University of Zurich[UZH]

Computation and Economics Research Group

Thesis August 10, 2014

Social Recommender Systems

Matthias Felix
of Zurich, Switzerland

Student-ID: 11-746-625
matthias.felix@uzh.ch

Advisor:

Prof. Sven Seuken, PhD
Institut für Informatik
Universität Zürich
http://www.ifi.uzh.ch/ce

Acknowledgements

I would like to thank Mike Shann for his great advice and for his cooperativeness. I would like to thank my wife Céline for her patience and for showing me so much understanding. And most importantly, I want to thank God, because without Him I would not have been able to write this thesis during a very stressful time.

Abstract

Online recommender systems have become an important help for users of various applications. They help users deal with the information overload often encountered in today's online world and filter out relevant alternatives. Social recommender systems moreover try to use information contained in social networks to enhance the performance of conventional recommendation methods.

This thesis focuses on the collaborative filtering recommendation approach and analyzes its performance in different environments. A close look is taken at the incorporation of social network information into the collaborative filtering algorithm. Further, concepts from network theory are used to analyze if there exist users who influence the taste of others and therefore could be better predictors.

Zusammenfassung

Online-Empfehlungsdienste sind in der heutigen Zeit zu wichtigen Hilfen für Benutzer verschiedenster Applikationen geworden. Sie helfen Benutzern, mit der Informationsflut zurecht zu kommen und relevante Alternativen heraus zu filtern. Soziale Empfehlungsdienste versuchen, mit Hilfe von Informationen aus sozialen Netzwerken die Empfehlungen herkömmlicher Systeme zu verbessern.

Diese Arbeit konzentriert sich auf die Collaborative-Filtering-Methode und untersucht deren Performance unter verschiedenen Bedingungen. Weiter wird die Einbindung von Information aus sozialen Netzwerken in den Collaborative-Filtering-Algorithmus genauer studiert. Es werden ausserdem Konzepte aus der Netzwerk-Theorie verwendet, um herauszufinden, ob es Benutzer gibt, welche den Geschmack anderer beeinflussen und somit bessere Indikatoren für Empfehlungen sein könnten.

Table of Contents

1

Introduction

Recommender systems have become a widely used tool for users of different kinds of online services. They can help users deal with the information overload and find content relevant for them more easily and reliably. There is a variety of different approaches to recommender systems, of which two main categories can be established: content-based filtering and collaborative filtering. The former recommends items with similar content to items a user has liked in the past, while the latter tries to recommend items to a user that other users with similar taste have liked before.

At the same time, online social networks are becoming more and more important in our society. There are endless possibilities to connect with new people, share content that you like or discover new content that others propose to you. These facts have led to the idea of using the information contained in social networks in order to enhance the performance of recommendation algorithms. Knowing that friends often have similar tastes, the information contained in a social graph can be used to find people who could be better indicators of a user's preferences. The combination of recommendation algorithms and social network information gave rise to a new class of algorithms called *social collaborative filtering*. There have been numerous studies on this topic, some of which report better performance with social collaborative filtering algorithms than with conventional ones (see [Zheng et al., 2008], [Konstas et al., 2009], [Liu and Lee, 2010]).

In this thesis, the focus lies on systematically inspecting and analyzing conventional and social collaborative filtering. Different social collaboravite filtering algorithms will be compared to conventional ones and in addition, graph and network analysis techniques will be used in order to find conditions under which social collaborative filtering might lead to better performance or to find users with special positions in a social network that could make them better predictors.

The main research questions that are asked and analyzed in this thesis are:

- Under which conditions does social collaborative filtering perform better than conventional collaborative filtering?
- Are there users for whom social collaborative filtering works better than conventional collaborative filtering?
- Are there users that are better predictors than others?

To answer these questions, computer experiments and simulations are performed on real-world and artificial datasets, with a software simulation environment built specifically for this purpose. A publicly available dataset from the social music streaming service last.fm is used, while artificial social networks are created using a well-suited network generation algorithm. Artificial rating data is generated as well using a realistic model.

Gaining a deeper understanding of the functioning of social recommender systems and the conditions under which they perform best will help to create recommender systems that are more suitable for their specific application areas and that make use of social network information in the best way possible. It will help people to make the right choices when designing new recommender systems and it will open up new possibilities to take into account information about users found in social networks.

2

Recommender Systems and Social Networks

This chapter is dedicated to the introduction of the relevant theoretical concepts that need to be understood in order to properly comprehend social recommender systems. It is arranged in three sections: The first section gives an overview over recommender systems' history, application areas and theoretical concepts. It also defines a formal framework for recommender algorithms and presents the different algorithmic approaches used in the experiments (chapter 4). The second section is devoted to social networks and their properties. Understanding social networks and their characteristics is important for the incorporation of information into recommender systems and also for the generation of artificial networks (chapter 3). The third section then combines the two fields by introducing recommender algorithms that incorporate social network information.

2.1 Recommender Systems

People face the challenge of making decisions and weighing different options every day. Often they take into consideration not only their own opinion and knowledge, but also the advice and recommendations of others. But these methods have their limits. A person might for example never hear about a good book that he would really like, if none of his friends know that book. And often the almost infinite number of alternatives (e.g. when deciding what movie to watch or what music to listen to) makes it impossible to properly evaluate all options and find the best one. This is where a computer-based system can outperform human recommendations. Recommender systems are tools that try to predict the valuation users have for different items. They aim at helping users in their decision-making processes, like what items to buy, what movies to watch or what music to listen to. Advantages over human recommendations include the much larger set of recommenders (even people who do not know a user personally might contribute some valuable information), the more systematical inclusion of a user's history and his already expressed preferences and the possibility to computationally deal with a much larger amount of alternatives. The possible application space of recommender systems is endless and there exist real-world examples in many different areas like books, restaurants, financial services [Felfernig et al., 2007] or even persons (online dating).

There are a lot of factors that determine the goodness and the practical success of recommender systems. Different application areas demand different abilities. In some areas, speed might be the crucial factor, in others transparency might be critical (users want to know how the recommendations are made), or it might be the case that users want a very broad range of new items recommended instead of only very similar ones. In reality, recommender systems are often used in commercial settings and the design of a particular recommender system should be highly dependent on the environment it will be used in. It is important to carefully evaluate the requirements posed by that environment and build the recommender system according to those requirements.

As a consequence of the various conditions and surroundings recommender systems operate in, there exist many different approaches and recommendation techniques. [Ricci et al., 2011] give a good overview over existing methods and algorithms. Here, two of the most popular approaches are described, while the focus of the experiments in this thesis lies explicitly on the collaborative filtering approach.

2.1.1 Content-Based Filtering

Content-based filtering looks at the similarity between different items. Based on the attributes and characteristics of items a user has rated in the past, the system will recommend new items with similar characteristics. If someone has watched a lot of action movies with Sylvester Stallone (and given them a good rating), it is likely that movies starring Sylvester Stallone or movies belonging to the action genre will be recommended to this person. Whereas in movies one often has to make appropriate categorizations manually, this process can be automated with text-based content (like books or webpages). Text can be scanned for frequently used keywords and recommendations can be made based on keyword frequency ([Parkes and Seuken, 2013]).

Challenges

There are numerous challenges to this approach. One that was already mentioned before is the categorization of items. If content is not text-based, it is hard to automatically extract information about the content, especially in movie or music domains. Another problem is the new-user problem: It is hard to provide accurate recommendations for new users who have only rated very few items so far, since there is not much information about their preferences. Furthermore, there exists also the danger of over-specialization, which means that the user from the Sylvester Stallone example might never be recommended a comedy movie if he has not watched one yet, even though he might actually like it ([Parkes and Seuken, 2013]).

2.1.2 Collaborative Filtering

The method of collaborative filtering (CF) does not use any information about items' characteristics in order to make predictions. Instead it takes into account the rating

history of all users. This new approach already addresses one of the big challenges of content-based filtering, that is the high cost to categorize items. The new-user problem however still exists.

In user-based CF, the idea is to find users with similar rating behaviour and predict a rating for an item based on these similar users' ratings for this item. Item-based CF tries to find items that have been rated similarly across all users and then makes a prediction for an item based on the user's ratings for similar items ([Parkes and Seuken, 2013]).

Challenges

As already mentioned, some of the main challenges of content-based filtering can be avoided with this approach, however there occur other, new challenges. One difficulty of CF methods can be the extremely high computational cost. Calculating all user-similarities can be a very time-consuming task, especially in domains with thousands of users and possibly millions of items. Another problem is the new-user or new-item problem. When a new user or item enters the system, there will be very few or no ratings at all in the beginning, making it very hard to predict any rating similarities for users or items ([Parkes and Seuken, 2013]).

2.1.3 Formal Framework

To mathematically describe the different algorithms and metrics that are used in the experiments, a formal framework is introduced here. Let $U = \{1, ..., u\}$ denote the set of users and let $G = \{1, ..., g\}$ denote the set of items. Let $r_i(j)$ denote the actual rating of user $i \in U$ for item $j \in G$. The ultimate task for every recommender system is to find a good estimate $\hat{r}_i(j)$ for an item j that user i has not rated yet. The closer $\hat{r}_i(j)$ is to the user's actual rating $r_i(j)$ (which exists even if the user has not explicitly rated the item yet; it should be seen as the actual valuation of the user for that item), the more accurate the recommendation is.

As mentioned in the introduction (chapter 1), there are many aspects that make up the quality of a recommender system and the performance might be highly dependent on the environment the system operates in. Nevertheless, accuracy is considered to be the most important metric to measure the performance of a recommender system ([Ricci et al., 2011]). One common measure to express the accuracy is the *root-mean-square error* (RMSE) ([Ricci et al., 2011]). For a series of n predictions $\hat{r}_n$ and actual ratings r_n, it is defined as follows:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(r_n - \hat{r}_n)^2} \tag{2.1}$$

Similarity Measures

User-based CF tries to find users that are similar to a particular user i. The assumption is that similar users will be good predictors for the estimation of $\hat{r}_i(j)$. The first step is

thus to find a measure of similarity between two users. There exist different measures (see [Amatriain et al., 2011] for a list), of which the two most commonly used ones will be presented.

Pearson Correlation Coefficient The Pearson correlation coefficient measures the similarity between two users i and i'. Let $\bar{r}_i$ denote the average rating over all items user i has rated so far. $G_{ii'}$ ist the set of items both users have already rated. Then the Pearson correlation coefficient is calculated as follows:

$$sim_P(i,i') = \frac{\sum_{j \in G_{ii'}} (r_i(j) - \bar{r}_i)(r_{i'}(j) - \bar{r}_{i'})}{\sqrt{\sum_{j \in G_{ii'}} (r_i(j) - \bar{r}_i)^2} \sqrt{\sum_{j \in G_{ii'}} (r_{i'}(j) - \bar{r}_{i'})^2}} \tag{2.2}$$

The Pearson correlation coefficient results in a similarity value between -1 and $+1$.

Cosine Similarity Coefficient A second similarity measure that is widely used to calculate how similar two users are is the cosine similarity. The cosine similarity calculates the similarity of the orientation of two vectors. The two vectors in this case are the rating vectors of users i and i' respectively, containing all ratings $r_i(j)$ and $r_{i'}(j)$ respectively of the items in set $G_{ii'}$. The cosine similarity is then calculated as follows:

$$sim_C(i,i') = \frac{\sum_{j \in G_{ii'}} r_i(j) \cdot r_{i'}(j)}{\sqrt{\sum_{j \in G_{ii'}} r_i^2(j)} \sqrt{\sum_{j \in G_{ii'}} r_{i'}^2(j)}} \tag{2.3}$$

The similarities calculated with equation (2.3) also lie in the range from -1 to $+1$.

Once all similarities $sim(i,i')$ for a user i are calculated, the other users can be sorted by their similarity to user i. It makes sense to only consider the most similar users for the prediction of $\hat{r}_i(j)$, since their ratings correlate the most with user i's ratings. The next step is thus to choose an appropriate subset of all users i', which is called the *neighbourhood* N of a user. The size of this neighbourhood can be determined in different ways; it can either be a fixed number (e.g. 50) or it can be determined through applying a threshold on the similarities (e.g. only include users with similarity greater than 0.5).

Prediction Measures

Once this neighbourhood set is determined, there exist different ways to compute the estimated rating $\hat{r}_i(j)$. We consider two approaches and a third one that combines the first two ([Parkes and Seuken, 2013]).

Weighted Sum The weighted sum aggregation takes into account that more similar users should have a bigger influence on the prediction than less similar ones. To predict rating $\hat{r}_i(j)$, the rating for item j of each user i' in the determined neighbourhood N is weighed with $sim(i,i')$:

$$\hat{r}_i(j) = \frac{\sum_{i' \in N} sim(i, i') \cdot r_{i'}(j)}{\sum_{i' \in N} sim(i, i')} \tag{2.4}$$

Adjusted Sum One aspect that the weighted sum aggregation does not take into account is that users might interpret the rating scale differently. Some users might generally give lower ratings than others. To account for this, the adjusted sum prediction includes the average rating $\bar{r}_{i'}$ of each user i' in the calculation.

$$\hat{r}_i(j) = \bar{r}_i + \frac{\sum_{i' \in N} (r_{i'}(j) - \bar{r}_{i'})}{\mid N \mid} \tag{2.5}$$

Adjusted Weighted Sum The adjusted weighted sum combines both aspects; it weighs the users' ratings according to their similarity and it adjusts the ratings with the average ratings:

$$\hat{r}_i(j) = \bar{r}_i + \frac{\sum_{i' \in N} sim(i, i') \cdot (r_{i'}(j) - \bar{r}_{i'})}{\sum_{i' \in N} sim(i, i')} \tag{2.6}$$

Note that in all three approaches the sum is over all users in neighbourhood N. One has to take into account the fact that it is not certain that all users in N have already rated item j. In fact, the sparser the rating matrix, the higher the probability that there are users who have not rated item j and are therefore left out of the prediction calculation. The actual set of neighbours that influence $\hat{r}_i(j)$ can thus be much smaller than $|N|$.

2.1.4 Related work

There is a vast variety of studies and papers that give a survey on the topic of recommender systems. [Ricci et al., 2011] give a good overview over existing approaches to recommender systems and explain in detail different properties of recommender algorithms. [Adomavicius and Tuzhilin, 2005] describe the state-of-the-art of recommender algorithms and give theoretical background on content-based, collaborative filtering and hybrid approaches. They further suggest possible extensions to enhance these algorithms.

2.2 Social Networks

Social networks are structures that have existed for a very long time. It is even safe to say that people would not be able to live only by themselves, without any social connections. There exist many kinds of social ties between people, including friendships, family or also work-related relationships. A social network is basically made up of two

components; the social actors (which often are people, but could also be organizations or other social entities) and the ties between those actors.

Whereas in the past, social network structures were often implicitly present, but hard to explicitly capture and analyze, the rise of the internet and the effects of globalization are responsible for the appearance of online social networks like Facebook or Twitter. These networks often are a good reflection of real social networks, with the big advantage that they can be empirically studied and analyzed, as well as included in computer-based applications like recommender systems.

2.2.1 Formal Framework

Social networks can be mathematically represented by graphs. A graph consists of a set of nodes $v \in N$ and a set of edges $e \in E$ which are dyadic links between two nodes. Graphs are often drawn with nodes as little circles and edges as connecting lines between the circles. The edges can either be undirected, meaning that the relationship is bi-directional, or they can be directed (drawn as arrows instead of lines) meaning that the relation goes only in one direction. Figure 2.1 shows an example of both a directed and an undirected graph.

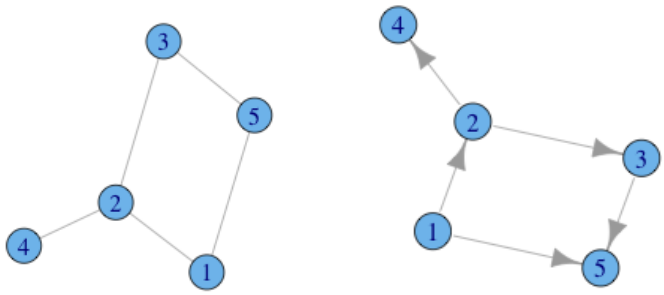

Figure 2.1: An undirected and a directed graph, each with 5 nodes and 5 edges.

In the space of online social networks, a Twitter graph would be a directed graph because you can follow a person without this person following you. The Facebook social network would be represented by an undirected graph, since friendships are always bi-directional. For this thesis, we will only consider two-way friendships and thus we will always work with undirected graphs.

Edges can further be weighted, with the weight of an edge indicating the strength of the tie between two nodes. This aspect is also not of importance in the networks studied in this thesis and therefore edges will not be weighted, or in other words will all have the same weight 1.

2.2.2 Graph Properties

Once an existing social network is transformed into its mathematical representation, a graph, we can analyze different properties of this graph in order to find out more about the structure of the network. Different properties of a graph will be presented and properties that are characteristic of real-world social networks will be explained. Knowing the characteristics of social networks will be very important to create artificial social networks (section 3.1) that are similar to real-world ones.

Degree The degree of a node is the number of edges adjacent to it. In a social network where edges represent friendships the degree is the number of friends a person has.

Distance The distance between two nodes is the number of edges of the shortest path between those two nodes. A *path* is a sequence of edges that connect a sequence of nodes, and the *shortest path* between two nodes is the path that connects those two nodes through the least number of edges (note: there can be multiple shortest paths between two nodes). In a social network where edges represent friendships between people, all your direct friends would have distance 1 from you, and friends of your friends would have distance 2.

Average Path Length The average path length is the average length of the shortest paths of all possible node pairs. It is an indicator of how well information or data can be transported through a network. A shorter average path length means that information will spread across the network faster.

Global Clustering Coefficient/Transitivity The clustering coefficient of a graph (also called transitivity) measures the tendency of nodes to cluster together and form cliques. A *clique* is a set of nodes where every node is connected to every other node. The global clustering coefficient is defined as $\frac{3 \times number\ of\ triangles}{number\ of\ connected\ triples}$, where a triangle is 3 nodes all connected with each other (a clique of size 3) and a connected triple is 3 nodes where at least one node is connected with both other nodes.

There also exists a local clustering coefficient which measures how close one node's neighbours are to being a clique.

Centrality The concept of centrality is a way to measure the importance of a node within a network. In social networks, it can be used to find influential users, e.g. users that have a big influence on others' opinion and taste. Centrality measures are functions that provide values for every node, which can then be sorted to produce a ranking from the most to the least central node.

Betweenness Centrality Betweenness centrality is a measure that gives every node a value which indicates its importance within the network. The number each node v receives is equal to the number of shortest paths that pass through v.

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{2.7}$$

σ_{st} stands for the total number of shortest paths from node s to node t, and $\sigma_{st}(v)$ is the number of those shortest paths that go through v.

Given this definition, the betweenness scores in large networks can be very high. To receive normalized scores $g \in [0, 1]$, one can divide g by the total number of node pairs not including v.

2.2.3 Social Network Characteristics

There exist many different social networks. There are social networks like Facebook where friends share content and keep in touch, scientific collaboration networks or also social ties on music streaming sites like last.fm [Las, 2014]. Of course the sizes and specific structures of those networks might vary, depending on the application area. But still there exist characteristics that most social networks seem to share and that makes it possible to distinguish them from other classes of networks. [Newman and Park, 2003] and [Zaidi et al., 2012] establish three main characteristics of social networks:

Scale-Free Network Social networks often are scale-free networks. A scale-free network has a degree distribution that approximately follows a power law. A *power law* is a function of the form $f(x) = \alpha x^{-\gamma}$. The fraction of nodes $P(k)$ with degree k is then $P(k) \sim k^{-\gamma}$, with γ often found to be in between 2 and 3.

The reason this node degree distribution is often found in social networks is that most people tend to know only few other people, while very few people (e.g. celebrities) have considerably more social connections. The process of *preferential attachment* (people who already have many connections over time tend to make even more connections than people with only few connections, for a detailed explanation see [Easley and Kleinberg, 2010]) accounts for this distribution. The development of this structure often comes very naturally, e.g. in a mathematic collaboration network where a mathematician has already contributed to many studies and is therefore quite popular, he is more likely to collaborate with even more people in the future than other, not very popular mathematicians.

Small-World Network Social networks also tend to reveal the small-world phenomenon. This phenomenon was first studied by [Milgram, 1967] and [Travers and Milgram, 1969] who conducted an experiment where people all over the United States were randomly chosen as "starters" that had the task to forward a message to a designated "target" person as fast as possible, with the only rule that they were only allowed to send the letter to someone they knew on a first-name basis. A surprisingly high number of messages reached the target, and the astonishingly low median of steps from the starter to the target was 6. This experiment has been the trigger of more research and has led to the term small-world network. A small-world network is a network with a low average

path length (the average path length grows logarithmically to the number of nodes).

Community Structure The third characteristic of social networks is that they tend to consist of communities. Communities are parts of the network with densely connected nodes, while connections between different communities are scarcer. There is no mathematical definition for community structure, which makes it often hard to algorithmically find in networks. However, there exist many approaches to this problem, of which the most important ones will be described in section 3.1.1. It is also easy to understand the presence of communities in social networks, *triadic closure* [Easley and Kleinberg, 2010] is one of the main reasons (two people who have a friend in common are more likely to get to know each other than two random people are) for the formation of communities.

2.2.4 Related Work

[Easley and Kleinberg, 2010] give a broad introduction into many of the network-related topics discussed in this thesis. [Newman and Park, 2003] show why and how social networks differ from other types of networks.

2.3 Social Recommender Systems

Conventional CF does not take into account any social relations that might exist between users to form an appropriate neighbourhood. The neighbourhood is selected solely by a similarity measure that compares users' ratings. This is done because it is assumed that similar users are the best predictors. But a user might give more weight to the opinion of a friend he personally knows than to the opinion of a complete stranger, and moreover a friend might know more about the user's preferences and he might also have a similar taste. Often people hang out with people sharing common interests and having similar tastes (e.g. music taste, travelling, political interest etc.). This gives rise to the combination of recommender systems and social networks –social recommender systems. Although there exist different approaches, here we will focus on social collaborative filtering (social CF), which will be introduced in more detail.

2.3.1 Social Collaborative Filtering

Social CF differs from CF only in the way the neighbourhood set N is chosen. Instead of taking into account the most similar users, one can take as neighbourhood set a user's direct friends in the network. Since the set of direct friends can be quite small, it could also be advisable to take into account the friends of a user's friends. This can be done for bigger distances, although this process of traversing the friend graph quickly gets very time-consuming and also does not make that much sense in terms of social contacts. With distance 2 or 3, it is already likely that there are many people in the neighbourhood set that you do not know at all, and with a distance of more than 3 it is almost certain that most people in the set will have almost no relation to you. In

addition to this, one can also apply a similarity threshold on the neighbourhood set in order to filter out friends or friends of friends that might not be similar to a user.

All other steps stay the same as in conventional CF; the similarity calculation between all users as well as the prediction calculation are performed in the same way.

3

Artificial Data Generation

3.1 Artificial Network Generation

There has been a decent amount of studies that analyzed the performance of social recommender algorithms with the help of real-world data. Examples include [Groh and Ehmig, 2007] who analyzed the social network of the german community Lokalisten [Lok, 2014], or [Liu and Lee, 2010] who distributed a survey to users of Cyworld (a popular South Korean social network site) [Cyw, 2014]. But obtaining good and complete real-world data can be a difficult task. Online social networking want to provide a certain amount of privacy for their users and they want users to trust them with their data. Additionally, there are also national laws concerning data protection and data security. For those reasons, it is often difficult to study real-world examples of social recommender systems. Amazon, Facebook and Netflix are examples of this situation.

While there is one suitable dataset from the music streaming platform last.fm (section) that is used in this thesis, it is also desirable to have artificial datasets with adjustable parameters in order to study the performance of social recommender algorithms in different settings. Therefore an algorithm needs to be found that constructs graphs with properties of a social network, e.g. scale-freeness (few nodes with very high degree, many nodes with low degree), small world (a low average path length) and a high clustering coefficient (**??**). Finally, social networks often have a clear community structure, which should also be found in the generated network.

Many network generation algorithms and models exist, a good overview can be found in [Zaidi et al., 2012]. The problem is that most algorithms produce networks with one or the other property, but fail to produce networks that have all the characteristics of a social network (including community structure). The authors of this paper therefore not only compare existing algorithms, but they also propose a new algorithm which seems to produce networks similar to real-world ones and having all desirable properties. The algorithm works as follows:

Social networks consist of groups of densely connected users, called communities, or sometimes even cliques (communities where all users are connected to all others). The first step of the algorithm is therefore to create cliques of various sizes. These will be the "building blocks" of the network. Adding cliques will give the network a high clustering coefficient, since many triangles are formed (a clique, no matter how many

nodes it contains, always has a clustering coefficient of 1). The sizes of the cliques have been shown by ... to follow an exponential distribution. After that, the cliques have to be connected to form one big network. In real networks, there also exist connecting points between different communities. A person might belong to a strongly connected community of friends and family and also to a community consisting of people from work. This person figures as a connecting point of those two (otherwise probably not connected) communities. The way these connections are achieved is to assign each node an "open connections" attribute which determines how many additional connections it can make. This correlates with the degree of the node, which is why we have to draw the number of open connections from a power law distribution. This results in few nodes having a very high degree and many nodes having a low degree. Finally the nodes are merged. Two nodes that still have open connections are randomly selected and merged into one, resulting in two connected communities. The open connection attribute of both nodes is lowered by 1 and then randomly one of the attributes of the two old nodes is assigned to the new node. This is done until there are no more remaining open connections.

1. The algorithm takes as input parameters a number of cliques k to be generated in the beginning. The cliques will have sizes between $minSize$ and $maxSize$.
2. Generate k cliques, each with a size between $minSize$ and $maxSize$, following an exponential distribution $p(x)\,dx = \frac{1}{\mu} \cdot e^{\frac{-x}{\mu}}\,dx$, with $\mu = 2$.
3. Each node receives an "open connections" attribute, drawn from an exponential distribution $p(x)\,dx = \frac{1}{\mu} \cdot e^{\frac{-x}{\mu}}\,dx$ with μ as input parameter.
4. While there are open connections, randomly draw two nodes that are not yet connected and that both still have at least one open connection. Merge the two nodes into one, lower both "open connection" attributes by 1 and randomly assign the new node one of the two attributes.

3.1.1 Community Structure Detecting Algorithms

As described above, social networks often reveal some sort of community structure, meaning there are groups of densely connected nodes and few connections between those groups. However, there is no mathematical definition for communities. Nevertheless, there exist different approaches to find community structure, of which the most important ones will be presented here.

To be able to compare the goodness of a particular division of a network into communities, a measure called **modularity** can be used. It measures the fraction of edges that are within the given communities minus the expected fraction of edges that would have been within the given communities if edges were distributed at random (keeping node degrees fixed).

To explain the calculation of the modularity of a given network division, we first consider the simple case of a division into 2 communities. For a graph with n nodes and

m edges, we define a membership vector s, with $s_v = 1$ if node v belongs to community 1, and $s_v = -1$ if node v belongs to community 2. We further need the adjacency matrix A, with $A_{vw} = 1$ meaning there exists an edge between nodes v and w, and $A_{vw} = 0$ meaning there is no edge. To calculate the expected number of edges between two nodes, we need a null-model where the degree of each node is kept but the edges are randomly rewired. The expected number of edges between nodes v and w is then $\frac{d_v d_w}{2m}$, with d_v being the degree of node v and $2m$ being the total number of rewiring possibilities. The actual number of edges between two nodes is just A_{vw}, either 0 or 1. We include the factor $\frac{s_v s_w + 1}{2}$, which is 1 if v and w are in the same community and 0 otherwise, which gives us the formula for modularity Q:

$$Q = \frac{1}{2m} \sum_{vw} (A_{vw} - \frac{d_v d_w}{2m}) \frac{s_v s_w + 1}{2} \tag{3.1}$$

The division of the whole term by $2m$ is merely conventional, so the value of the modularity score is in the range [-1/2,1). This formula can be extended to the case of more than two communities very easily by replacing the factor $\frac{s_v s_w + 1}{2}$ with a function $\delta(c_v, c_w)$ that is 1 if $c_v = c_w$ (that is, nodes v and w are in the same community) and 0 otherwise.

Minimum-cut

This is one of the oldest and simplest algorithms to partition a network into communities. It cuts the network into a fixed number of groups, usually of approximately the same size, minimizing the number of edges between the groups. This approach has its application areas where it works quite well (e.g. load balancing for parallel computing), but it is clearly not well-suited for detecting the communities in a social network, since it always finds a fixed number of communities (all of approximately the same size), regardless if they really exist in the network or not. In social networks, often communities of different sizes will have to be detected and the number of communities is unknown a priori.

Greedy Modularity Optimization

The most popular community detection methods use some form of modularity optimization. Exhaustive optimization of modularity has been proven to be an NP-complete problem [Brandes et al., 2006], thus in practice other, greedy optimization methods must be used. One of the most popular approaches using modularity optimization is the one proposed by [Clauset et al., 2004]. It belongs to the class of agglomerative methods. Methods of this class start with all vertices being in n seperate communities, and then repeatedly join communities until all vertices belong to one community. This results in a so-called *dendrogram*, where cuts on different layers represent different divisions of the netowrk into communities.

The algorithm of [Clauset et al., 2004] now joins at each step the two communities that result in the biggest increase (or smallest decrease) in modularity Q. At each step, a division with a certain value of Q is obtained and it is then trivial to choose the division

with the highest Q. This algorithm runs in time $O((m+n)n)$ or $O(n^2)$ on sparse graphs (which social networks often are).

Edge Betweenness

An algorithm that detects community structure with the help of edge betweenness () was proposed by [Girvan and Newman, 2002]. Other than the agglomerative algorithm of [Clauset et al., 2004] described above, this is a divisive algorithm. It starts with one single community containing all nodes, and then gradually removes edges which also results in a dendrogram revealing the community structure of the network. The algorithm is quite simple:

1. Calculate the betweenness for all edges in the network.
2. Remove the edge with the highest betweenness.
3. Recalculate the betweenness for all edges that were affected by the removal.
4. Repeat steps 2 and 3 until no edges remain.

The big disadvantage of this algorithm is that is runs in worst-case time $O(m^2n)$. This makes it unsuitable for large social networks. In our case (networks with a few thousand nodes), the edge-betweenness algorithm already takes impracticably much time.

Community Walktrap

This is an algorithm proposed by [Pons and Latapy, 2005]. Their approach is based on the assumption that random walks on networks tend to stay within communities, since there are many connections inside communities and only few between them. It runs in worst-case time $O(mn^2)$, but since most real-world networks are very sparse, it usually runs in $O(n^2 \log n)$.

[Pons and Latapy, 2005] introduce a distance measure r between two nodes, which is large if the nodes are in different communities and small otherwise. This distance is computed from the information given by random walks in the graph. The algorithm itself then is also an agglomerative one; it starts with n communities containing one node each, and then merges communities based on their distance. At each step, the two communities are merged that minimize the mean of the squared distances between each node and its community.

3.2 Artificial Rating Generation

The generation of artificial ratings is a field that has not been in the focus of recent research like network generation. The application area of artificially generated ratings is very limited and mostly constricted to empiric research and experiments. In the next section, the design choices that were made are explained and justified in more detail.

3.2.1 Design Choices

First, a look at the existing dataset from last.fm that is used in this thesis shall be taken. The rating set contains 92'834 ratings of 1'892 users on 17'632 artists.The rating distribution (figure 3.1) is very skewed to the left, indicating that users give most items a rather low rating and only few items have a very good rating. The outlier at rating 5 is due to the normalization process which is done in a way that normalizes each user's personal favorite artist to a rating of 5. Therefore in the specific setting of online music streaming, we can assume that users have one or a few favorite artists that they listen to significantly more than to all others.

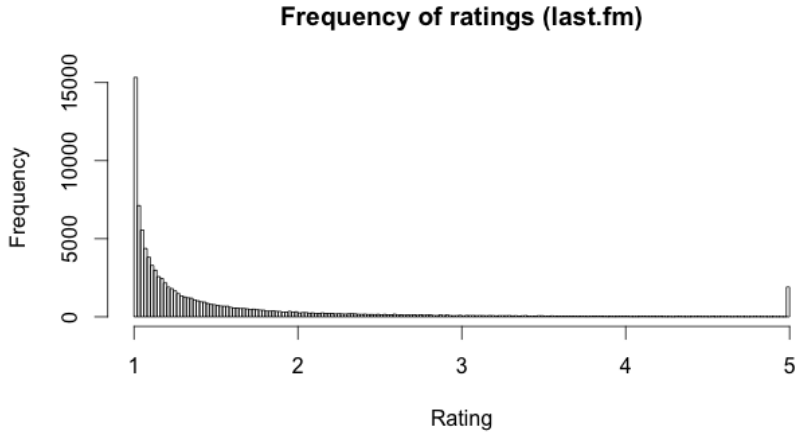

Figure 3.1: Frequency of ratings of the last.fm dataset

Since it is assumed and to be shown that there exists a certain correlation between friends' ratings, this also has to be taken into account for the rating generation. Varying the degree of correlation will help to show if social network information can increase collaborative filtering performance. Further, to create a realistic rating generation model, there needs to be some sort of quality differences between items, as well as different user and community tastes. For the algorithm, we will draw those three factors from an exponential distribution. It is realistic to assume that there are very few items of very high quality and a lot of low quality items, and the same goes for the tastes of individuals and groups.

Furthermore, the number of ratings each user provides is assumed to follow a normal distribution. This distribution choice can not be verified or falsified in the last.fm dataset, since this set has been arranged so that almost every user has given 50 ratings.

3.2.2 The Algorithm

The rating generation algorithm takes as starting point a previously generated social network (section 3.1) and the resulting community partition, resulting from applying a

community detection algorithm that places every user in a single community. Once the number of users n and the k communities they belong to are known, every user i is given a taste n_i drawn from an exponential distribution. Let there be a total number of t items, and every item j is also given a quality t_j drawn from an exponential distribution, as well as every community c is given a community taste k_c. All these tastes are afterwards normalized to be in the range the creator wants the ratings to be (often $[0, 1]$ or $[1, 5]$).

Before ratings are created, every user i also receives a number of ratings l_i he will give, which is drawn from a normal distribution. The mean and standard deviation of this normal distribution can be seen as parameters; they directly determine how sparse or how dense the resulting rating matrix will be.

As it can be assumed that not only the ratings of users in the same community correlate, but also the set of items they have rated, each community also receives a community item set Q, with q items, drawn randomly from the whole item set. The total number of items t is a parameter of the algorithm as well as q.

The next step is to create for every user the specified number of ratings. User i's rating on item j depends on 3 factors: item j's quality, user i's personal taste and user i's community's taste. It is determined by the following procedure:

1. Choose a user i.
2. Randomly choose an item j. With probability p, choose it from the whole item set, and with probability $1 - p$, choose it from user i's community's item set.
3. If the user has not already rated this item, calculate $r_i(j)$ as $a \cdot t_j + b \cdot n_i + c \cdot k_c$, with $a, b, c \in [0, 1]$ and $a + b + c = 1$. $r_i(j)$ is thus a convex combination of the three factors t_j, n_i and k_c.
4. Repeat steps 2 and 3 until user i has l_i ratings.
5. Repeat steps 1 to 4 for all users.

4

Experiments

To evaluate different versions of collaborative filtering algorithms and compare their performance to social collaborative filtering algorithms, simulations were run with the help of software designed specifically for this purpose. This chapter explains the experiments that were conducted and explains and interprets the results. The five basic experiments were:

1. Different versions of user-based collaborative filtering are evaluated on the last.fm dataset.
2. Different versions of social user-based collaborative filtering are evaluated on the last.fm dataset and their performance is compared to the performance of the conventional ones.
3. Different artificially generated networks and ratings are used to again compare collaborative filtering and social collaborative filtering and find conditions under which social collaborative filtering might perform better.
4. The hypothesis that central users in communities might be better predictors was analyzed.
5. The hypothesis that social collaborative filtering works better for certain users was analyzed.

4.1 Experimental Setup

4.1.1 Last.fm Dataset

The dataset from last.fm is publicly available and can be obtained from the "Social Computing Research at the University of Minnesota" website [Gro, 2014]. It contains information about 1892 (anonymized) users and 17632 artists. Since the users on last.fm do not explicitly give ratings to an artist, the degree to which a user likes or dislikes an artist is expressed as "listening-count" (how many times a user has listened to an artist). There are 92'834 $(user, artist, listeningcount)$-tuples. The dataset further contains 12'717 bi-directional user friend relations in the form of $(user_i, user_j)$-pairs. The

dataset further also contains informations about tags that users assigned to artists, but these will not be used in the experiments. However, the friend relations will be used and are indeed very important for the social collaborative filtering algorithms.

Normalization and cleaning-up of the Dataset

The rating file consisting of $(user, artist, listeningcount)$-tuples can already be used as input for a recommender algorithm. With k-fold cross-validation, the tuples can be partitioned into training and test sets. The problem is that the performance of the algorithms will be quite hard to evaulate, since the RMSE that is often used to measure the performance largely depends on the range of the ratings. The listeningcounts that are used as ratings in the last.fm dataset range from 1 to 352'698. There are users who have listeningcounts in the range of a couple of dozen, and others that only have listeningcounts beyond 1'000. This suggests that the listeningcounts need to be normalized so that the resulting RMSE's can acually be interpreted to make statements about the performance of the recommender algorithms.

There are a few steps in the normalization and clean-up process. Firstly, the few users that have the same listeningcount 1 for every artist they have listened to are completely removed. There is no information about the preferences of those users contained in their ratings. It is justifiable to do this since real recommender systems will hardly ever have to deal with a user that gives every item the same rating. Secondly, users with less than 10 ratings are also removed. This is done because of the partitioning of the dataset into training and test sets. There should always be users in both the partitioning and the training set, even if k is as high as 10.

After these removals have been performed, the listeningcounts need to be normalized in the following way:

Data: All $(user, artist, listeningcount)$-tuples $x = (u, a, l) \in X$, min (e.g. 1), max (e.g. 5)
Result: Normalized $(user, artist, rating)$-tuples $x' = (u, a, r) \in X'$ with r between min and max
for $u \in X$ **do**
  currentMin = findMinimum(u);
  currentMax = findMaximum(u);
  for *each* x *with* $x(u) = u$ **do**
    normalizingFactor = $\frac{x(l) - currentMin}{currentMax - currentMin}$;
    r = $normalizingFactor \cdot (max - min) + min$;
    $x' = (u, a, r)$;
  end
end

Algorithm 1: Normalize Ratings

After this algorithm is run, all ratings are in the range $[min, max]$ and are suited as input for the recommender algorithms. As a consequence, the friend relations also need to be updated. All relations containing one of the users removed beforehand are also

removed. This leaves us with 92'770 of the original 92'834 ratings and 12'508 of the original 12'717 friend relations.

4.1.2 K-fold Cross-Validation

Cross-validation is a technique used in statistical analysis to evaluate the accuracy of a prediction model on a given dataset. The dataset is partitioned into a training set, which will be used to "train" the model or algorithm, and a test set, which will be used to measure the prediction accuracy.

In k-fold cross-validation, the dataset is randomly split into k subsets of equal size. One of the subsets will serve as the test set and the other $k-1$ will be used as the training set. This is repeated k times, with each subset serving as test set exactly once. This method has the advantage over completely random splitting that each data point is guaranteed to be used in the test set (exacty once).

4.2 Experiment 1

The first experiment was run on the normalized last.fm dataset with the goal of comparing the performance of different versions of user-based collaborative filtering algorithms. The dataset was split using 5-fold cross-validation. To adjust the cross-validation to this particular setting, the process was adjusted a little. Instead of randomly partitioning the 92'770 ratings into 5 subsets, this was done for every user's ratings seperately. This extra step was taken to ensure that for every user there is training data as well as test data.

The two different similarity measures and the three prediction measures presented in section 2.1.3 were tested in all possible combinations. The neighbourhood sizes were determined either by a fixed number n (from 1 up to all users) or a similarity threshold λ was applied (from 0.0 to 1.0). All resulting parameter combinations were applied to the 5-fold cross-validated dataset, and this was done 5 times, resulting in 25 runs for each parameter setting. The resulting RMSE's were averaged over those 25 runs. The results can be seen in figures 4.1 and 4.2.

It can be seen that the differences between the different similarity measure/prediction measure combinations are relatively small. Averaged over all different neighbourhood options, the combination of cosine similarity and adjusted-weighted sum performs best. Looking at the prediction measures, one can quickly conclude that the consideration of the average rating of a user (as done by the adjusted sum) is more important and makes the predictions more accurate than the consideration of the weights. In fact, the adjusted sum performs almost as good as the adjusted-weighted sum.

Looking at the shapes of all curves, two things stand out: The maximum at neighbourhood $n = 50$ and the maximum at threshold $\lambda = 1.0$. At first sight, this is quite astonishing, since other studies have shown a local minimum at neighbourhoods of around 50. It seems weird that there is no such minimum, but instead a clear maximum. To explain these effects, one important realization is to understand that the size n of the

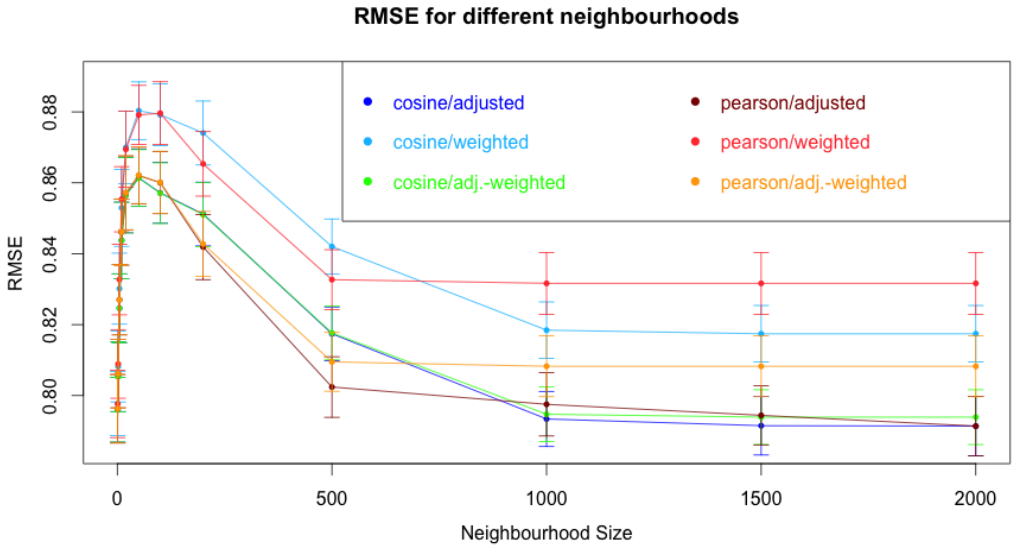

Figure 4.1: Results for neighbourhood

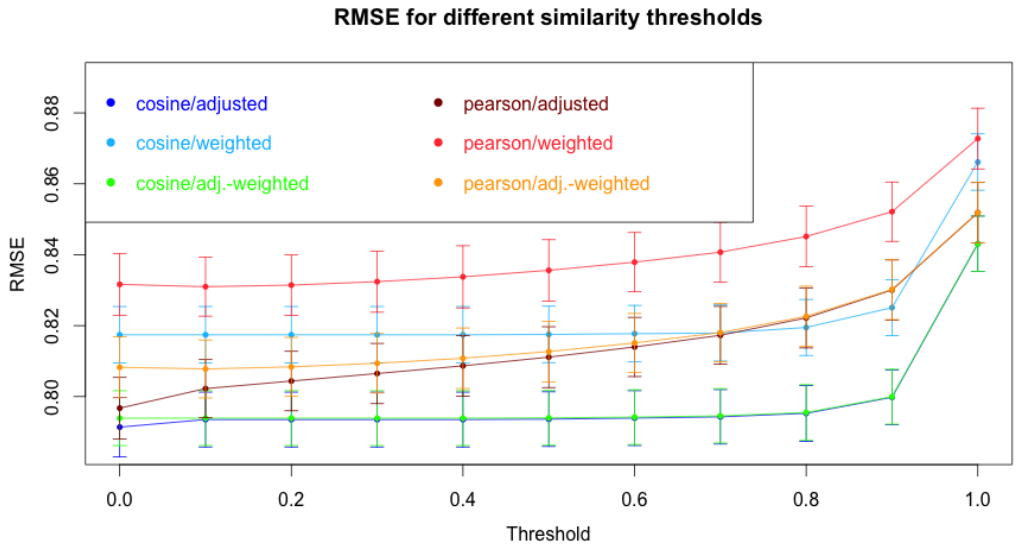

Figure 4.2: Results for threshold

neighbourhood N is not necessarily equal to the number of users included in the prediction calculation. As explained in section 2.1.3, out of all users in N, only the ones who have actually rated item j can be included in the calculation. In sparse rating matrices,

chances are that the actual set of users included in the prediction calculation is much smaller than n. For the last.fm dataset with 1'873 users and 17'612 items after normalization, there are 1'873 x 17'612 = 32'987'276 possible ratings of which only 92'770 are actually provided. The rating matrix thus has a density of only 0.3 % or a sparsity of 99.7 %. The results suggest that in a setting with very sparse rating matrices, larger neighbourhoods should be chosen so that enough users are included in the prediction.

The dataset of last.fm unveils another characteristic that amplifies this effect: There are many items that have only received one rating. Of course, when the rating for one of these items falls into the test set and has to be predicted, there is no possible information from other users' ratings to take into account. In this case, the algorithm chooses as prediction just the average rating of user i over all his ratings. The experiments showed that the average had to serve as prediction in $^1/_2$ of the predictions for a fixed neighbourhood of size 70 (with an average of only 1.87 users per prediction that had actually rated the item). The average of users actually included in the prediction hit 50 at $n = 1050$.

The smaller actual neighbourhood sizes explain why the algorithm gets better with very large neighbourhoods, but they cannot be responsible for the good performance with very small neighbourhoods. The answer for the good performance of the algorithm with neighbourhoods ¡ 10 lies in the performance of the average approach. This approach just returns a user's average over all his ratings as a prediction, and it actually performs pretty well on this dataset. The average RMSE over 5 5-fold cross-validations was 0.78. With very small neighbourhoods, a lot of averages have to be used, which makes the predictions better.

In the case of a similarity threshold, the explanation has to be found elsewhere. There, the actual neighbourhoods often have a decent size. The problem of a similarity threshold of 1.0 is that only users whose ratings are perfectly correlated are included in the prediction. This is, with very few exceptions, only the case if the users have co-rated only one item and given it the same rating. Of course, the similarity of those users results to 1.0, but with only one co-rated item the meaningfulness of this number can be questioned. With a similarity threshold of 1.0, consequently a lot of the users in the neighbourhood are not necessarily very similar to user i, which explains the large RMSE for all prediction and similarity measures. In the experiments, the average number of co-rated items for threshold 1.0 was 1.00 while for threshold 0.9 it was 3.10.

4.3 Experiment 2

The second experiment was conducted to measure the influence of the inclusion of social network information on the performance of collaborative filtering. For this, different versions of social collaborative filtering were implemented and tested on the same dataset that was also used for experiment 1. As prediction measure, the adjusted-weighted sum was used and the similarities between users were calculated with the cosine similarity coefficient.

As explained in section 2.3.1, the main difference of social collaborative filtering in

comparison to the conventional approach is the choice of the neighbourhood set N. Instead of using either a fixed size of the most similar users or applying a threshold, the users closest to user i in the social network can be chosen as his neighbourhood. In this experiment, the friend graph of every user was traversed with a breadth-first search (BFS), and different levels of the resulting node tree were taken as neighbourhoods. If all nodes with distance 1 are included, this corresponds to all direct friends of a user. Distance 2 corresponds to the user's friends and all their friends, and so on.

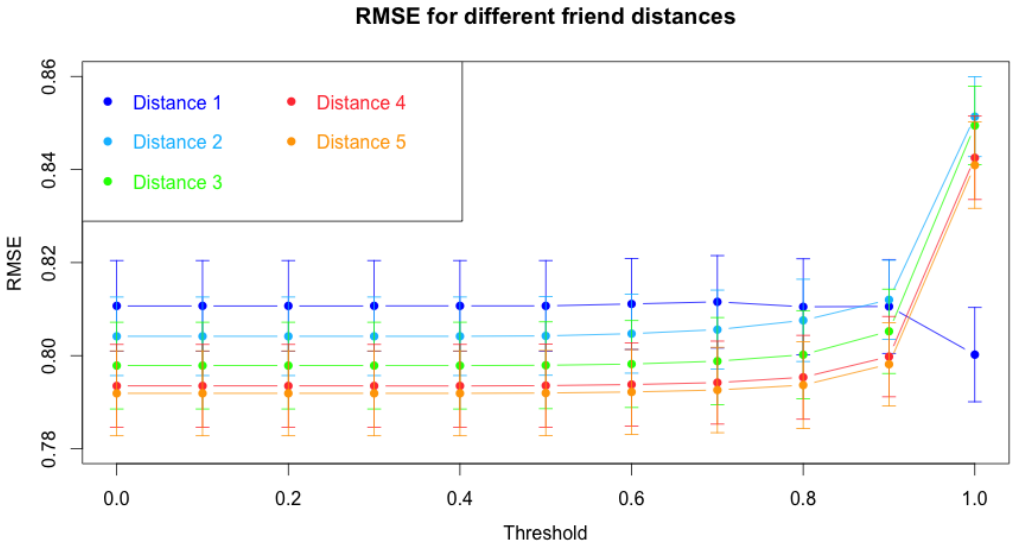

Figure 4.3: Results for social collaborative filtering

The results seen in figure 4.3 show some interesting behaviour. The case where only a user's direct friends are included in the neighbourhood set (*Distance 1*) shows slightly different behaviour than the others. While all other algorithms show a significant drop in performance with a similarity threshold of 1.0, the *Distance 1*-algorithm performs best at this threshold. The explanation for this behaviour can also be found in the fact that with small neighbourhoods and large thresholds, the user's average must be used for a lot of predictions, because there are no users in the neighbourhood who have rated the item to predict. Since the average algorithm (see experiment 1, section 4.2) actually also performs better than the social user-based CF algorithms, this makes the *Distance 1*-algorithm better at threshold 1.0.

The other algorithms with larger distances consequently have larger neighbourhood sets. Due to this, their behaviour is very similar to the conventional CF algorithms, which also show a performance drop at threshold 1.0. The explanation for this is given in section **??**.

Another factor that needs to be considered is the runtime of the algorithms. While in

the conventional CF, this factor did not play a big role and all algorithms had similar runtimes, the social CF algorithms show significant differences.

Algorithm	mean RMSE	mean runtime (s)
Distance 1	0.810	0.25
Distance 2	0.810	2.19
Distance 3	0.804	28.16
Distance 4	0.799	96.28
Distance 5	0.797	127.66

Table 4.1: RMSE and runtime of all social CF algorithms

Table 4.1 shows the mean RMSE and the mean runtime of the social CF algorithms over all runs. While the gain in performance with larger friend distances is marginal, the time for the computation of the neighbourhood set increases significantly. This is largely due to the breadth-first search-like traversing of the friend graph, that takes a lot of time with increasing distances.

These results suggest that the benefit gained from including friends with distance 3 or more does not outweigh the loss in performance. Since the results very much resemble those from conventional CF, it can even be assumed that the inclusion of information from the social network does not improve performance for this specific dataset from last.fm at all. If this were the case, we would expect the algorithms who include socially close users (distances 1 or 2) to perform better.

4.4 Experiment 3

After experiments 1 and 2 which were both run on the same dataset from last.fm, experiment 3 was run to compare the performance of CF and social CF in different environments. For this, instead of real-world data, artificially generated data was used. With a social network generated by the algorithm described in section 3.1, rating datasets were generated with different parameters. The parameters used are described in table 4.2.

The rating generation algorithm was run with all these parameter combinations, with the restriction that $a + b + c = 1$. This resulted in 160 test runs. For all runs, the same generated social network was used, to prevent any random differences in the results that could have been caused by the network generation.

Every generated rating set was partitioned using 5-fold cross-validation and both conventional and social CF algorithms were run on it. Conventional CF was run with different similarity thresholds and social CF was run with neighbourhood sets consisting of direct friends and friends of friends (distances 1 and 2), also with different similarity thresholds applied.

Parameter	Variable	Values	Explanation
Size of community item set	q	50, 100, 200, 400	This is the size of the item set that each community has. It is included in the rating generation so that users of the same community have a higher probability of rating the same items.
Probability of community set	p	0.05, $^1/_3$, $^2/_3$, 1	This is the probability that for a user's rating, the item is chosen from the whole item set. 1 - p is the probability that it is chosen from the user's community's item set.
Influence of item quality	a	0, $^1/_3$, $^2/_3$, 1	This is a factor in the range [0,1] that determines the influence of the item quality in the rating generation
Influence of user taste	b	0, $^1/_3$, $^2/_3$, 1	This is a factor in the range [0,1] that determines the influence of the user taste in the rating generation
Influence of community taste	c	0, $^1/_3$, $^2/_3$, 1	This is a factor in the range [0,1] that determines the influence of the community taste in the rating generation

Table 4.2: Rating generation parameters

The results are interesting and complex. It would be tedious to explain the outcome of every parameter combination in detail, but rather some clear tendencies and dependencies are highlighted.

- It is clear that the average size of the neighbourhood is largest with the conventional approach, since it includes all users before applying a similarity threshold. The *distance 2*-algorithm uses smaller neighbourhoods, while the *distance 1*-algorithm results in the smallest neighbourhoods. As a consequence, in settings where community taste has little or no influence (c is low), conventional CF performs best, followed by distance 2 social CF and distance 1 social CF. With only your friends in the neighbourhood, the actual set of users included in the prediction is too small to make good predictions (since there is no large correlation between friends). As an example see figure 4.4. In this case, the item quality accounts for $^2/_3$ of the generated rating and the user's individual taste for the other $^1/_3$. Since item quality is an important factor, the inclusion of more users who have also rated the item

will make the prediction better. The massive drop in performance at threshold 1.0 can be explained again by the shrinking of the neighbourhood set and thus the smaller number of users actually included in the prediction.

- When the community taste has a lot of influence on the performance (c is large), we should expect social CF to perform better in comparison to conventional CF. This is indeed the case, as can be seen in figure 4.5. They direct friends of a user are likely to be in the same community and since the community taste has a big influence, those users are better predictors than other users in the network. The significant drop at threshold 1.0 is again due to the neighbourhoods getting really small, with the consequence of having a lot of prediction made by taking users' average ratings. The average-based algorithm performs worse than the CF algorithms in this setting (with a mean RMSE of 0.762), as opposed to the last.fm dataset where it showed better performance than the CF algorithms.

- All the algorithms show special behaviour at similarity threshold 1.0. This threshold implies that users whose ratings perfectly correlate with a user's ratings should be included in the prediction. Doing this does not make that much sense in most environments. Requiring perfect correlation will in many cases mean to leave out very similar users who might have co-rated a large set of items and only show minor differences in rating behaviour. But one differing rating is enough to make the similarity drop below the threshold of 1.0. The problem is therefore that in most environments, users only have similarity 1.0 if they have only co-rated a small number of items (most often 1 item) and those ratings are the same.

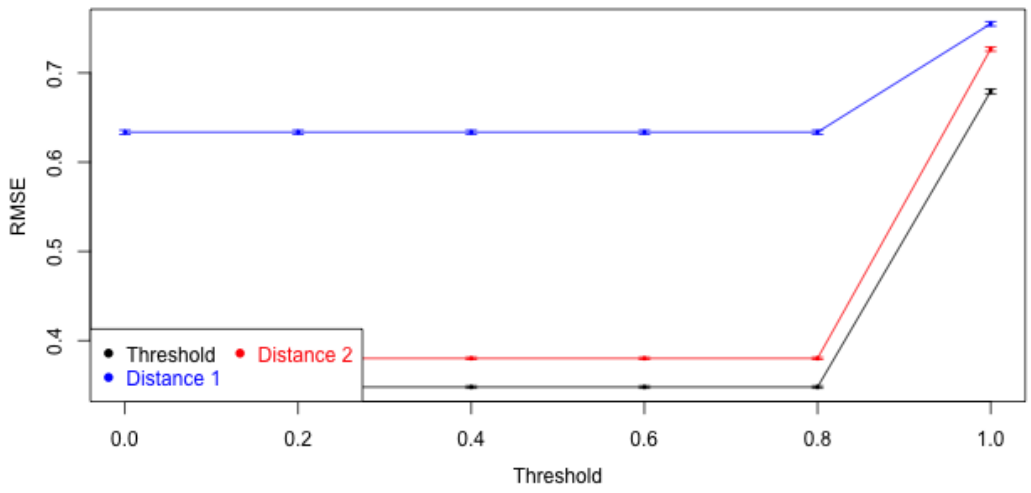

Figure 4.4: RMSE's for parameters q = 50, p = $^1/_3$, a = $^2/_3$, b = $^1/_3$, c = 0

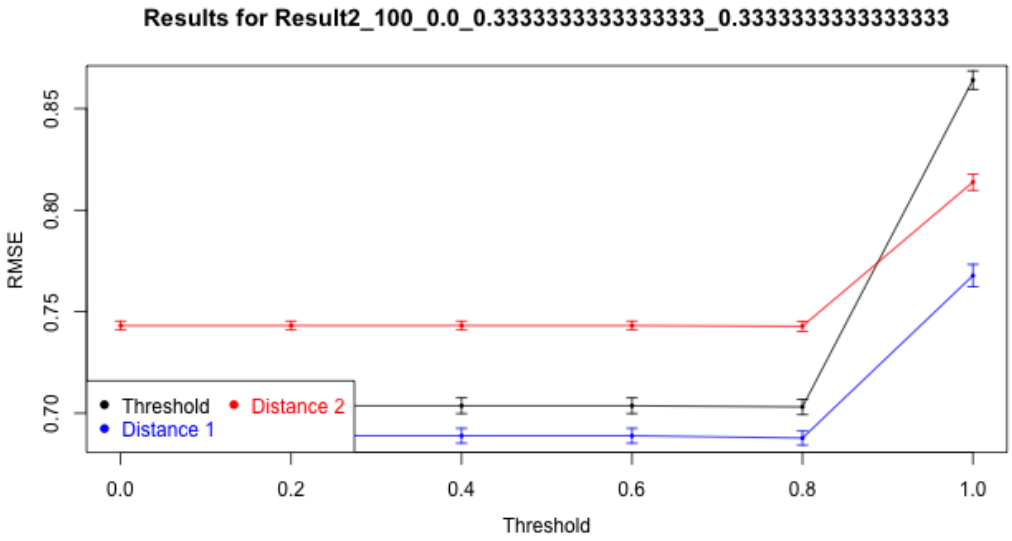

Figure 4.5: RMSE's for parameters q = 100, p = $^1/_3$, a = 0, b = $^1/_3$, c = $^2/_3$

4.5 Experiment 4

Experiment 4 aimed to analyze if users that are central in a social network or in a community within the network could be better predictors than others. This hypothesis was tested on the last.fm dataset. Using the similarities between users, it was tested if there was a stronger correlation of users with central and influential people than with the rest of a community.

The experiment was conducted in the following way:

1. A community detection algorithm was run on the last.fm social network and every user was put into one community. As community detection algorithm (see section 3.1.1), for this experiment the walktrap algorithm of [Pons and Latapy, 2005] was chosen since it seems to partition the network into more and smaller communities than the modularity optimization algorithm by [Clauset et al., 2004], which suffers a resolution limit (see [Fortunato and Barthelemy, 2007]) and would partition the network into only few communities that are very large.

2. The users are sorted by decreasing centrality. As centrality measure, betweenness centrality is chosen (see 2.2.2).

3. For each community, the n most central users are put into a group of *influencers*.

4. For each user, the average rating similarity between him and the *influencers* is calculated.

5. Steps 3 and 4 are repeated for all n from 1 to 100.

6. Additionally, the average rating similarity between users and their community is calcuated for each user, and then averaged.

The resulting similarities can be seen in figure 4.6. The red line is the average correlation of users with their community calculated in step 6. The blue dots indicate the similarities with the group of *influencers* calculated in steps 3 and 4, for n between 1 and 100. It is clear that the blue line will converge to the red line, since for very large n's the *influencer* group will consist of the whole community and thus be equal to the red line.

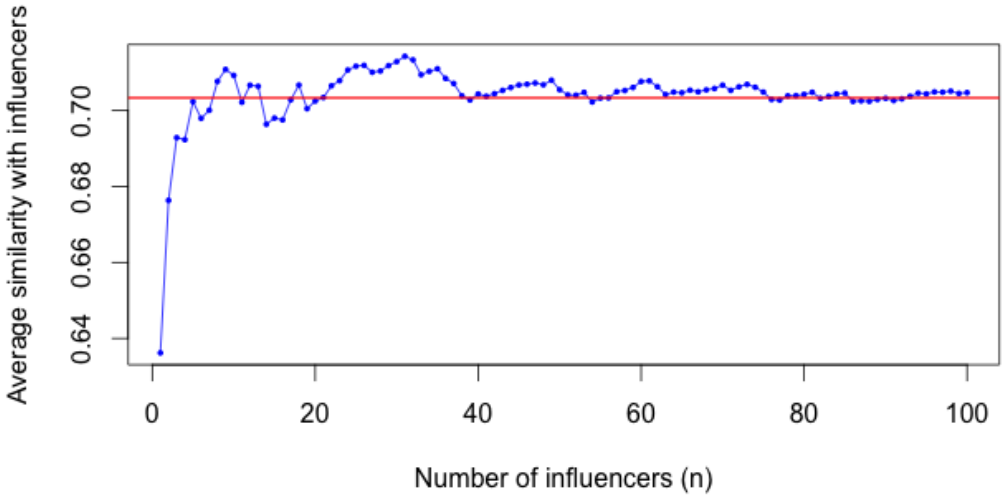

Figure 4.6: Average similarity of users with *influencers*

These results are a clear indication that some users in communities have influence on the taste (and therefore on the rating behaviour) of other users in those communities. At $n = 31$, the maximal similarity of 0.715 can be found, which is above the average rating similarity between users and their whole community (0.703). With this insight, one could possibly improve the social CF algorithm by taking into account information about centrality (e.g. by giving central users more weight in the prediction).

4.6 Experiment 5

The last experiment aimed to answer the question if there are users or groups of users for which social CF works better. After experiment 4 (section 4.5) showed that there exist users that are better predictors than others because of their position and influence in the

social network, the next question to ask is whether there are also positions in the network or factors that influence how well social CF performs for specific users in those positions.

To examine this question, the last.fm dataset was once again studied. The set was partitioned with 5-fold cross-validation and then conventional CF algorithms were run as well as social CF algorithms, each with different similarity thresholds from 0.0 to 1.0. Instead of calculating the average RMSE for a whole run as in previous experiments, RMSE's were collected for each user individually. Then, for each user, averages were taken over all conventional CF runs and over all social CF runs. The average social CF RMSE was subtracted from the average conventional CF RMSE for each user. Negative results imply that conventional CF works better for that user, while positive results imply that social CF works better. This error difference was stored for each user and users were sorted by their error difference, in ascending order. The result can be seen in figure 4.7.

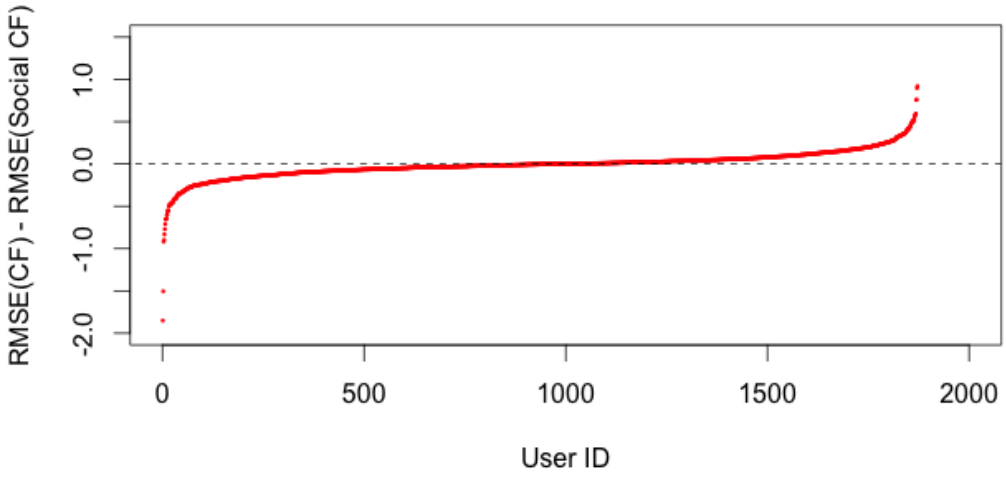

Figure 4.7: Average conventional CF RMSE - average social CF RMSE of all users, sorted from lowest to highest

Figure 4.7 shows that with very few exceptions, the difference between CF and social CF RMSE's lies between -0.5 and 0.5. There are only 15 users with a difference < -0.5 (the ones at the very left in figure 4.7), and only 13 users with a difference > 0.5 (the ones at the very right). The assumption that social CF might work better for users with many friends (central users) does not prove to be true, as there are central as well as decentral users on both ends of the scale.

The RMSE differences are quite small overall, revealing no specific group of users for which social CF seems to work better than conventional CF. However, it is clear that in

many settings, social CF will indeed work better for users that are in a community that influences the taste of its members.

4.7 Discussion

The experiments that were conducted showed the behaviour of CF and social CF in different situations and settings. The most important and interesting take-aways will be discussed in this section.

First, it appears that the cosine similarity measure performs slightly better than the pearson correlation similarity measure. Although both measures are very similar, the algorithms using cosine similarity perform a bit better, except in situations with very small neighbourhoods. However, the differences are not very large and the general behaviour of both measure seems to be the same. [Lathia et al., 2008] even stated that in their analysis of similarity measures, using a random similarity measure sometimes yielded better results than using one of the popular approaches (also see [Amatriain et al., 2011]).

Comparing the different prediction measures, the experiments suggest that the weighing of the ratings with the users' similarities does not affect the prediction as much as adjusting it to the average ratings. Using the adjusted sum yields almost the same results as using the adjusted-weighted sum, while using only the weighted sum results in worse performance.

A very important point to consider when using neighbourhood-based algorithms is the sparsity of the rating matrix. As already highlighted in experiment 1, a sparse rating matrix can lead to the effect of actually having very few users included in the prediction calculation, which worsens the performance of CF and social CF algorithms. This problem is known as new-user or new-item problem (also see section 2.1.2), and it is a known challenge of the CF approach to make predictions in environments with very sparse rating matrices.

Comparing conventional and social CF directly shows that both approaches work similarly well on the last.fm dataset. Figure 4.8 shows for both CF and social CF their best algorithm at each threshold, resulting in a (purely theoretical) mixed algorithm that chooses at each threshold the algorithm with the settings that performs best. The social CF algorithms have the edge at every threshold, with the largest difference at threshold 1.0. This is due to the performance of the *Distance 1*-algorithm, which actually gets better at threshold 1.0 (see section 4.3).

Experiment 3 showed that the use of social CF can enhance performance in settings where community taste has a stronger influence (see section 4.4). Although this experiment was performed on artificially generated datasets, the network and rating generation algorithms produce data similar to real-world datasets, and therefore the results strongly indicate that the algorithms would behave similarly in certain real-world settings.

The last two experiments analyzed the positions of certain users in the network, show-

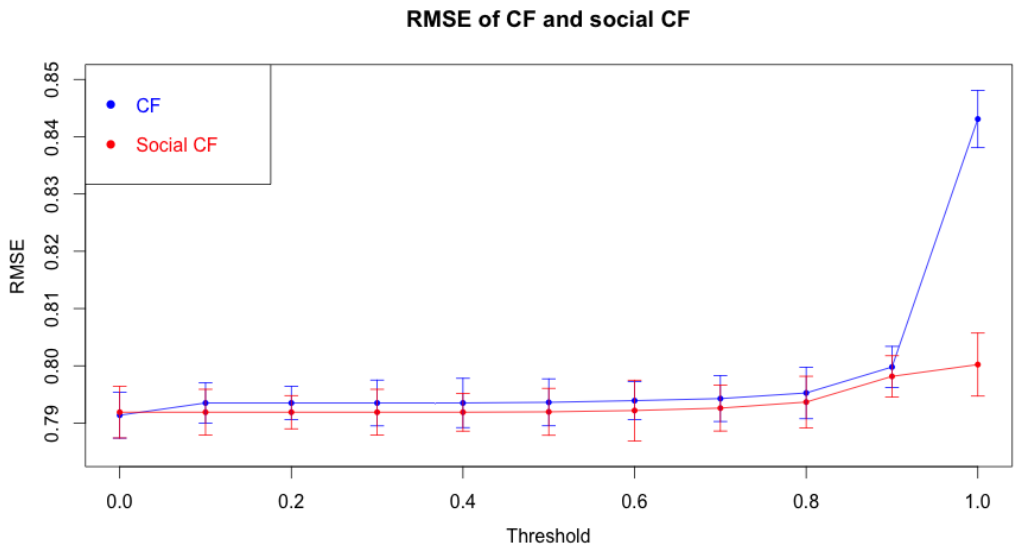

Figure 4.8: Best CF vs best social CF performances

ing that central users might indeed be better predictors than others. The existence of a rating correlation between users and the *influencers* of their community will have to be further analyzed on other real-world data. The last.fm dataset already revealed this correlation, but there are certainly recommender systems where the social network has more importance than in this specific music streaming environment (e.g. holiday recommendations).

5

Conclusion

The main contribution of this thesis are the findings discussed in section 4.7. CF and social CF algorithms were analyzed in detail and conditions were found under which social CF outperforms CF and vice versa. The influence of the existence of communities in social networks on recommender algorithms has also been analyzed, revealing that information about the community structure in a network might help to further enhance social CF algorithms.

There is still a lot of room for future research. Many of the findings will have to be confirmed by validating them on different real-world datasets. The hypothesis that there exist users for which social CF works better than CF remains to be proven true. Another interesting field of possible future work would be to focus on items rather than users, investigating whether there exist items or groups of items for which social CF works better than others.

References

[Cyw, 2014] (2014). Cyworld social network site. *http://www.cyworld.com*.

[Gro, 2014] (2014). Grouplens - social computing research at the university of minnesota. *http://grouplens.org/datasets/hetrec-2011/*.

[Igr, 2014] (2014). igraph network analysis tool. *http://www.igraph.org*.

[Las, 2014] (2014). Last.fm music streaming site. *http://www.lastfm.com*.

[Lok, 2014] (2014). Lokalisten freundes-community. *http://www.lokalisten.de*.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. (2005). Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749.

[Amatriain et al., 2011] Amatriain, X., Jaimes, A., Oliver, N., and Pujol, J. M. (2011). Data mining methods for recommender systems. In *Recommender Systems Handbook*, pages 39–71. Springer.

[Arias et al., 2012] Arias, J. J. P., Vilas, A. F., and Redondo, R. P. D. (2012). *Recommender systems for the social web*, volume 32. Springer.

[Brandes et al., 2006] Brandes, U., Delling, D., Gaertler, M., Görke, R., Hoefer, M., Nikoloski, Z., and Wagner, D. (2006). *On modularity - np-completeness and beyond*. Citeseer.

[Clauset et al., 2004] Clauset, A., Newman, M. E., and Moore, C. (2004). Finding community structure in very large networks. *Physical review E*, 70(6):066111.

[Easley and Kleinberg, 2010] Easley, D. and Kleinberg, J. (2010). Networks, crowds, and markets. *Cambridge University*.

[Ekstrand et al., 2011] Ekstrand, M. D., Riedl, J. T., and Konstan, J. A. (2011). Collaborative filtering recommender systems. *Foundations and Trends in Human-Computer Interaction*, 4(2):81–173.

[Felfernig et al., 2007] Felfernig, A., Isak, K., Szabo, K., and Zachar, P. (2007). The vita financial services sales support environment. In *PROCEEDINGS OF THE NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE*, volume 22, page 1692. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

[Fortunato, 2010] Fortunato, S. (2010). Community detection in graphs. *Physics Reports*, 486(3):75–174.

[Fortunato and Barthelemy, 2007] Fortunato, S. and Barthelemy, M. (2007). Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41.

[Girvan and Newman, 2002] Girvan, M. and Newman, M. E. (2002). Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826.

[Groh and Ehmig, 2007] Groh, G. and Ehmig, C. (2007). Recommendations in taste related domains: collaborative filtering vs. social filtering. In *Proceedings of the 2007 international ACM conference on Supporting group work*, pages 127–136. ACM.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

[Konstas et al., 2009] Konstas, I., Stathopoulos, V., and Jose, J. M. (2009). On social networks and collaborative recommendation. In *Proceedings of the 32Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 195–202, New York, NY, USA. ACM.

[Lathia et al., 2008] Lathia, N., Hailes, S., and Capra, L. (2008). The effect of correlation coefficients on communities of recommenders. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 2000–2005. ACM.

[Liu and Lee, 2010] Liu, F. and Lee, H. J. (2010). Use of social network information to enhance collaborative filtering performance. *Expert Systems with Applications*, 37(7):4772 – 4778.

[Milgram, 1967] Milgram, S. (1967). The small world problem. *Psychology today*, 2(1):60–67.

[Newman, 2004a] Newman, M. E. (2004a). Detecting community structure in networks. *The European Physical Journal B-Condensed Matter and Complex Systems*, 38(2):321–330.

[Newman, 2004b] Newman, M. E. (2004b). Fast algorithm for detecting community structure in networks. *Physical review E*, 69(6):066133.

[Newman, 2006] Newman, M. E. (2006). Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582.

[Newman and Park, 2003] Newman, M. E. and Park, J. (2003). Why social networks are different from other types of networks. *Physical Review E*, 68(3):036122.

[Parkes and Seuken, 2013] Parkes, D. C. and Seuken, S. (2013). *Economics and Computation*.

[Pazzani and Billsus, 2007] Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer.

[Pons and Latapy, 2005] Pons, P. and Latapy, M. (2005). Computing communities in large networks using random walks. In *Computer and Information Sciences-ISCIS 2005*, pages 284–293. Springer.

[Ricci et al., 2011] Ricci, F., Rokach, L., and Shapira, B. (2011). *Introduction to Recommender Systems Handbook*, pages 1–35. Springer US.

[Salton and McGill, 1983] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill, New York.

[Travers and Milgram, 1969] Travers, J. and Milgram, S. (1969). An experimental study of the small world problem. *Sociometry*, pages 425–443.

[Zaidi et al., 2012] Zaidi, F., Sallaberry, A., Melançon, G., et al. (2012). Generating artificial social networks with small world and scale free properties.

[Zheng et al., 2008] Zheng, R., Wilkinson, D., and Provost, F. (2008). Social network collaborative filtering.

A

Appendix

A.1 Implementation

For the experiments, the author of this thesis implemented a simulation software that is able to run and test different recommender algorithms on different datasets. It is also able to generate artificial data as described in chapter 3. Although the software is built specifically for the purpose of this thesis, it is built in a modular design so that it can easily be reused and quickly adapted to run similar algorithms and simulations.

The different parts and modules of the software will be presented here, without going into too much detail.

Recommender Algorithms There are different recommender algorithms, that all implement the same interface. Each recommender algorithm needs to implement only two methods, one that trains the algorithm when training data is provided, and a prediction method that returns a predicted value for an unknown rating. The recommender algorithms currently implemented are:

- User-based CF
- Item-based CF
- User-based social CF
- Average-based recommendation

The interface makes it very easy to add more algorithms.

Similarity and Prediction Measures All the similarity and prediction measures presented and used in the thesis (see section 2.1.3) are implemented. These are:

- Similarity measures:
  - Cosine similarity
  - Pearson correlation

- Prediction measures:
  - Adjusted sum
  - Weighted sum
  - Adjusted-weighted sum

This list can also easily be extended by other measures.

Simulation Running Application The core of the software is the application that actually runs simulations of recommender systems. One can choose which recommender algorithms to run with which similarity and prediction measures, what neighbourhood determination to use and on what datasets to run the simulations. If needed, the outputs (RMSE and runtime of each algorithm) are written to a text file and can be further processed.

As inputs, the simulation accepts rating lists in the form of $(user, item, rating)$-tuples. For social CF algorithms, social networks in the form of $(user_i, user_j)$-tuples are accepted. The ratings can also be partitioned using k-fold cross-validation (see section 4.1.2).

Artificial Data Generation The software is capable of generating artificial social networks as well as artificial rating data. The algorithms are implemented as described in chapter 3.

Centrality Testing This module is dedicated to the testing of different users' centrality and their influence on their community. It measures the rating correlation between users and influential people in their community. This module was built for experiment 4 (section 4.5).

The software is written in Java. The social network and graph analysis parts are written in C and called by the Java code through the *Java Native Interface* (JNI). Git was used for version control.

List of Figures

List of Tables