DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2017

A Hybrid Film Recommender System Based on Random Forest

Designing for simplicity and how it affects performance

ANDREAS BROMMUND

DAVID SKEPPSTEDT

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

A Hybrid Film Recommender System Based on Random Forest

Designing for simplicity and how it affects performance

ANDREAS BROMMUND

DAVID SKEPPSTEDT

Bachelor in Computer Science
Date: June 5, 2017
Supervisor: Jens Lagergren
Examiner: Örjan Ekeberg
Swedish title: Ett hybrid-rekommenderingssystem för filmer byggt med Random Forest - Designat för enkelhet och hur det påverkar prestanda
School of Computer Science and Communication

Abstract

Thanks to the internet an abundance of information is available just one click away. All this information is difficult to digest. Thus, a lot of time has been devoted to research how to build powerful and efficient information filtering systems. Recommender systems typically use either collaborative filtering or content-based filtering and there also exist hybrids of these methods. This thesis explores how complex a film recommender system need to be to obtain adequate performance. Therefore, a hybrid recommender system, TRÄD, was developed using k-nearest neighbour for collaborative filtering and random forest for content-based filtering. A dataset from MovieLens with ratings of films was used to train and evaluate the performance of TRÄD's recommendations. To evaluate how complexity affects performance, a comparison with a recommender system based on k-nearest neighbour and artificial neural networks was conducted. The results show that TRÄD is 7 percentage points less effective than the reference. However, the results show that random forest has reasonable performance. The fact that TRÄD use random forest instead of neural network is probably not the reason behind the result. It is more likely that too few attributes are considered in the content-based filtering and if more where to be considered a boost in performance would occur. In conclusion, it is clear that the combination stage is vital for performance of a hybrid recommender system and the rules that governs the combination of results from the collaborative and content-based are probably more important than the choice of classifier.

Sammanfattning

Tack vare internet har mängden av information exploderat, enkelt åtkomligt bara ett knapptryck bort. En direkt konsekvens av detta är det blir svårt att tillgodogöra sig information. Därför har mycket tid lagts på att forska och utveckla system som är bra på att filtrerar information. Rekommenderingssystem bygger vanligtvis på två olika metoder, kollaborativ- och innehållsfiltrering. Hybrider av dessa metoder är också vanliga. Den här rapporten undersöker vilken nivå av komplexitet ett rekommenderingssystem behöver ha för att uppnå adekvat prestanda. I denna rapport presenteras TRÄD, ett hybrid rekommenderingssystem för filmer. TRÄD bygger på k-nearest neighbour och random forest. För att träna och utvärdera prestandan av rekommendationerna från TRÄD användes en databas från MovieLens med betygsatta filmer. För att kunna utvärdera hur komplexitet påverkar prestanda gjordes en jämförelse med ett mer komplicerat rekommenderingssystem byggt på k-nearest neighbour och artificial neural networks. Resultaten visar att TRÄD har 7 procentenheter sämre prestanda än referenssystemet. Dock visar resultatet att random forest har rimlig prestanda. Att TRÄD presterar sämre är klart, men att det skulle beror på bytet av klassificerare är inte lika självklart. Det är mer troligt att innehållsfilteringsystemet tar för få egenskaper (hos filmerna) i beaktning och om TRÄD skulle undersöka fler egenskaper skulle troligtvis prestandan öka. Slutsatsen är att kombinationssteget är vitalt för prestandan i ett hybrid-rekommenderingssystem och att designa bra kombinationsregler är viktigare än valet av klassificerar.

Contents

Glossary

ANN Artificial neural network. 10, 11, 27, 28

CBF Content-based filtering. 6, 7, 14, 23, 26–30, 32

CF Collaborative filtering. 5–7, 9, 14, 22, 27–29, 33

CS-HRS Christakou-Stafylopatis hybrid recommender system. vi, 14–17, 26–32, 37

HRS Hybrid recommender system. 7, 28

kNN K-nearest neighbours. 7, 8

PC Pearson correlation. 7, 30

RS Recommender system(s). 3, 4, 6, 7, 16, 30, 31

TRÄD Our hybrid recommender system. vi, 2, 17, 19, 20, 22, 23, 25, 27–30, 32, 33, 38

Chapter 1

Introduction

In today's digital society the battle for our attention is torn between many on-line services e.g. Amazon, Spotify and Netflix. All these services have in common that they have a larger supply of items than their physical counterpart. Take e.g. Netflix, a streaming service, with millions of films just one click away. This is the essence of the phenomena called the long tail, a physical store could never offer a wider variety of films than Netflix does, simply because of storage limitations as well as not being able to customise the offering based on individual customers. As a consequence, leading the store to only offer the most popular films. However, Netflix will customise the experience for all customers but will also offer both popular and niche films. On the other hand, the average consumer, whose supply has skyrocketed, needs a way to find suitable films to watch. Therefore, it has become a paramount part of any online service to be able to make good recommendations that will keep the user around longer. To solve this problem, time and money has been devoted to research and development of recommender systems whose purpose is to generate valuable recommendations. [15, 12].

1.1 Purpose of this study

During the literature study, we encountered the article *A Hybrid Movie Recommender System Based on Neural Networks* [4]. This article caught our attention because it describes a complex recommender system in detail. This lead us to pose the question: When building a film recommender system, is it necessary to design a recommender system in such a complicated fashion or does a simpler approach suffice?

Knowledge about how to build simple yet effective software systems is of utter must importance in our industry. A well-known software design principle is "Keep it simple, stupid". The purpose of this study is to investigate how complex a film recommender system needs to be to performer reasonably well and what can be removed from the system without reducing the performance too much.

1.2 Problem statement

Building upon earlier work and the recommender system presented in [4], we want to explore whether there is a gain or penalty in performance when we build a hybrid recommender system that replace the neural network classifier with the random forests classifier. Furthermore, we aim to analyse how the removal of two attributes from the content-based filtering, synopsis and actor attribute impacts the performance of the system.

Thus, we propose the following question:
How does the performance of our recommender system (TRÄD) compare to the recommender system described in [4]?

The scope of this thesis is constrained to build a hybrid recommender system using the dataset MovieLens published 1998. The only attribute used in content-based filtering is film genre.

Chapter 2

Background

2.1 Recommender system

Today's digital services (Spotify, YouTube, etc.) presents their customers with an abundance of choice. A person using such a service is called a user and to aid users, services can apply information filtering to recommend items to users. Items can be many things, e.g. books, films, news, music etc. This type of software is called recommender systems (RS). Spotify, YouTube and Amazon are examples of services that highly utilise RSs[13].

For example, every Monday, all Spotify users are greeted by the *Discover Weekly* playlist. The advanced machinery at Spotify has generated personalised playlists for all users that should cater the users' individual music taste. The songs in a playlist is selected by the system based on what a user have previously listen to in combination with current music trends that Spotify can observer on their platform.

Recommender systems needs information to function, data about the users and items are crucial. This data can be collected either directly or indirectly. Directly means that the user of a service explicitly gives feedback on items, e.g. rating films they have watched. Indirectly means that the system analyses the users' interaction with the service, all input is implicit, users does not rate for a specific item. For example, user A watch film B to the end but does not leave a rating. However, the system will still consider A to like B. Direct and indirect data col-

lection is often used in combination[13]. This study focuses on direct information because the RS presented in this thesis considers films that users have rated.

	F1	F2	F3
U1	4	1	
U2	1		
U3		5	

Figure 2.1: U1, U2 and U3 represent users and F1, F2 and F3 represent films. The relation between a user and a film is the rating. A empty cell means that the film has not been rated yet. Recommender systems are designed to fill in the blanks, basically to help predict if the user will like the unrated films or not.

The relation between users and items are conveniently placed in a matrix, called the utility matrix or rating matrix, as can be seen in figure 2.1. A cell represents a user's rating for a specific item. Most of the cells will be empty because most users rate few items. The goal of RSs is to fill in the blanks, predicting what the users will like[12].

Strategies in RSs are divided into three categories, collaborative filtering, content-based filtering and a hybrid filtering method between the two aforementioned approaches.

2.2 Collaborative filtering

People typically rely on their peers for recommendations and to discuss decisions. This is the idea that collaborative filtering (CF) is based on and CF-systems are designed to mimic the power of word of mouth recommendations[7, 6]. In a CF system, similarity is found between users based on their preference on items they have in common. If they like the same items, they are similar and vice versa. The system will predict recommendations for a user based on the most similar users' ratings[14]. In essence, CF calculates similarity between users and recommends items derived from that.

Consider the following example, in a CF system we have two similar users, A and B. The system will recommend A the films that B enjoyed and gave a high rating. Films that A and B both have watched are not interesting, only films unknown to A. The films that are interesting should be clarified from equation 2.1 and from the Venn-diagram in figure 2.2.

$$A_{rec} = B - A \cap B \tag{2.1}$$

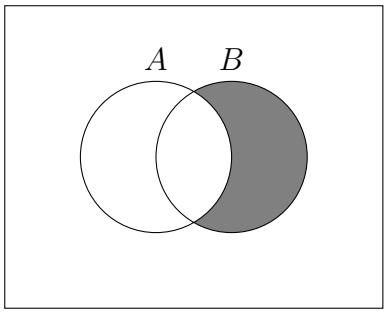

Figure 2.2: Shaded area is the films that B have seen, that A should be recommended

One of the most obvious problems with CF is that no recommendation can be made for any item without a rating. Another issue, is in a setting with few users and many items, similarity between users can be hard to find because of missing overlap of ratings[14]. Another problem is new users. Since there is no history of their preference, the system cannot find similar users. This is referred to as the cold start

problem. One way to resolve this problem is to subscribe a generic preference model to a new user that gradually changes as the user continues to interact with the service[13].

2.3 Content-based filtering

History plays an important role when making decisions, in RSs history is used to find patterns of what kind of items users liked before and this is used to recommend new items. Designing a RS with his approach is called content-based filtering (CBF). The system is designed to find a pattern examining attributes of items that the user previously liked. For example, if a person bought a few novels in the crime genre crime written by a specific author, a safe bet is to recommend another crime novel by the same author. The important part of a content-based filtering system is to decide which attributes will specify an item and how they will relate and match to a user's previous choices. In this study, where the items are films the following attributes could be used: genre, length, director, actors of the main characters[10].

Consider the following example, user U has rated film A and B positively. Both films are comedies which are both ca 90 minutes long and have a female protagonist. There exist three other films: C, D and E. C is an action film and will be discarded, D and E are both comedies, but E is 120 minutes long and has a male lead as where D is 99 minutes and the lead is female. Film D will be recommended by the CBF system.

Unlike CF, CBF systems can recommend items that has not been rated by any user. A drawback with CBF is that it suffers from over specialisation, it will not recommend anything unexpected. For example, if a user only rates action films then all recommendations are going to be action films. CBF systems have the same type of cold start problem for new users as CF systems. The preference for a user cannot be model without any history to which items the user likes[13].

2.4 Hybrid recommender system

Hybrid recommender systems (HRS) can broadly be defined as a combination of a CBF system and a CF system. HRS try to minimise the weaknesses exhibited in CBF and CF-systems. By playing on both methods strengths the accuracy of recommendations can rise when done correctly. [4]. Basically, a hybrid recommender system is a collaborative system and a content-based system that is combined using a ruleset. The rules describe in what way the results from the two systems should be prioritised and recommended.

2.5 Neighbourhood classifiers

Neighbourhood based classifiers are one of the most popular choice for RS that use collaborative filtering. For every user in a RS a neighbourhood can be created. A neighbourhood for a user is essentially a list of other users called neighbours. The neighbours are ordered by their similarity to the specific user. Another name for this type of classifier is k-nearest neighbours (kNN), where k is a threshold which decide the number of nearest neighbours to include. The reason to calculate similarity is twofold. First, it achieves the goal to select a sample of users that are close to the one compared, the so-called neighbours. Secondly, it gives a chance to weight the neighbours' opinions differently. There exist several ways of calculating similarity, e.g. Euclidian distance, Minkowski Distance and Cosine similarity. The similarity calculation discussed here is Pearson correlation (PC) as this was used in [4].

PC calculates the correlation between two users, the sign of the value is an indication of whatever the correlation is direct or inverse. The magnitude, between 0 and 1, equates to the strength of the correlation[5]. To calculate the correlation between two users u, v the following formula is used.

$$PC(u,v) = \frac{\sum\limits_{i \in I_{uv}} (r_{ui} - \overline{r}_u)(r_{vi} - \overline{r}_v)}{\sqrt{\sum\limits_{i \in I_{uv}} (r_{ui} - \overline{r}_u)^2 \sum\limits_{i \in I_{uv}} (r_{vi} - \overline{r}_v)^2}} \tag{2.2}$$

where r_{ui} is user u's rating for item i, $\overline{r}_u$ is the mean rating for user u, same for user v.

The purpose of neighbourhood classifiers is to simulate several friends the user could ask for film recommendations. As discussed, a list of neighbours needs to be generated based on similarity. Once the neighbourhood is created it can be used to predict ratings. A common method to classify an item i is to have a frequency list for the rating values. The neighbours are used to update the frequency list, if a neighbour have rated the film with 3 it increments the frequency counter for rating 3 with the similarity weight of that neighbour. When all neighbours have been considered the rating, whose counter has the maximum value is used as the classification for that film. This is formulated as:

$$v_{ir} = \sum_{v \in N_i(u)} \delta(r_{vi} = r) w_{uv} \tag{2.3}$$

where v_{ir} is the classification for the item i, $\delta(r_{vi} = r)$ is 1 if $r_{vi} = r$ is equal and 0 otherwise, $N_i(u)$ is the neighbours to u that has seen the film i, w_{uv} is the similarity weight for the user u and v[5].

Two approaches exist for collaborative filtering using kNN. User-user and item-item. The difference between these two methods is what similarity is calculated based on. In user-user the neighbours are similar users, in item-item the neighbours are similar items[5].

Consider the following figures, where circles represent users, squares represent films, black lines indicate which film a user have seen and the rating the user gave, green lines represent users or items which are similar, and the purple lines show which films should be recommended to a user.

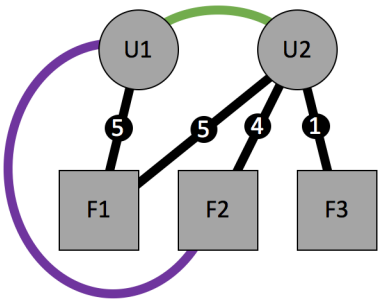

Figure 2.3: A user-user based approach where U1 and U2 are similar and F2 is recommended to U1.

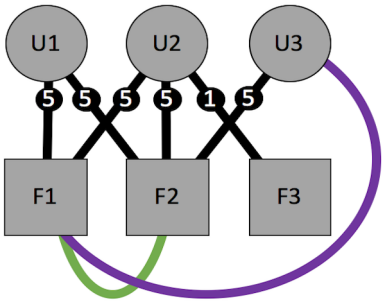

Figure 2.4: An item-item based approach where F1 and F2 are similar and F1 is recommended to U3.

The user-user based CF is illustrated in figure 2.3. Two users, U1 and U2, have rated a film (F1) equally, then they are considered similar. When U1 receives recommendations, it will be based on what U2 have rated as good films and vice versa. In this case F2. The item-item based CF is illustrated in figure 2.4. Two users (U1 and U2) have rated two films, F1 and F2, equally, then the films are considered similar. When user U3 wants to watch a film, a recommendation for F1 will be offered.

2.6 Artificial neural network

Artificial neural network (ANN) is a concept inspired by the human brain. Neural networks can be used to classify data, e.g. classifying a film into a rating based on its genre. As the name implies it is a network, with nodes and edges. The nodes are divided in three layers: input, hidden and output. The input layer sets the attributes for the item, for a film this would be what genre this film belongs to. The hidden layer is where the classification is performed, based on the information from the input layer. Finally, the output layer represents the different classes that an item can be classified into. Continuing with the film example, this is how likely it is to receive a rating based on the item and the current user. Figure 2.5 should make the overall structure clear.

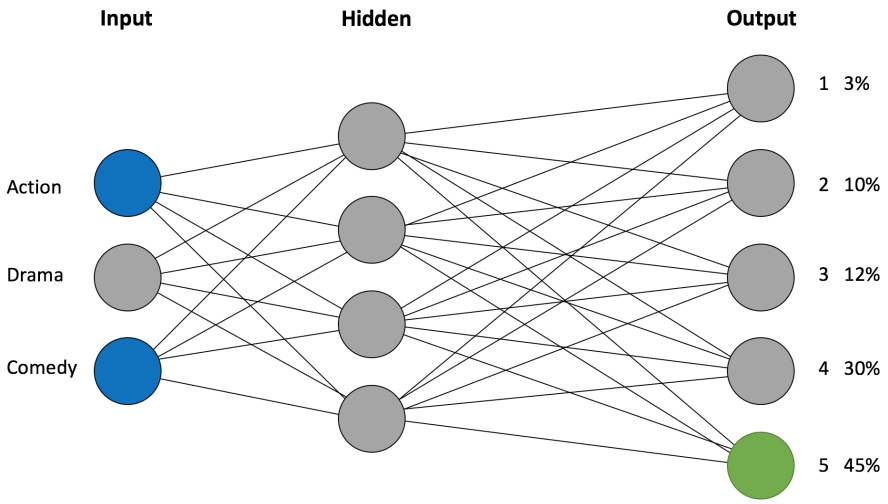

Figure 2.5: ANN where the input represents the genre of the film and the outputs are the predicted rating. The film being classified belongs to the action, comedy genre, the ANN predicts a 5-star rating with 45% probability.

The purpose of ANN is to optimise the parameters associated with the nodes and edges. Tweaking these parameters, bias and weight, is known as training (learning) the network. To be able to train the network, it is supplied with input values together with corresponding an-

swers. Using the values from the input and output layer together with a method called back-propagation bias and weight for the network is calculated. When the network is trained, it can be used to classify new items. When the attributes are inserted into the input layer, a method called forward propagation is started. It will calculate a parameter for the next node in the hidden layer. This parameter is scaled with the edges weight and then passed along to the nodes classifier which in turn calculates a new value, scaled with the nodes bias. This value is then passed along to the next layer until the it arrives at the output layer as the result[13].

One of the main issues with ANN is to optimise the training of the neural network as this task is very time consuming. However, with sufficient optimisation methods and lots of processing time ANN will generally have great performance[8].

2.7 Tree classifiers

2.7.1 Decision tree

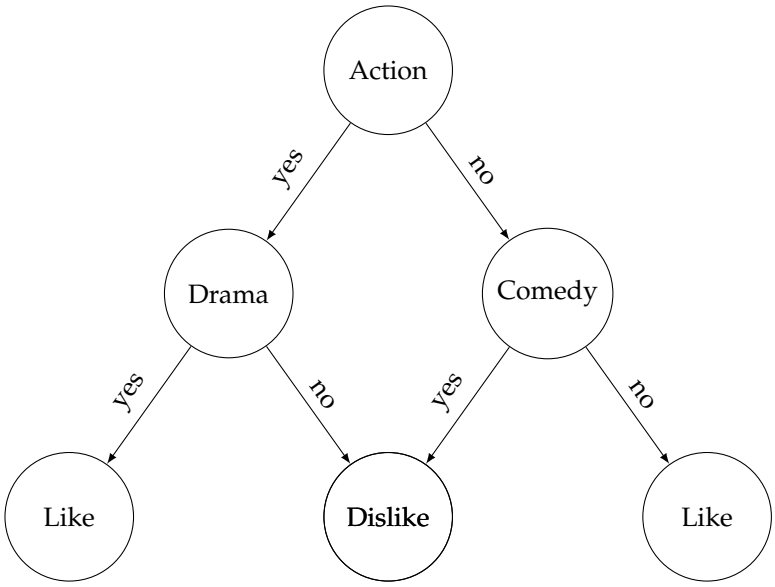

Figure 2.6: A decision tree recommending films based on the genre.

Behind a decision there is often a few questions that needs to have been answered before reaching a conclusion. For example, do I need this, do I have the money to spend, etc. This is the idea behind decision trees, classifying items based on their attributes, basically asking questions about the item. A decision tree has two types of nodes, decision and leaf nodes. Decision nodes are used to test attributes and branching one way or another. Leaf nodes will indicate what class an item belongs to. When training a decision tree it is a matter of splitting items on decision nodes in the best way possible, the better the split into two individual classes the better. When a node only contains items of the same class it means that the nodes is a leaf node, and the training is done for that item.

Consider the following example, the decision tree in figure 2.6 will classify an action-drama film as something the user will like. However, a comedy film will be classified as something that the user will dislike 2.6.

The advantage of decision trees is how fast they classify items and that is inexpensive to build a decision tree. As well as that they handle irrelevant attribute, which means that attributes that does not further improve the split when branching can be discarded[1].

A problem with decision trees is overfitting the training data when to many attributes are tested in the tree. Overfitting means that the model used for predictions are to complex i.e. must satisfy to many parameters which in turns make it incredibly sensitive to fluctuations and trains a classifier that cannot handle different data. This means that it will categories training data perfectly but when testing new data, it will classify badly[9].

2.7.2 Random forest

Random forest is a method that uses many decision trees to aggregate the answer. It tries to solve the drawbacks of regular decision trees. When classifying an item, it will create many different decision trees but each tree will have randomly selected attributes that they will work on. Thus, a random forest is going to consist of K different decision trees. When the random forest is going to classify an item, each of the K decision trees classifies the item and the class that is in majority is going to be the result of the random forest. The accuracy of the random forest depends on the precision of all individual trees and how diverse the trees are. If all the trees would have branched on the same attributes, the combination of them would not have been as useful as if the attributes are equally varied. *The Law of Large Numbers* says that if we perform an experiment many times the average result will converge to the expected value of the function. In [2] it is showed that because of this law, random forests do not overfit the training data when more trees are added.

2.8 Summary of previous work

In 2005, the article *A Hybrid Movie Recommender System Based on Neural Networks*[4] was published. This article presents a hybrid recommender system (henceforth CS-HRS) that use an artificial neural network for content-based filtering and k-nearest neighbours for collaborative filtering. It is a film recommender system and the data used in the article is from the MovieLens dataset and type and synopsis text are retrieved from `imdb.com`. The dataset is split 70%/30% into a training- and test-set.

The content-based filtering system creates three different neural networks for each user. The different content attributes investigated by these networks are: stars (actors, writers, etc.), kind (genre) and frequency of words from the synopsis text (the terms genre and actors will be used interchangeably for kind and stars). The synopsis analysis counts the occurrence of specific words in the synopsis. Common words that are of no interested are discarded, e.g. I, You, Me, Him, And, Or. All the networks have five outputs that corresponds to the rating scale of 0 to 5 stars. The result from the networks is stored in three matrices later to be used in the combination step.

For the collaborative filtering system, a k-nearest neighbour classifier is employed. Similarity between users is calculated with the Pearson formula. For all neighbours that has a similarity of more than 0.4 are considered alike to the specific user. However, if a neighbour receives a similarity value of -0.5 it is considered a dissimilar user. The neighbours that archive either one of these criteria are stored in separate lists, similar and dissimilar neighbours. Every film will have a negative and positive counter. For all similar neighbours the positive counter will be incremented if they liked the film and if they disliked it the negative counter will be incremented. The procedure is vice versa for the dissimilar neighbours.

The hybrid part of CS-HRS, is a set of seven rules that are used to combine the results from the CBF and CF system. The rules are described in appendix A. The goal of CS-HRS is to create a list of five films to recommend, this list is called the recommendation set. The recommendation set is a subset of the films from the test set that is used to evaluate the performance of CS-HRS.

The result of the CS-HRS presented in the article is that the system has a mean success rate per user of 80% and it fails to make recommendations to only 2.23% of the users. In the article, they argue that it is important to classify on more than one attribute in the content-based system to gain a recommendation rate over 60%[4].

Chapter 3

Method

3.1 Overview of approach

The aim of this study was to construct a hybrid recommender system and make experiments to achieve results comparable to CS-HRS. Therefore, a hybrid recommender system was developed using k-nearest neighbour for collaborative filtering and random forest for content-based filtering. To gain results comparable to CS-HRS the same dataset from MovieLens[11] was used to generate the results. The design of the RS is described in section 3.3 and is based on the problem statement in section 1.2.

Precision and recall was used to calculate the performance of the recommendations, this method was used in [4] to calculate the results of CS-HRS. To be able to make a relevant comparison these methods was therefore chosen.

3.2 Dataset

In [4] it is stated that CS-HRS used data from the MovieLens application, a project from research group GroupLens. In this thesis, the same dataset from 1998 is used to get comparable results. It contains 100 000 ratings, with a range of 1 to 5 and a step of 0.5. The ratings are based 943 users and there are 1682 films in the dataset. All users have rated at least 20 films. The dataset was released in April 1998[11].

3.3 TRÄD

The hybrid recommender system created for this study, TRÄD, was designed to be comparable to CS-HRS presented in [4]. Therefore, its goal is to recommend five films for every user and calculate the performance. A recommendation is considered successful if the user has rated the film ≥ 4 and unsuccessful otherwise.

When the term TRÄD is used in the rest of the thesis, it will mean the hybrid recommender system, when referring to its individual components (collaborative filtering or content-based filtering) it will be explicitly stated.

3.3.1 Collaborative filtering

The collaborative filtering classifier applied in TRÄD is a neighbourhood method. The dataset is split 0.7/0.3 into a training- and a test set. The training-set was used to build the rating (utility) matrix, mapping users, films and ratings. The system created two lists for all users. One list with similar users and another with dissimilar users. As discussed in section 2.5, the Pearson formula calculates correlation and this was used as the similarity measure. Dissimilar users were considered interesting as they have the opposite opinion about films, thus the inverse of their opinion can be used to make recommendations. The approach with a dissimilarity list is inspired from CS-HRS. To qualify for the similarity-list a user needs to have a correlation of at least 0.3 and for the dissimilarity-list a correlation lower than -0.3. These lists

are used to classify a film into the classes: recommend and not recommend. The classification is based on a score received from the similar and dissimilar users as described below.

A film has two counters, one for positive ratings and one for negative ratings. To allow a film to be recommended the ratio between the value of the positive counter and the total number of ratings (positive + negative) need to be larger than 0.5 as seen in equation 3.1.

$$\frac{positive}{ratings} > 0.5 \tag{3.1}$$

For the similar users, the following applies. Users that have rated the film with a rating between 4 and 5 will add 1 point to the positive counter. Users have rated the film between 0.5 and 3.5 will add 1 to the negative counter. On the other hand, for dissimilar user the following applies. Users that have rated the film with a 4 or a 5 will add 0.5 point to the negative counter. Users have rated the film between 0.5 and 3.5 will add 0.5 to the positive counter. It is worth noting, the opinion of dissimilar users' is worth half as the similar users. The reasons behind this that users tend to value the opinion of similar users more[4].

Finally, the films in the test data will be classified and the system calculates the performance. For every user, the system will calculate: number of correctly classified films, number of films that was recommended and the number of films that the user likes. The latter are used to calculate the precision and recall for all the users. The classification method described in section 2.2 was obviously not used. The reasoning behind this was to be able to compare our values with the article[4].

3.3.2 Content-based filtering

For content-based filtering in TRÄD the random forest classifier is used. The datasets of ratings and film genre are split 0.7/0.3 into a training- and a test set. The training-set was used to build two lists, one with ratings and one with film attributes. To train the classifier these two lists are used. The classifier was trained with 400 trees, each with a depth of four. These parameters were chosen based on the fact that more trees do not overfitt the training data as discussed in section 2.7.2 and higher numbers of these parameters did not increase the performance. All the films from the test set was classified by random forests based on the genre attribute found in the data from MovieLens[11]. From the resulting classification only films with rating ≥ 4 was kept.

For every user, the system will calculate: precision and recall, number of correctly classified films, number of films that was recommended and the number of films that the user likes. The latter are used to calculate the precision and recall for all the users. The reasoning behind this was to be able to compare our values with the article

3.3.3 Combination of recommendations

The combination system is built to recommend five films to a user. The following paragraph will explain how the selection is done.

The collaborative system and the content-based system produced two lists of film recommendations. The combination step begins with taking the intersection between these two lists, if there are more than five films we choose the ones with the highest similarity measures from the collaborative part. However, if the intersection result has less than five films, the next step in the combination rules is to consider the list from the collaborative system, adding the films with a similarity of more than 70%. The third step in the combination phase is to add films from the collaborative systems results as needed to reach five films. If the recommendation set still has not reached a size of five, films from the content-based filtering list will be added. To ensure that five films will be recommended, a failsafe last step will add films from a global list of top rated films until the recommendation set contains five films. The

global top list is based on the films that have received the most ratings by all users in the data-set.

3.4 Programming language and frameworks

The following tools where used to build TRÄD.

Python 3.6.0 with the following frameworks:
numpy 1.12.0 `http://www.numpy.org/`,
pandas 0.19.2 `http://pandas.pydata.org/`,
scikit-learn 0.18.1 `http://scikit-learn.org/stable/`

The program was run on two different, MacBook Pro with Intel Core i5 2.4 GHz, 16 GB 1600 MHz DDR3 RAM on macOS Sierra 10.12.3

3.5 Performance measurement

Precision and recall are used to measure the performance of TRÄD.

3.5.1 Precision

Precision is used to calculate the relevance of a result. Precision can be described as *the probability that a recommended film is relevant*.

$$Precision = \frac{|relevant\_films| \cap |recommended\_films|}{|recommended\_films|} \tag{3.2}$$

To calculate the precision, divide the number of hits (i.e. films that the user like) by the total number of recommended films[3].

3.5.2 Recall

Recall is another method to calculate relevance. Recall can be described as *the probability that a relevant film is recommended*.

$$Recall = \frac{|relevant\_films| \cap |recommended\_films|}{|relevant\_films|} \tag{3.3}$$

Recall is calculated by dividing the number of hits (i.e. films that the user like) by the numbers of relevant films[3].

Chapter 4

Results

4.1 TRÄD

This section presents the results from the collaborative, content-based and hybrid recommender system implemented in this study. All the results used the MovieLens dataset from 1998 and all 943 users where considered.

4.1.1 Collaborative system

The following results were obtained when recommending films to users with TRÄD's CF. Precision and recall is displayed in table 4.1. Table 4.2 contains the maximum and minimum value from the Pearson correlation coefficient obtained in this study. The result from table 4.2 shows that the inverse correlation between users are low.

Precisions (%)	**Recall (%)**
67	70

Table 4.1: Result obtained from TRÄD using only CF.

	Pearson correlation coefficient
Maximum value	0.73
Minimum value	-0.22

Table 4.2: Maximum and minimum value of Pearson correlation coefficient obtained in this study. Pearson correlation coefficient must be in the interval $[-1, 1]$.

4.1.2 Content-based system

The following results were obtained when recommending films to users with TRÄD's CBF, implemented with a random forest classifier. Precision and recall is displayed in table 4.3.

Precisions (%)	**Recall (%)**
65	76

Table 4.3: Result obtained from TRÄD using only CBF.

4.1.3 Hybrid

In this section, the results from TRÄD is laid out. Table 4.4 shows the precision for all users and numbers of users with 0% precision.

Overall percentage of successful recommendations	75%
Number of users with 0% precision	32
Percentage of users with 0% precision	3.39%

Table 4.4: Recommendation results from TRÄD

4.1.4 Comparison between classifiers

The box plots in this section display the precision for the different classifiers when recommending films for 942 users. The y-axis in the box plots represent the hit rate where 0 represent a hit rate of 0% and 1 represent a hit rate of 100%.

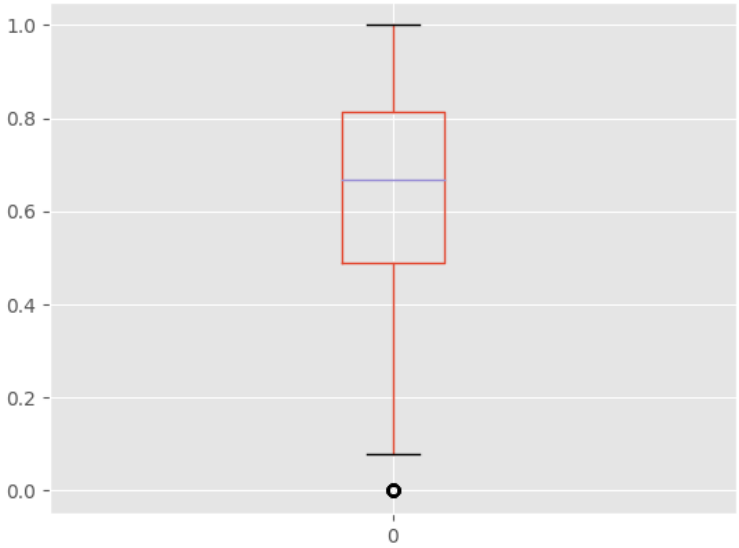

Figure 4.1: Precision collaborative filtering

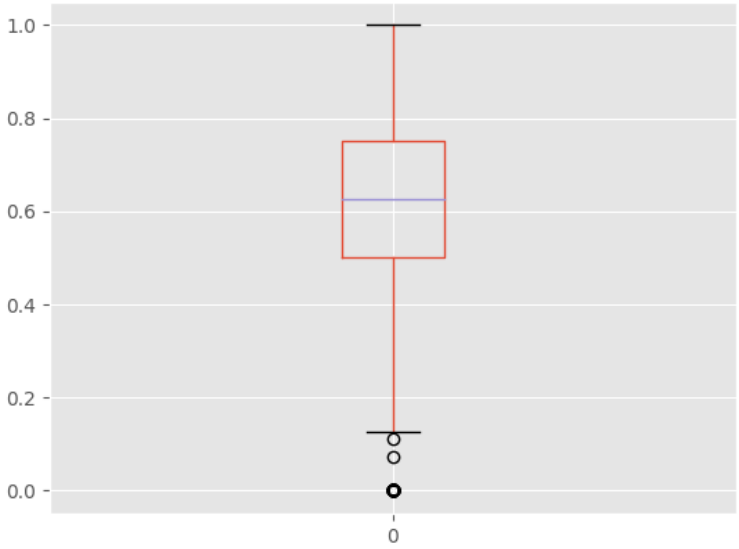

Figure 4.2: Precision content-based filtering

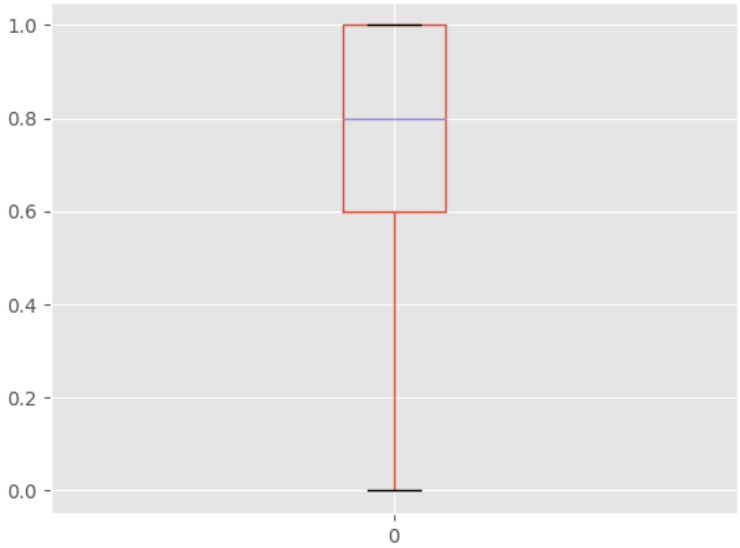

Figure 4.3: Precision hybrid filtering

The following result is a comparison of TRÄD's individual systems and the hybrid of how many users that receive 0% precision. The number of users with 0% precision significantly decrease when the hybrid system is used.

	Collaborative	**Content-based**	**Hybrid**
Users with 0% precision	91	68	36

Table 4.5: Number of users with 0% precision for all the different parts in TRÄD

The following result is the performance of a naive recommender system only using a global top list to make recommendations. The global top list contains the films that have received the most ratings from all users. This naive approach is clearly not accurate.

Overall percentage of successful recommendations	13%
Number of users with 0% precision	517
Percentage of users with 0% precision	54.9%

Table 4.6: The results in this table is from recommendations based on the highest rated films from all users in the training dataset. It

The result in table 4.6 shows a significant decrease of *overall percentage of successful recommendations* and significant increase of the *numbers of users with 0% precision* compared to table 4.4.

4.2 Results from CS-HRS

This section will present the result CS-HRS as presented in [4].

Table 4.7 contains results from the CBF in CS-HRS. The three first rows in table 4.7 is the result when only one specific attribute is used. The last row is the result when all the attributes are combined. The combined result is 11 to 13 percentage point better than the results with only one attribute.

	Precision (%)	**Recall (%)**
Kinds-based	61	62
Stars-based	59.3	62.5
Synopsis-based	59.4	65.6
Collaboration	72	78.5

Table 4.7: Results from CS-HRS's CBF. The first three rows are the result when only one attribute is used to classify films. The last row is the result when kind, stars and synopsis is combined when classifying films.

The results in table 4.8 can be compared with the results in table 4.4 and table 4.6.

Overall percentage of successful recommendations	82%
Number of users with 0% precision	21
Percentage of users with 0% precision	2.23%

Table 4.8: Recommendation results from CS-HRS[4] when both CF and CBF are combined.

4.3 Comparison between TRÄD and CS-HRS

This section summarises the comparison between the result from TRÄD and the result from CS-HRS as presented [4]. The difference is calculated using the difference in percentage points between the two results.

Table 4.9 is a comparison between table 4.4 and table 4.8, the last column is the difference between TRÄD and CS-HRS. The CS-HRS has better performance than TRÄD

	TRÄD	**CS-HRS**	**Difference**
Overall percentage of successful recommendations	75%	82%	7
Number of users with 0% precision	32	21	11
Percentage of users with 0% precision	3.39%	2.23%	1.16

Table 4.9: Comparison between TRÄD and CS-HRS when CF and CBF are combined. This is a comparison between table 4.4 and 4.8.

Table 4.10 show that random forest performance better than CS-HRS ANN when only considering film genre.

	Precision (%)		**Recall (%)**	
	TRÄD	**CS-HRS**	**TRÄD**	**CS-HRS**
Kinds-based	65	61	76	62

Table 4.10: Comparison between TRÄD and CS-HRS only considering CBF on kinds (film genre) attribute. TRÄD has better performance.

Chapter 5

Discussion

This thesis explores what performance effect would incur in a film HRS when designing the system in the spirit of simplicity. The results show that CS-HRS outperforms TRÄD. CS-HRS had 7 percentage points better performance as can be seen in section 4.3, table 4.9. It is interesting that the difference in performance is not more significant. Especially since in [4], it is argued that using few attributes in CBF would reduce performance significantly. Furthermore, it is clear from section 4.3 and table 4.10, that the choice of classifier is vital for the efficacy of the CBF system when the system only regards one attribute (in this case, film genre). Random forest gave results 4 percentage points greater in precision, 14 percentage points greater in recall than the ANN CBF classifier in CS-HRS. A possible reason why random forest achieved better results than ANN is that random forest is easier to configure and train. As mentioned in background section 2.6, ANN needs lots of processing time, data and a more complex problem (for example, image classification or speech recognition) to really take advantage of its potential. With more data such as, MovieLens' new 20 million dataset, a neural network approach likely would have received better results than the more rudimentary random forest classifier. However, the penalty of training the ANN would still be a potential bottleneck.

Comparing the results between the individual systems (CF, CBF) in TRÄD and the hybrid, circa 10 percentage point increase in precision is clear as shown in sections 4.1.1, 4.1.2 and 4.1.3. CS-HRS also achieved

10 percentage points increase, when comparing its content-based filtering system and hybrid system (Section: 4.2). As expected, TRÄD, is more efficient than its individual CF and CBF systems alone. The results presented in section 4.1.4 shows that the median value of successful recommendations is 0.8 (Figure 4.3), compared to 0.7 (Figure: 4.1) and 0.65 (Figure: 4.2) for CF and CBF, respectively. The results also show that the spread is smaller in TRÄD, 75% of the users have a successful recommendation rate between 0.6 and 1.0. However, it is worth noting that TRÄD introduces a lower quartile that includes 0%. In collaborative filtering and content-based filtering users with 0% precision are outliers (as seen in Figures: 4.1, 4.2). This is simply explained by that the distance to outliers will increase due to the fact the distance between the upper and lower quartile has increased. Nevertheless, comparing the number of users with 0% precision, table 4.5 in section 4.1.4, it is clear that TRÄD can recommend films to more users than either CF or CBF can alone. It is clear from these results that even TRÄD's simple design can achieve adequate performance.

It is noteworthy that TRÄD's overall percentage of successful recommendations increased with 10 percentage points. Even though, TRÄD only considers one attribute in CBF compared to CS-HRS that considers genre, actors and film-synopsis. Unsurprisingly, considering more attributes in the CBF will increase the overall performance. As shown by table 4.7 and 4.9 in sections 4.2 and 4.3, the combined CBF systems of CS-HRS has greater performance than genre based CBF of TRÄD. CS-HRS is also able predict recommendations to more users than TRÄD. However, using a simpler classifier like random forest is clearly feasibly as the difference in performance is not that huge. Thus, the main reason that TRÄD has worse performance than CS-HRS could be that classifying films only on genre is too general. Just because two films are in the same genre does not mean that a user will like both films. Therefore, combining different parameters (for example, genre and actors) would yield a more precise recommender system.

Hybrid recommender systems are used to overcome the problems of systems built solely on CF or CBF. We found that that the combination step is important to achieve a high percentage of successfully recommend films. The rules used in TRÄD was inspired by the rules used in CS-HRS, explained in section 3. However, CS-HRS's combi-

nation rules are more advanced due to the fact that it considers more attributes in its CBF and can combine the results in several ways. The final step in both CS-HRS and in TRÄD is to add films from a top-list of all films until the number of recommended films reach five. This is the crux of the problem, recommending films from the top-list is an inaccurate way to recommend films, shown clearly in table 4.6. As CS-HRS has more rules in its combination algorithm it is less likely to end up recommending films from the generic top list and thus CS-HRS will more likely recommend films that the users will like and receive a better success rate. This explains why CS-HRS has better performance and why it has fewer users with 0% precision (Section: 4.3).

Another important finding was the results in table 4.2, section 4.1.1. Surprisingly, the minimum negative correlation was found to be -0.22. Strangely enough, in [4] it is argued for a threshold of –0.5. Furthermore, in [4] it is described that CS-HRS used a list of users with a similarity coefficient equal or less than -0.5 to make predictions with (described in sections 2.8 and 3.3.1). However, this result show that neither CS-HRS nor TRÄD would consider this list of dissimilar users. No users were even close to have -0.5 in correlation. The reason that this threshold was offered in [4] could have been that CS-HRS used a different library for calculating PC. Nevertheless, a difference of 0.28 between TRÄD's and CS-HRS's PC calculations cannot be contributed only to different implementations of a standard correlation formula. Therefore, it begs to question whether the idea of dissimilar users was only discussed in theory and no research of the dataset was conducted. This is of course a reasonable approach when considering that a RS used in production should not be tailored to a specific dataset, as most services will update data constantly. On the other hand, our initial result show that users of film-services might not be that different to justify a threshold level of -0.5 which is why TRÄD used -0.3. However, more investigation of this issue is necessary. Ideally using a larger dataset to be able to draw a more general conclusion about users of film-services and whether the method of using a dissimilarity list yields any improvement or not.

For future research, we offer three areas that could be interesting to explore. First, it would be interesting to combine the attributes, genre, actors and synopsis in TRÄD's CBF and investigate what performance increase or penalty would incur. This would yield a more equal com-

parison with CS-HRS. Using this approach, only the change of classifier would differ. Secondly, it would also be interesting to run the experiment on a larger dataset. As mentioned earlier, MovieLens offers a new dataset of 20 million ratings. Finally, studying how RSs with different level of precision affects user experience. For example, it could be asked whether better theoretical performance is always desirable or if there are any user experience considerations that affects the performance unexpectedly.

Chapter 6

Conclusions

This study has explored whether building a recommender system with simplicity in mind is feasibly or not. The question we aimed to answer was what performance penalty or gain TRÄD would incur in comparison with a reference system. This study concludes that a penalty is incurred when switching CBF classifier to random forest and only considering film genre as item attribute. However, it is probably the removal of attributes from the CBF system rather than the switch of classifier that is the reason for the penalty. The underlying reasons for this conclusion are twofold. First of all, we have shown that random forest had better performance than artificial neural networks when only considering film genre. This means that the random forest classifier works better than CS-HRS artificial neural network classifier in this case. Secondly, one of the most vital design decisions of a hybrid recommender system is the combination rules. The number of step in the combination rules are in this case associated with the number of attributes used in content-based filtering. When considering more attributes several steps can be added to the combination rules which will prevent recommendations from the top list which we have shown to be inaccurate. Thus, using random forest as a classifier for CBF is an appropriate design decision.

We have also reinforced the argument that hybrid recommender systems boost the percentage of successful recommendation and when built correctly it will decrease the number of users that does not receive recommendations. However, precaution is advised when pon-

dering over whether to use the notion of dissimilar users or not in the CF. This thesis has shown that in this case, it was an unnecessary distraction and complication in TRÄD's CF system. Nonetheless, we concluded that more research into this topic is necessary to discard the dissimilarity list approach completely. Furthermore, we have shown that the penalty of using only one attribute in content basted filtering does not worsen the performance significantly.

In conclusion, a film recommender system designed with simple classifiers will yield reasonable performance and therefore, adhering to KISS is feasible when building a film recommender system.

Bibliography

[1] Xavier Amatriain et al. "Data Mining Methods for Recommender Systems". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, 2011, pp. 39–71. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_2. URL: http://dx.doi.org/10.1007/978-0-387-85820-3_2.

[2] Leo Breiman. "Random Forests". In: *Machine Learning* 45.1 (2001), pp. 1–33. ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. URL: http://dx.doi.org/10.1023/A:1010933404324.

[3] Michael Buckland and Fredric Gey. "The relationship between Recall and Precision". In: *Journal of the American Society for Information Science* 45.1 (1994), pp. 12–19. ISSN: 1097-4571. DOI: 10.1002/(SICI)1097-4571(199401)45:1<12::AID-ASI2>3.0.CO;2-L. URL: http://dx.doi.org/10.1002/(SICI)1097-4571(199401)45:1%3C12::AID-ASI2%3E3.0.CO;2-L.

[4] C. Christakou and A. Stafylopatis. "A hybrid movie recommender system based on neural networks". In: *5th International Conference on Intelligent Systems Design and Applications (ISDA'05)*. Sept. 2005, pp. 500–505. DOI: 10.1109/ISDA.2005.9.

[5] Christian Desrosiers and George Karypis. "A Comprehensive Survey of Neighborhood-based Recommendation Methods". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, 2011, pp. 107–144. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_4. URL: http://dx.doi.org/10.1007/978-0-387-85820-3_4.

[6] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. "Collaborative Filtering Recommender Systems". In: *Found. Trends Hum.-Comput. Interact.* 4.2 (Feb. 2011), pp. 81–173. ISSN: 1551-3955. DOI: 10.1561/1100000009. URL: http://dx.doi.org/10.1561/1100000009.

[7] A. Felfernig, G. Friedrich, and L. Schmidt-Thieme. "Guest Editors' Introduction: Recommender Systems". In: *IEEE Intelligent Systems* 22.3 (May 2007), pp. 18–21. ISSN: 1541-1672. DOI: 10.1109/MIS.2007.52.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[9] Douglas M. Hawkins. "The Problem of Overfitting". In: *Journal of Chemical Information and Computer Sciences* 44.1 (2004). PMID: 14741005, pp. 1–12. DOI: 10.1021/ci0342472. eprint: http://dx.doi.org/10.1021/ci0342472. URL: http://dx.doi.org/10.1021/ci0342472.

[10] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. "Content-based Recommender Systems: State of the Art and Trends". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, 2011, pp. 73–105. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_3. URL: http://dx.doi.org/10.1007/978-0-387-85820-3_3.

[11] MovieLens. *MovieLens*. 2017. URL: https://grouplens.org/datasets/movielens/ (visited on 02/16/2017).

[12] Anand Rajaraman and Jeffrey David Ullman. *Mining of Massive Datasets*. New York, NY, USA: Cambridge University Press, 2011. ISBN: 9781107015357.

[13] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Recommender Systems Handbook". In: *Recommender Systems Handbook*. Ed. by Francesco Ricci et al. Boston, MA: Springer US, 2011. ISBN: 978-0-387-85820-3. DOI: 10.1007/978-0-387-85820-3_1. URL: http://dx.doi.org/10.1007/978-0-387-85820-3_1.

[14] Loren Terveen and Will Hill. "Beyond Recommender Systems: Helping People Help Each Other". In: *HCI in the New Millennium* (2001), pp. 487–509.

[15] Wikipedia. *Netflix Prize — Wikipedia, The Free Encyclopedia*. [Online; accessed 16-February-2017]. 2017. URL: %5Curl%7Bhttps://en.wikipedia.org/w/index.php?title=Netflix_Prize&oldid=761558843%7D.

Appendix A

Combination rules

Rules used in CS-HRS[4] combination step:

1. Initially, select films that are proposed by all four individual criteria – kind, contributors, summary (content criteria) and opinion of other users (collaborative criterion).
2. Add the films that satisfy the collaborative criterion using a high threshold.
3. Add the films that are proposed by all the content criteria (especially in the case of new movies, for which no evaluations are available).
4. Add the films that are proposed by exactly two content criteria plus the collaborative criterion.
5. Add the films that are proposed by exactly two content criteria.
6. Add the films that are proposed by one content criterion.
7. Finally, if movies are still missing from the recommendation set, propose the most popular films.

Appendix B

Code repository

The code for TRÄD is located here

`https://github.com/AndreasBrommund/TRAD`