University of Zurich[UZH]

Computation and Economics Research Group

Thesis July 30, 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 advices and his cooperativeness. I would like to thank my soon-to-be wife Céline for her patience and for showing me so much understanding. And most importantly, I would like to thank God, because without Him I would not have been able to do this work.

Abstract

tbd

Table of Contents

1

Introduction

Recommender systems have become a widely used tool for users of different kinds of online services. They can help to 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 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 lead to the idea of using the information contained in social networks in order to enhance the performance of recommendation algorithms. The information contained in a social graph can be used to find users with similar tastes 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 [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 networks that could make them better predictors.

The main research questions that will be 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 convenctional collaborative filtering?
- Are there users that are better predictors than others?

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

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 properties is important for the incorporation of information into recommender systems and also for the generation of artificial social networks (chapter 3). The third section then combines the two fields by introducing recommendation algorithms that incorporate social network information, thus creating social recommender systems.

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 user might for example never hear about a good book that he would really like, if none of his friends knows that book. And often the almost infinite number of alternatives (e.g. 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 the user in his 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 where it will be used. It is important to carefully evaluate the requirements posed by the 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 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 to the user. 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.

Challenges

There are numerous challenges to this approach. One that was already mentioned before is the categorization of items. Where 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.

2.1.2 Collaborative Filtering

The method of collaborative filtering 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

methods, that is the high categorization-cost. The new-user problem still exists, but is less severe.

In user-based collaborative filtering, the idea is to find users with similar rating behaviour and predict ratings for an item based on these similar users' ratings for this item. Item-based collaborative filtering 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.

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 above, there exist many aspects that contribute to 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. One common measure to express the accuracy is the *root-mean-square error* (RMSE). For a series of n predictions $\hat{r}_n$ and actual ratings r_n, it is defined as follows:

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

User-based collaborative filtering tries to find users that are similar to a particular user. 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, of which two 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:

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

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

Cosine Similarity

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 orientation of two vectors. The two vectors x_i and $x_{i'}$ 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:

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

The similarities calculated with (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 certain 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).

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.

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 weighted 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

A problem 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 uses the average rating $\bar{r}_{i'}$ of each user i'.

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

Adjusted Weighted Sum

The adjusted weighted sum combines both aspects; it weighs the users' ratings by 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 therefore be much smaller than $|N|$.

Item-based Collaborative Filtering

A different approach is item-based collaborative filtering. Instead of finding users with similar rating behaviour, this method looks for similarities between items. In contrast to content-based methods, no information about the actual content of the items is needed, but instead the rating profiles of the items across all users are considered. If a set of items similar to item j can be found, the ratings user i has given those items can be used to estimate $\hat{r}_i(j)$.

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. The main difficulty with collaborative filtering can be the extremely high computational cost. Calculating all user- or item-similarities can be a serious issue, especially in domains with thousands of users and possibly millions of items.

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. A recommender algorithm can be evaluated and compared to others with respect to these properties. By far the most used property to assess the quality of a recommender algorithm is its prediction accuracy. Since recommender algorithms aim at providing recommendations to users, it is safe to assume that users prefer algorithms

that have a high prediction accuracy, that is they predict ratings that are close to the actual rating of a user.

[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 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.

Social networks can be mathematically represented by graphs. A graph consists of a set of nodes N and a set of edges 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.

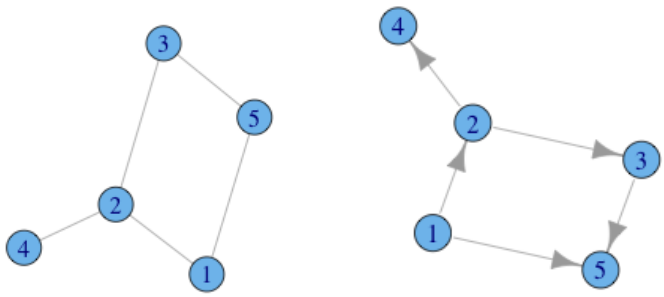

Figure 2.1: An undirected and a directed graph, each with 5 nodes and 5 edges.

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 as 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 also be weighted, with the weight of an edge indicating the strength of the tie between two nodes. This aspect is also beyond the scope of this thesis and edges will not be weighted, or in other words will all have the same weight 1.

2.2.1 Graph notions and 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 are presented and also properties that are characteristic for real-world social networks will be explained. This will play a big role in the graph generation section, where the task is to generate networks with realistic social network properties.

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 someone 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. In a social network where edges represent friendships between people, all your direct friends would thus 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 tendence 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 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.

These are 4 basic and very important graph properties. They can help to categorize graphs and to find typical patterns in certain categories of networks.

2.2.2 Related Work

2.3 Social Recommender Systems

2.3.1 Social Collaborative Filtering

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]. The main problem is to obtain real-world data. 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.

There is a very popular open source library for network analysis called "iGraph" [Igr, 2014]. It is available as R package, python library or C library. The algorithm described above was written in C and with the help of Java's JNI (Java Native Interface) included in the java code. JNI makes it possible to directly call methods from C libraries in the Java code.

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 that is used in this thesis (from last.fm, section) shall be taken. The rating set contains 92'834 ratings of 1'892 users on 17'632 artists. After normalization of the ratings (section), the rating distribution 3.2.1 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.

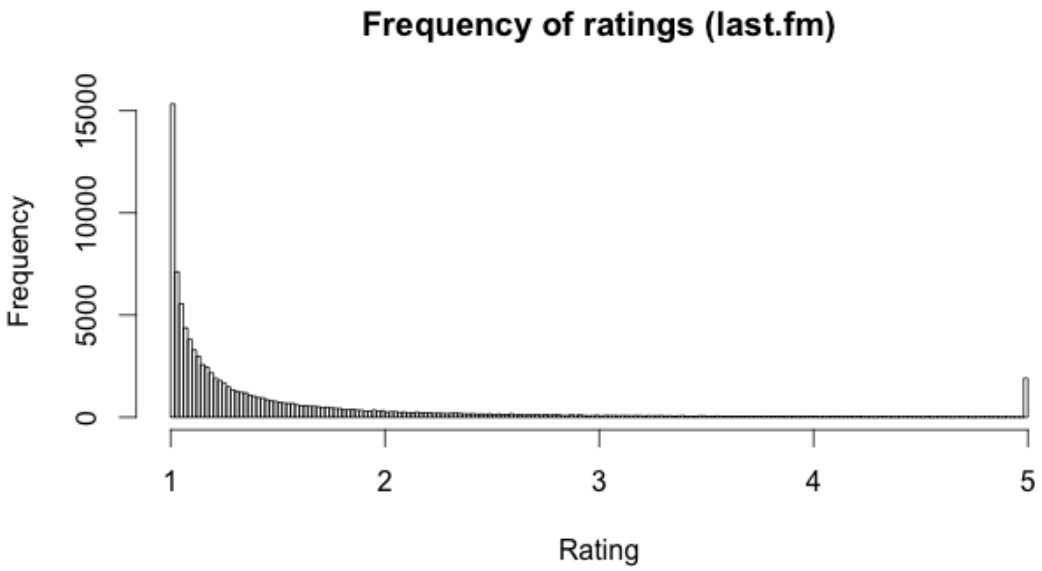

Figure 3.1: Frequency of ratings of the last.fm dataset

Since it is assumed and to be shown that there exists a certain correlation between friends' ratings, this also has to be included in the rating generation. Varying the degree of correlation will help to show if social network information can increase collaborative filtering performance.

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 compared to 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. In this particular setting, item-based collaborative filtering was not practicable. Calculating all user-user similarities for around 1'800 users (around 3'250'000 similarities) takes almost 100 times less time than calculating all item-item similarities for around 17'500 items (around 306'250'000 similarities), actually resulting in equal or even better performance.

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 **??** 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 4.2 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.

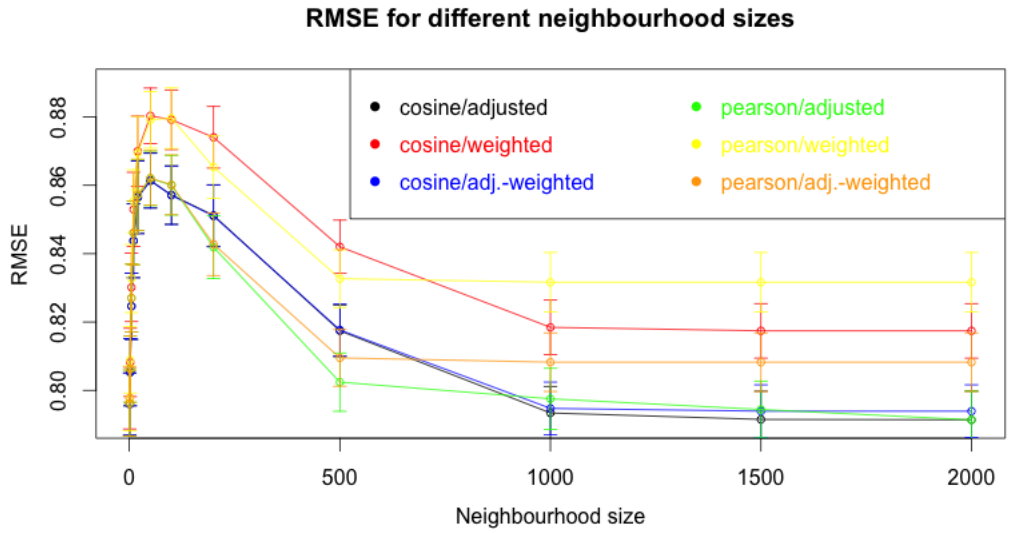

Figure 4.1: Results for neighbourhood

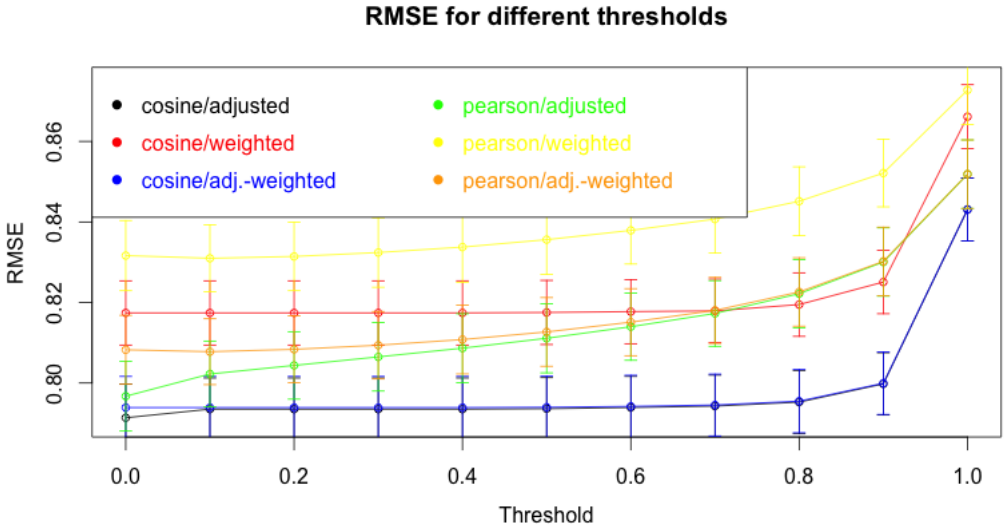

Figure 4.2: Results for threshold

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 neighbourhood N is not necessarily equal to the number of users included in the prediction calculation. As explained in **??**, 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 (4.2). As prediction measure, the adjusted-weighted sum was used and the similarities between users were calculated with the cosine similarity coefficient.

As explained in 2.3, 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 on level 1 are included, this corresponds to all direct friends of a user. Level 2 corresponds to the user's friends and all their friends, and so on.

4.4 Experiment 3

4.5 Experiment 4

4.6 Experiment 5

5

Conclusion

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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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 Lorem Ipsum

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam a tellus. Aliquam commodo dui non ipsum. Duis mollis nisi id turpis. Donec quis ipsum. Curabitur sed nibh. Morbi suscipit justo quis orci. Ut massa tortor, ultricies vitae, lacinia eu, facilisis eu, nisl. Nulla mattis urna sed metus imperdiet ornare. Praesent sodales. Etiam laoreet. Mauris quam magna, sagittis et, pharetra eget, congue vitae, arcu. Fusce sollicitudin justo. Suspendisse lectus. Sed lobortis dolor quis lectus scelerisque ornare. Integer purus. Phasellus vel elit at nibh sagittis lobortis. Aliquam iaculis malesuada eros. Mauris metus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam a tellus. Aliquam commodo dui non ipsum. Duis mollis nisi id turpis. Donec quis ipsum. Curabitur sed nibh. Morbi suscipit justo quis orci. Ut massa tortor, ultricies vitae, lacinia eu, facilisis eu, nisl. Nulla mattis urna sed metus imperdiet ornare. Praesent sodales. Etiam laoreet. Mauris quam magna, sagittis et, pharetra eget, congue vitae, arcu. Fusce sollicitudin justo. Suspendisse lectus. Sed lobortis dolor quis lectus scelerisque ornare. Integer purus. Phasellus vel elit at nibh sagittis lobortis. Aliquam iaculis malesuada eros. Mauris metus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam a tellus. Aliquam commodo dui non ipsum. Duis mollis nisi id turpis. Donec quis ipsum. Curabitur sed nibh. Morbi suscipit justo quis orci. Ut massa tortor, ultricies vitae, lacinia eu, facilisis eu, nisl. Nulla mattis urna sed metus imperdiet ornare. Praesent sodales. Etiam laoreet. Mauris quam magna, sagittis et, pharetra eget, congue vitae, arcu. Fusce sollicitudin justo. Suspendisse lectus. Sed lobortis dolor quis lectus scelerisque ornare. Integer purus. Phasellus vel elit at nibh sagittis lobortis. Aliquam iaculis malesuada eros. Mauris metus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam a tellus. Aliquam commodo dui non ipsum. Duis mollis nisi id turpis. Donec quis ipsum. Curabitur sed nibh. Morbi suscipit justo quis orci. Ut massa tortor, ultricies vitae, lacinia eu, facilisis eu, nisl. Nulla mattis urna sed metus imperdiet ornare. Praesent sodales. Etiam laoreet. Mauris quam magna, sagittis et, pharetra eget, congue vitae, arcu. Fusce sollicitudin justo. Suspendisse lectus. Sed lobortis dolor quis lectus scelerisque ornare. Integer

purus. Phasellus vel elit at nibh sagittis lobortis. Aliquam iaculis malesuada eros. Mauris metus.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam a tellus. Aliquam commodo dui non ipsum. Duis mollis nisi id turpis. Donec quis ipsum. Curabitur sed nibh. Morbi suscipit justo quis orci. Ut massa tortor, ultricies vitae, lacinia eu, facilisis eu, nisl. Nulla mattis urna sed metus imperdiet ornare. Praesent sodales. Etiam laoreet. Mauris quam magna, sagittis et, pharetra eget, congue vitae, arcu. Fusce sollicitudin justo. Suspendisse lectus. Sed lobortis dolor quis lectus scelerisque ornare. Integer purus. Phasellus vel elit at nibh sagittis lobortis. Aliquam iaculis malesuada eros. Mauris metus.

List of Figures

List of Tables