference on Human Factors in Computing Systems, CHI '02 extended abstracts on Human factors in computing systems, p. 803-831, Minneapolis, Minnesota, USA.

WASSERMAN, S. & FAUST, K. (1994). *Social Network Analysis – Methods and Applications.* New York: Cambridge University Press.

ZIEGLER, C. & LAUSEN, G. (2004). *Spreading Activation Models for Trust Propagation.* IEEE International Conference on e-Technology, e-Commerce and e-Service, p. 83-97, March 2004.

In addition, internal material from Ericsson AB Research has been used.

APPENDIX A: COMMENTS FROM THE FIRST USER TEST (IN SWEDISH)

User A

- *Det är konstigt att filmer som jag redan röstat på eller klickat "not interested" dyker upp igen. Gör processen lång. Dessutom konstigt att de dyker upp på recommendations, speciellt när jag aktivt sagt att jag är "not interested" av dem.*

- *O så vore det kul att kunna bjuda in fler vänner till appen, inte bara de tre som fanns tillgänliga när jag började.*

User B

- *Jag skulle vilja feedback på hur många filmer jag har bedömt.*

- *Istället för save friends borde det stå choose friends.*

- *Det skulle vara användbart om man kan sortera/söka upp filmer under quickrate, genom ett sökfält med text eller likande så man kan betygsätta inte endast slumpvis valda filmer utan även söka upp filmer själv om man precis sett en bra film man vill rekommendera till sina vänner exempelvis.*

- *Går det på något sätt att lägga in en egen film?*

- *Går det att se andra vänner rekommendationer? Det är dessa som är intressantast att se enligt mig i alla fall. Det jag skulle se som huvudsyftet med tjänsten.*

User C

- *När man är inne på My Recommendations och redan har ratat en film, står det fortfarande "Not seen It" i select listan.*

User D

- *I QRate kommer samma film upp flera gånger (inte i rad, men ändå). Osäker på om den ska det.*

APPENDIX B: ABOUT MOVIEBUDDY

Welcome to MovieBuddy!

The MovieBuddy application is a part of a thesis work at the Royal Institute of Technology (KTH) at Stockholm, Sweden. Its purpose is to create movie recommendations for you, the user, based on your and your friends movie taste. Below you will find instructions on how to use MovieBuddy, as part of its test run. As it is still in development, we hope that you keep your calm and patience if functionality issues arise during use. You are more than welcome to invite other friends to join you at MovieBuddy, we do not provide a quick link to do this yet.

Instructions for user test

1. **Start by rating some movies under QuickRate**, it displays random movies from our movie database. You can always see the movies you have rated so far under Rated movies.

2. **Go to Select friends and choose which of your friends you want as movie recommenders.** This means that their taste will influence which movies are recommended to you. All of your friends who also have this application are visable and you can choose as many friends as you'd like. This information will not be visible to any of your friends.

3. **Next you will be asked to Rate friends.** Here you can rate how much you want your chosen friends to influence your recommendations on a scale of 1-10. A 10 indicates that you believe that he or she has a briliant taste in movies, and a 1 means that you have quite the opposite taste. Again, these ratings are keept anonymous throughout the application.

4. **Now it is time to rate your movie recommendations. These last two steps are the most important part of our evaluation.** Go to My recommendations. For all the movies (up to 20 movies) you are to rate the ones you have seen, *'You want to see'* or is *'Not interested'* in. If you haven't seen the movie or don't know what it is, just continue browsning the list until your reach the final movie.

5. **When you are done, follow THIS LINK to complete a survey about MovieBuddy.** Then we're done! Thank you for your time and if you want you're welcome to keep using the application.

APPENDIX C: ANSWERS FROM THE SURVEY(IN SWEDISH OR ENGLISH)

1. I am a:

	Answers	Percent (%)
Woman	4	40%
Men	6	60%
Total	10	

2. My age is:

	Answers	Percent (%)
0-15	0	0
16-24	3	30%
25-30	7	70%
31-	0	0
Total	10	

3. How familiar were you with the movies shown in the MovieBuddy application?

	Answers	Percent (%)
1 (Had not heard of any of them)	0	0
2	0	0
3	2	20%
4	7	70%
5 (Had heard of all of them)	1	10%
Total	10	

4. Had you seen any of the movies in your recommendation list?

	Answers	Percent (%)
1 (None)	0	0
2	1	10%
3	3	30%
4	6	60%
5 (All of them)	0	0
Total	10	

5. Did you like the list of recommended movies you were given?

	Answers	Percent (%)
1 (No, not at all)	0	0
2	2	20%
3	4	40%
4	3	30%
5 (Yes, really appreciated it)	1	10%
Total	10	

6. Did the recommendations peak your interest in any new movies?

	Answers	Percent (%)
No	1	10%
Yes	9	90%
Total	10	

7. What are you looking for in a personalized movie recommender?

	Answers	Percent (%)
Get inspiration on what to watch	10	100%
Keep updated	9	90%
Find movies of a certain type	6	60%
Find a movie for my current mood	6	60%
Find movies for more than one (e.g. for a movie night)	5	50%
Rate movies	2	20%
Write reviews	2	20%
Not interested in personalized movie recommendations	0	0
Total	40	

8. What was your overall perception of the recommendation list?

- *Found some movies that I´ve seen, but forgotten about which is always nice*
  Most movies were movies I've already seen, some were movies I don't want to see, and some were movies I'd like to see.
- *cool application!*

- *i need this thing.*

- *It was very good. I have an account at Lovefilm.se and their recommendations list is very flawed,*
- *especially compared to this one. And here I had only graded around 20 movies compared to over 200 at Lovefilm, so either the algorithm is really good, or I have friends with really good taste in movies. :)*

- *I would like to sort and search for movies.*
- *För stor. Det hade räckt med typ 10 filmer. Dessutom var det förvirrrande att man skulle betygsätta filmerna även där.*

- *Bra men lite slö att ladda*

9. How NOVEL would you say that the recommendations were?

	Answers	Percent (%)
1 (Boring)	0	0
2	2	20%
3	5	50%
4	3	30%
5 (Novel)	0	0
Total	10	

10. How BOLD would you say that the recommendations were?

	Answers	Percent (%)
1 (Safe)	0	0
2	1	10%
3	7	70%
4	2	20%
5 (Bold)	0	0
Total	10	

11. What was your overall perception of the MovieBuddy application?

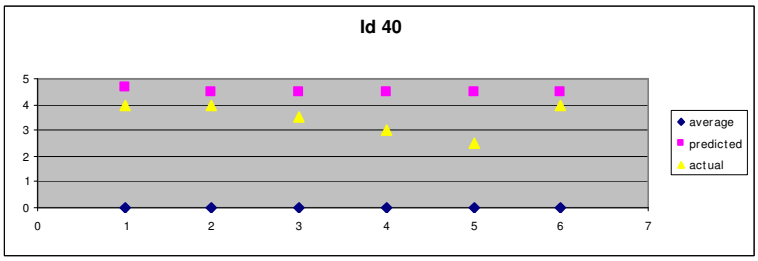
- *Good idea, especially is you know the taste of your friends well*
- *Pretty nice, with some work and some more functions it will be a really good app!*
- *liked it. was kinda surprised of the movies i got recommended. i'm not into action movies usually... thnx.*

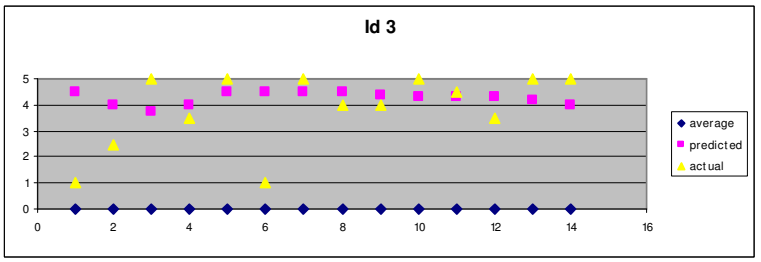
- *It was pretty good. It was a bit slow at times, which was a bit annoying.*

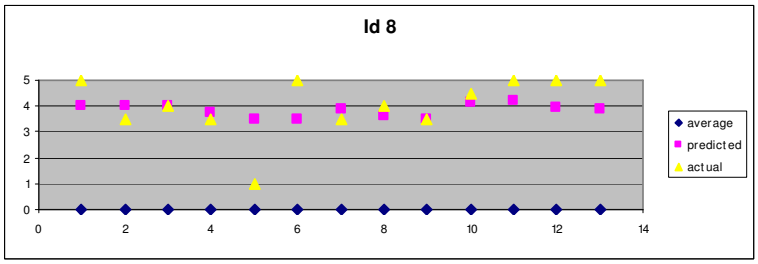
- *The "I want to see this movie" and "I'm not interested" were a bit big compared to the rating drop-down menu.*

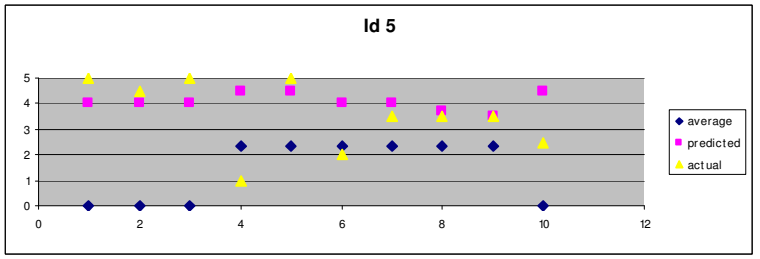
- *The amount of information on each movie was very varying from absolutely nothing (Ip Man) to quite long texts. I would also like more info like a cast list and the length of the movie and so on, which would be good for movies you haven't already seen. Or perhaps simply a link to imdb. The recommendations list was very impressive however, making the app well worth using. But as expected in the beginning of rating movies, you mostly get movies you have already seen and liked, but hopefully this will change as you rate more and more of the movies already seen.*
- *I think it would be more useful if I had more friends to rate.*
- *good*
- *Bra idé väl genomförd. Den kan nog spetsas till ännu mer och få ännu skarpare filmrekommendationer.*

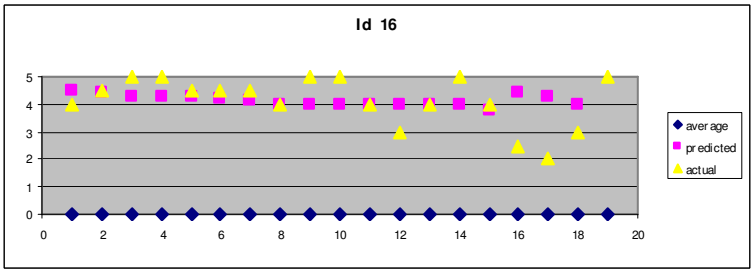
APPENDIX D: DIFFERENT USERS' AVERAGE RATINGS COMPARED TO PREDICTED AND ACTUAL RATINGS

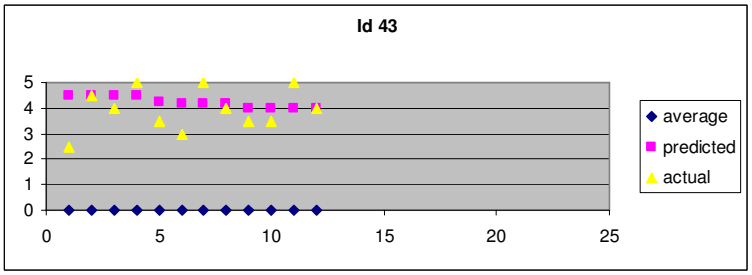


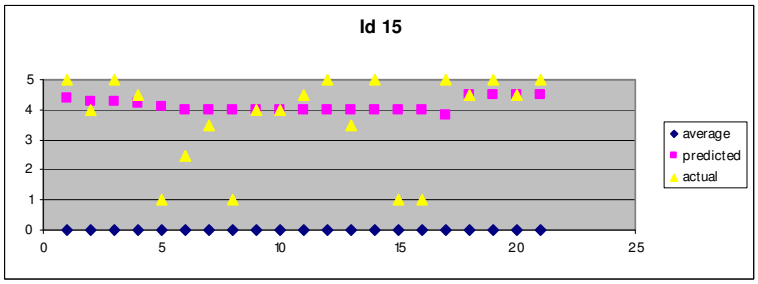


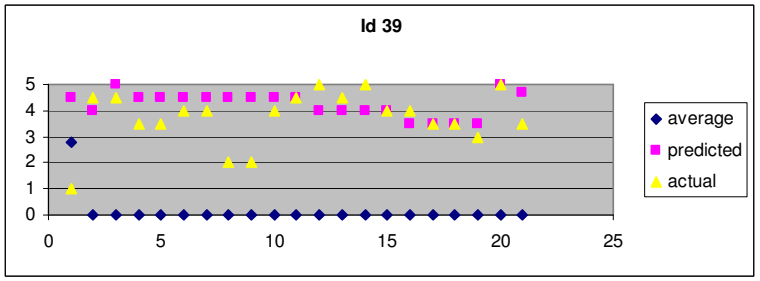


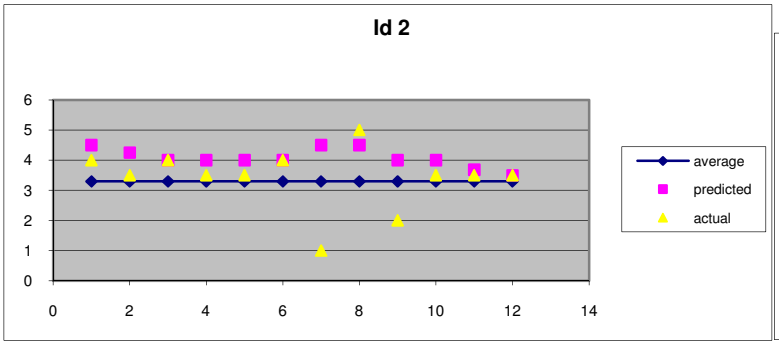


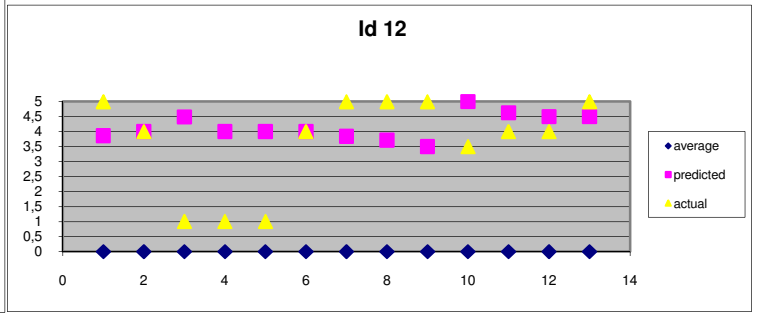


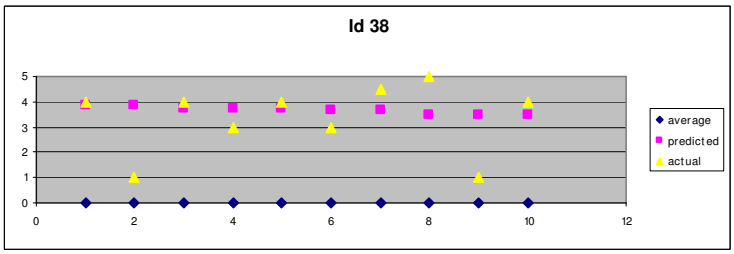


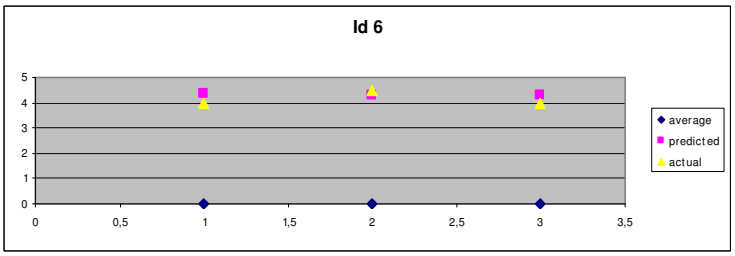


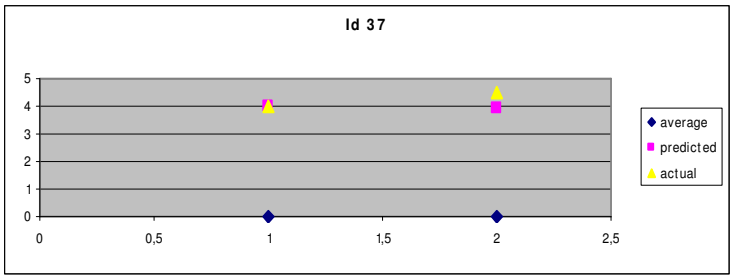


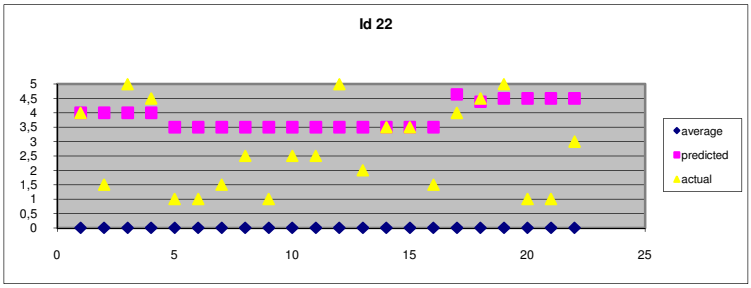





Id 38
5
4
3
2
1
0
0
2
4
6
8
10
12
average
predicted
actual


Id 6
5
4
3
2
1
0
0
0,5
1
1,5
2
2,5
3
3,5
average
predicted
actual


Id 37
5
4
3
2
1
0
0
0,5
1
1,5
2
2,5
average
predicted
actual


Id 22
5
4,5
4
3,5
3
2,5
2
1,5
1
0,5
0
0
5
10
15
20
25
average
predicted
actual

TRITA-CSC-E 2010:123
ISRN-KTH/CSC/E--10/123--SE
ISSN-1653-5715

www.kth.se