Movie Recommendation

Semester Thesis

Cyril Arnould

carnould@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Barbara Keller, Michael König
Prof. Dr. Roger Wattenhofer

February 11, 2014

Acknowledgements

This is a continuation of Markus Frei's semester thesis "Find your Personal Movie Critic", whom I want to thank for his tremendous help and the access to his crawled Rotten Tomatoes data. I also want to thank my supervisors Barbara Keller and Michael König for their flexibility and all their inputs on the project, as well as Tobias Langner for letting me access the IMDb data he crawled. Lastly, I want to thank everyone who participated in testing the algorithms that were the topic of this thesis.

Abstract

This thesis comprises the development of two fundamentally different algorithms and their slight variations to recommend movies. The algorithms are to first gather useful information about the user and then to predict how the user would rate movies. They make use of an extensive database of reviews and other movie-information, which has been improved to suit the needs of the algorithms. A method to test their performance has been developed as well, which is used to determine suitable parameters for the algorithms and evaluate them.

Contents

Chapter 1
Introduction

Nowadays, people have a lot of ways to get movie recommendations. Many of these either require intimate knowledge of the user's taste in movies, or give an unpersonal recommendation based on the popularity of the movie. Often, they only highlight current top movies, while older ones are ignored. Finding possibly interesting movies without recommendations can be frustrating as well: There can be a high number of reviews per movie, both praise and criticism. Searching through all of them requires time and knowledge on which critics to trust. And in the end, one might end up knowing everything about the movie without even watching it.

The main goal of this semester thesis is to develop different algorithms that do not require a lot of input but still give satisfying results. In order to determine whether they do, it is necessary to test the algorithms on their quality. To create the same conditions for each algorithm, we chose to let an algorithm query the user whatever information needed, so long as the information does not exceed a predetermined number of bits. This number of bits some specific information takes is the binary logarithm of the number of possiblities. The algorithms are designed to search for recommendations and predict how the user would rate these. For comparison, each algorithm predicts the ratings of movies the user already knows, and the closest predictions determine the best algorithm.

Another way to set equal initial conditions besides an equal number of bits is to feed each algorithm the same amount of user-rated movies as the only source of information. This either restricts the possibilities of how an algorithm can be implemented, or makes it more complicated as the algorithm needs to infer any other information about the user's preferences from the ratings if it is not allowed to simply ask for them. Therefore, the approach with a predetermined number of bits is employed.

The algorithms and the testing environment were developed in Java[2].

Chapter 2

Data Base

The data this thesis is built on was readily available prior to the beginning as an SQLite[1] database, it has been crawled from the website of the movie review aggregator Rotten Tomatoes[3]. Rotten Tomatoes has a set of requirements for reviews to be accepted, they need to come from acclaimed critics with a broad audience and a strong position in the business. The criteria can be met in either print media, broadcast or online publication[4]. It makes for a reliable source of information.

2.1 Overview of the Data

The data base mainly consists of the collected reviews, which individually consist of a rating from the critic, a 'critic tomato' which indicates independently of the rating whether the critic recommends the movie or not and a short written extract from the review itself. The rating is linearly mapped from the scale the critic used to a scale of 0 to 100.

In addition to the reviews, other information was collected as well for each movie: cast, directors, genres, writers, film length, release date in theaters and/or of the DVD, the box office income, the production company, a short written description of the movie, the tomatometer (which indicates the percentage of critics that recommend this movie) and the percentage of the audience (i.e. the users of Rotten Tomatoes) that recommends this movie.

2.2 Changes for this Thesis

One piece of information that has not been crawled is whether a movie is a sequel to another movie or not. This information is important, otherwise the algorithms will ask the user to rate every movie of a series without gaining much additional information. This tends to happen as many sequels have a lot in common with their predecessor and will likely be picked again.

The sequels therefore need to be filtered, which is done based on the movies' names. Each identified pair of movie and sequel is stored in a separate table in the database. Detailed information on how a title needs to be formatted to be recognized as a sequel can be found in appendix B. Certain titles have to be added manually because it is impossible to determine from the name whether they are a sequel or not. This is done for rather popular movies that have a high probability of being picked by one of the algorithms, an example of such titles are "Batman Begins" and "The Dark Knight".

The Rotten Tomatoes data also contains television series, which are deleted from the database using different variations of keywords like 'season', 'series', 'episodes' and 'collection'.

Chapter 3

Algorithms

Two algorithms and their variants are implemented as part of this thesis, each of them are separated into an input and an output part. An input algorithm gathers information about the user, and the output algorithm then recommends movies based on the input's information. This makes it possible to let the output algorithm run with different configurations using the same input.

The input part of each algorithm can ask the user about any information needed at the very beginning. After that, it always first selects a movie to be rated, presents it to the user and then stores the rating in the database. It then also calculates and stores any other information that needs to be inferred from the rating of the movie.

An output algorithm selects suitable movie candidates for recommendations (a maximum of two thousand in order to keep the calculation time from spiking) and then rates these. With each rating, the algorithm adds a weight to the rating that specifies how much information is backing up this rating. The movies with the highest product of weight and rating are then recommended.

3.1 Critic Algorithm

The critic algorithm is loosely based on Markus Frei's solution in his thesis[5]. His algorithm compares the user's ratings with each critic in the database, finds 10 critics whose ratings overall come closest to the user's and then recommends movies based on the top-rated movies by these critics. Simultaneously, with each movie the user rates, the genres the movie belongs to get rated as well, enabling the genres to influence the recommendations.

In the implementation in this thesis, the algorithm considers not only critics with matching tastes, but critics with tastes opposite to the user as well. Each critic gets rated based on how well or how badly his ratings reflect the user's ratings, respectively. The rating of a critic is on a scale of 0 to 100, 100 meaning his movie ratings matches the user's ratings every time; 0 meaning he rates the exact opposite of the user every time.

A critic can not be regarded as sufficiently rated if only one of his ratings was compared to the users', so the number of compared movies is used to determine a weight for the critic. In order for critics not to receive too much weight if they have a lot of ratings, a weighting limit is introduced. For each critic, a 'positive weight' and a 'negative weight' gets calculated as in Equations 3.1 and 3.2 below, the positive weight being higher the more his rating reflects the user's, the negative weight being higher the more his ratings are opposite of the user.

$$PW = \begin{cases} \frac{\sum_i 100-|RC_i-UR_i|}{n} \cdot \sqrt{n}, & \text{if } n < WL. \\ \frac{\sum_i 100-|RC_i-UR_i|}{n} \cdot \sqrt{WL}, & \text{otherwise.} \end{cases} \tag{3.1}$$

$$NW = \begin{cases} (100 - \frac{\sum_i 100-|RC_i-UR_i|}{n}) \cdot \sqrt{n}, & \text{if } n < WL. \\ (100 - \frac{\sum_i 100-|RC_i-UR_i|}{n}) \cdot \sqrt{WL}, & \text{otherwise.} \end{cases} \tag{3.2}$$

PW/NW	Positive/Negative weight
n	Number of movies the critic and user have in common
RC	Critic's rating of the movie i
UR	User's rating of the movie i
WL	Weighting limit

Following this idea, a lot of parameters can influence the outcome of the algorithm. It is possible to change the number of critics that are used, the weighting limit, how many critics with matching taste vs how many critics with opposite taste are used and how much influence the combined rating of the critics with opposite taste on a movie's rating has.

3.1.1 Implementation

The input part of this algorithm is designed to find critics with high positive/negative weights as fast as possible, while it still looks for critics which might have even better ratings (but are not rated yet). When the algorithm determines what movie the user has to rate, it takes a certain number of critics that have less than 'weighting limit' ratings and the currently highest positive/negative weights and finds the movie that is rated by the highest number of these critics.

When it comes to recommendations, the algorithm selects the movies of its currently highest weighted critics. The movie candidates are sorted descendingly by the number of reviews they have, and the first 2000 movies get evaluated.

The final rating and its weight for each movie is calculated as in one of the following three cases:

1. The movie has only been rated by critics which match the user's taste.

$$MR_P = \frac{\sum_i RC_i \cdot CPW_i}{\sum_i CPW_i}$$

$$MW_P = \frac{\sum_i \min CN_i, WL}{\sqrt{\max N_P, WL}}$$

2. The movie has only been rated by critics which do not match the user's taste.

$$MR_N = (100 - \frac{\sum_i RC_i \cdot CNW_i}{\sum_i CNW_i})$$

$$MW_N = \frac{\sum_i \min{(CN_i, WL)}}{\sqrt{\max{(N_N, WL)}}}$$

3. The movie has been rated by both critics which match the user's taste and by critics with opposite taste. In this case, the movie's rating is a combination of the two:

$$MR_T = MR_P \cdot (1 - BCI) + MR_N \cdot BCI$$

$$MW_T = \frac{\sum_i \min{(CN_i, WL)}}{\sqrt{\max{(N_N + N_P, WL)}}}$$

MR	Movie's rating
MW	Weight of the rating
RC	Critic i's rating of the movie
CNW/CPW	Critic i's 'Negative/Postive Weight'
CN	Number of ratings the critic i has
WL	Weighting limit
N_N/N_P	Total number of critics with opposite/matching taste
BCI	Bad critic importance; sets the ratio between MR_P and MR_N

In summary, the higher a critic's positive/negative weight is, the more influence his rating of the movie has on the total rating of the movie. And the more critics have rated a movie/the higher weighted the critics are, the higher is the weight of the total rating.

3.1.2 Different Versions

Markus Frei's idea also accounts for a movie's genres, so it is incorporated in this algorithm as well. There's a lot of room for how it should go about doing so, therefore the algorithm is implemented in three different versions.

The first version lets the user eliminate certain genres, movies belonging to any of these can not be selected anymore by the algorithm. The second version ignores the genres, its advantage being that it is able to let the user rate more movies before it needs to make recommendations, since more of the information-bits are available. The first implementation loses one bit of information per genre that it needs to ask. The third version asks the user for his three favorite genres, and then only selects movies that belong to at least one of those genres.

The number three for the favorite genres is chosen so that the number of eligible movies shrinks dramatically, but still allows for some variety. Again, the third version loses one bit of information per genre it needs to ask.

3.2 Criterion Algorithm

The criterion algorithm takes the idea of using additional information and recommends movies using multicriteria optimization. The movies get rated according to how well they fulfill the following fourteen criteria, which are assigned to one of three categories:

Category 1: Depends on user's taste, several per movie.

> **Criteria:** Genre, Director, Writer, Actor.

Category 2: Depends on user's taste, one per movie.

> **Criteria:** Release year (either in theaters or on DVD, whichever comes first), Length, Production Company.

Category 3: Higher score is better.

> **Criteria:** Audience Rating, Tomatometer, Story, Soundtrack, Imagery, Acting, Impact.

3.2.1 Implementation

These criteria might not be equally important to different users. Therefore, the first thing the algorithm does is asking the user to rate the criteria. After that, the input algorithm finds as much information as possible about the criteria most important to the user, detailed information can be found in subsection 3.2.2.

The output algorithm then selects (a maximum of 2000) movies that perform well in the (to the user) most important criteria. This means that the algorithm selects movies of the currently top-rated factors in the criteria of category 1 (and of the top-rated critics). The rating of the criterion determines how many movies of the maximum of 2000 are selected from that criterion. If the criteria of category 2 (plus audience rating/tomatometer) have a rating of higher than half the scale, the algorithm only allows movies that supply this information, seen as some movies don't supply every information needed. Finally, the algorithm rates these movies according to the user's preferences in all other criteria, meaning the movie will receive a rating in each criterion between 0 and 100.

3.2.2 Different Versions

As with the critic algorithm, three different versions of this one are implemented. The first version lets the user order the criteria in descending order of preference, so each criterion will get a distinct rating between 1 and 14. The number of bits this takes is $\log_2(14!) \approx 36.34$.

When its input algorithm searches for movies, it orders all eligible movies according to how much information they can provide for each criterion of category 1.

Example 3.1. Say a user rates the criterion 'director' higher than 'actor', and there's three eligible movies: one with 2 directors and 10 actors, one with 2 directors and 17 actors and another with 1 director and 13 actors. This algorithm would then first pick the movie with 2 directors and 17 actors, then the one with 2 directors and 10 actors and then the one with 1 director and 13 actors.

The release year and length matter as well in the picking process. The algorithm orders the movies in the same manner as with the criteria from category 1, but according to how far away their lengths/release years are from the currently already rated lengths/release years. In the unlikely event that all the eligible release years/lengths already have been rated, the one with the least ratings will come first.

The second version is very similar to the first, it lets the user order the criteria in the same manner, but then searches differently for movies. It orders the movies not by one criterion after the other, but how much information they provide overall. The information score each movie gets is calculated as a weighted sum of how much information they provide per criterion on a scale of 0 to 100, using the rating of the criterion as the weights.

Example 3.2. Given that the eligible movies have 3, 2, 1 or 0 listed directors, the information ratings in the 'director' criterion would be 100 for movies with three directors, 66 for movies with two directors, 33 for movies with one director and 0 for movies with 0 directors.

For the information score of the release year and length, the currently furthest (or least rated) are determined and dubbed ‘best year/length’. If no years/lengths have been rated yet, the best year is determined at random. The eligible movies are then rated according to how far their release year/length is from the best year/length, as is shown in Equation 3.3. In comparison to the other criteria, the year/length uses the distance squared. The closest eligible year will get a score of 100, the furthest eligible one a score of 0.

$$YR = 100 - 100 \cdot \frac{(RY - BY)^2 - SCY}{SFY - SCY} \tag{3.3}$$

$$SCY = (CY - BY)^2$$

$$SFY = (FY - BY)^2$$

YR	Year’s rating
RY	Release year of the currently rated movie
BY	Best year
SCY/SFY	The score of the closest/furthest year
CY/FY	Closest/Furthest eligible year

The third version uses the same mechanism to find movies as the second, but it doesn’t make the user order the criteria at the beginning. Instead, they have to rate each independently on a scale of 0 to 3 or 4 (further explanation of the rating scale can be found in chapter 4). In this version, it is possible to entirely exclude criteria from the algorithm by rating them 0. The number of bits this query takes is $14 \cdot \log_2(\text{rating scale}) \approx 22.18$ or 28.

All three versions narrow the selection of movies further down by excluding the currently most rated genre/director/writer/actor/production company in order to give others a chance to get rated.

3.2.3 Rateable Criteria

Since certain criteria heavily depend on the user’s taste, the input algorithm needs to find that information. Whenever the user rates a movie, each of the movie’s factors from category 1 and 2 get rated as well. The algorithm stores an average rating for every genre, director, etc.; and as with the critics, a weight and the number of ratings is stored alongside the rating. Again, a weighting limit is used when calculating the factor’s weight to prevent often-rated factors from gaining the upper hand.

The rating that the output algorithm calculates in these criteria is the average rating of all the factors, for example the average rating of all the participating actors. The weight of this rating is the average of $\sqrt{\min(\#\text{ of ratings, weighting limit})}$

of all the actors. If one of the factors has not been rated yet, it will not influence the rating, but lower the weight. If the length or the release year of the movie has not been rated yet, a rating/weight gets calculated for the missing year/length using linear interpolation.

3.2.4 Keywords in Critic Reviews

The criteria in category 3 (apart from audience rating and critic tomato) are determined from the critic reviews. Each review has been checked for a set of keywords, a list of which can be found in appendix A. The handling of these criteria will be explained by the example of the criterion ‘story’.

If a review contains a keyword from ‘story’ and the review is positive (critic tomato=1), the review is marked as +1 for the criterion ‘story’. If the review is on the other hand negative (critic tomato=0), it is marked as -1 for ‘story’. However, people don’t always agree on what makes a good story. Therefore, the critics get rated on their abilities to rate movies. For each movie the critic has rated like the user (only regarding whether the rating is positive or negative), the critic will get +1 point, otherwise -1.

The rating and weight for the keyword criteria are explained in Equations 3.4 and 3.5. If a critic has no rating, his number of points is set to one, because it is assumed that all the critics are capable of expressing their opinion of a movie to some degree. Critics with a positive number of points get more influence over the rating, while critics with negative points get a lower influence than non-rated critics. And the more critics have a positive numbers of points, the higher the total weight of the rating goes.

$$CR = \begin{cases} \frac{\sum_i KR_i \cdot \sqrt{\min(CP_i+1, WL)}}{\sum_i \sqrt{\min(CP_i+1, WL)}}, & \text{if } CP_i \geq 0. \\ \frac{\sum_i KR_i \cdot \sqrt{|1/CP_i|}}{\sum_i \sqrt{|1/CP_i|}}, & \text{otherwise} \end{cases} \tag{3.4}$$

$$CW = \begin{cases} \frac{\sum_i \sqrt{\min(CP_i+1, WL)}}{n}, & \text{if } CP_i \geq 0. \\ \frac{\sum_i \sqrt{|1/CP_i|}}{n}, & \text{otherwise} \end{cases} \tag{3.5}$$

CR	Criterion’s rating
CW	Weight of the rating
KR	Keyword rating from critic i (either 0 if negative, or 100 if positive)
CP	Critic i’s number of points
WL	Weighting limit
n	Total number of reviews that contained a keyword from that criterion

Another idea to determine whether a keyword is used in a positive or negative way is to search for secondary keywords describing the primary keywords from appendix A. However, many ratings use rather positive terms like ‘good’ to express a negative attitude towards the criterion. A few examples that were found when searching for ‘good story’:

- “[...]but what Spottiswoode and company do to that good story is unforgivable.”
- “[...]takes a perfectly good story and kills it.”
- “[...]is a good story gone mediocre.”

Only depending on the critic tomato is unfortunately not perfect either, for instance if a critic praises the soundtrack but finds fault with the story and as a result has disliked the movie, both soundtrack and story will receive negative remarks. But most of the time, critics usually focus on either why they like the movie or why they dislike it, and as a result, correctly found ratings overshadow the incorrect ones.

3.2.5 Final Rating

The final rating (and final weight) of the movie is a normalized weighted sum, using the calculated ratings of each criterion and the user’s rating of the criteria as weights.

The nature of the algorithm usually does not allow for ratings to reach 100, since any missing information or any rating below 100 concerning any factor will result in a lowered final rating. Therefore, the algorithm takes its highest calculated rating as a norm and scales all the ratings to 100. An exception to this rule must be made if the algorithm is supposed to only rate one movie or a rating of 100 is not expected.

Chapter 4

Testing the Algorithms

The algorithms depend on a universal rating scale on which the user needs to rate movies (and the criteria in the third version of the criterion algorithm). Rating scales of 0 to 3 and 0 to 4 are chosen for testing, to compare scales that allow for a neutral rating and scales that don't. On the scale of 0 to 3, the user has to decide whether he likes the movie or not, whereas on a scale of 0 to 4, he can just rate it as two out of four if he is indecisive.

In order to keep the number of test runs low, the rating scale is chosen randomly for each user to be either three or four at the beginning of the test.

4.1 Test Procedure

For each algorithm, arrays of parameter configurations are fed to the testing environment. For each configuration, the input algorithm first gets called with 50 bits of information to be gathered. Then, the output part recommends 10 movies that get rated by the user. The input algorithm gets called again, this time with 20 bits to make a total of 70 bits asked, and then the output recommends another ten movies.

After the first version has been called in its first parameter configuration, the next algorithm version takes its turn. The users are given a chance to finish the testing before the next parameter configurations get called. When each parameter configuration has been run (or the user has opted out), the test gets evaluated. Each output algorithm calculates a rating for all the movies that are rated by the user but have not been asked by its corresponding input algorithm.

For each of the tests (i.e. the recommendations after 50/70 bits and the calculated ratings for movies in other configurations, named 'other movies'), the following results get calculated/stored:

1. The average distance between the user's rating and the algorithm's prediction in this test, the variance of that distance and the number of predictions used to calculate this.

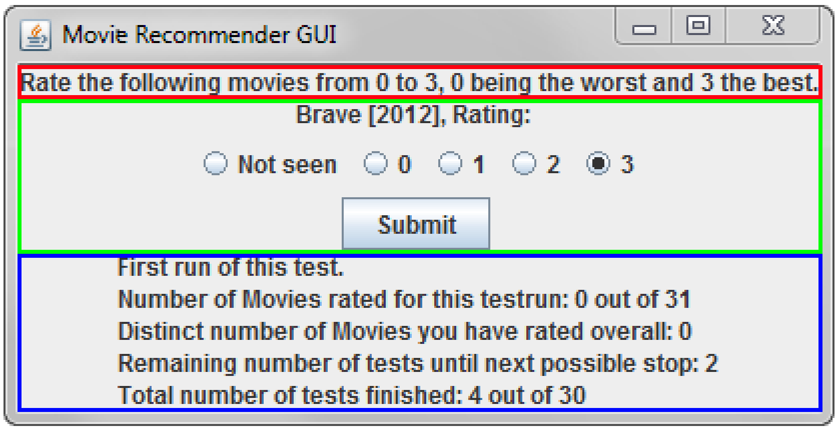
2. The number of predictions that are closer/farther than the average distance, and the variance of only their distances.

3. The average distance of only the movies the user has rated positively/negatively (2/3 or higher, 2/4 or higher respectively 1/3 or lower, 1/4 or lower), the variance of that distance and the number of movies that were rated positively/negatively.

At the very end, the testing environment also calculates these results for an algorithm that always predicts the same rating (for every rating out of the scale), for an algorithm that uses a random prediction between 0 and 100, and for algorithms that always use the audience rating respectively the tomatometer as a prediction.

4.2 User Interface

The user interface for the testing environment is composed of Java Swing[2] elements. For the initial queries of the individual algorithms, screenshots can be found in appendix C. A screenshot of the interface to rate movies can be found in Figure 4.1. It is separated into three parts:


Movie Recommender GUI

Rate the following movies from 0 to 3, 0 being the worst and 3 the best.

Brave [2012], Rating:

Not seen [ ] 0 [ ] 1 [ ] 2 [ ] [✓ 3]

First run of this test.
Number of Movies rated for this testrun: 0 out of 31
Distinct number of Movies you have rated overall: 0
Remaining number of tests until next possible stop: 2
Total number of tests finished: 4 out of 30

Figure 4.1: User interface when rating movies.

The instruction panel (red/top), which is also used to display error information, when the user clicks on the submit button without having selected a rating for example.

The information/input panel (green/middle), which displays a loading message when no movie can be rated at the moment. Alternatively, if an algorithm has selected a movie that has already been rated by the user, the algorithm displays how many already rated movies have been re-used so far.

The progress (blue/bottom) is shown during the entire movie rating phase of the test, and displays in detailed fashion how many movies are to be rated in the current test and how many tests are left.

4.3 Choice of Parameters

In order to keep the main test relatively short, preliminary tests have been evaluated to determine which parameters are suitable for testing.

4.3.1 Critic Algorithm

There are four parameters that can be changed in the critic algorithm: The weighting limit, the number of critics, the distribution of the critics (stored as the percentage of the number of critics that have matching taste) and the influence of the critics with opposite taste on the final rating of a movie, named 'bad critic importance', abbreviated with 'BCI'. They have been chosen for all three versions of the algorithm as follows:

Weighting limit: 6
Regardless of how the other parameters are configured, the average distance has a local minimum at six in the preliminary test results.

Number of critics: 20, 30
Early tests show that low numbers of critics (< 10) lead to very unsatisfying results (with average distances above 40 in 'other movies'). The results with 20 and 30 are both satisfying, but too close to each other to make a distinction which is better, therefore both are in the actual test.

Critic distribution: 0, 0.25, 0.5, 0.75
Interestingly, setting 100% of the number of critics to have matching tastes leads to consistently very bad results (average distances ranging from 40 to 60 in 'other movies'), which is why it is exempt from the main test. The results for the critic distribution set to 0.75 shows equally bad results, but has been kept because of rather good results in the recommendations after 50/70 bit inputs. All combinations of number of critics and critic distribution are tested except 20/0.5 and 30/0.25.

Bad critic importance: 60 (50, 70)
As with the weighting limit, the average distance has a local minimum when set to 60, meaning 60% of the final rating of a movie comes from the critics with opposite taste (if any of them have rated the movie). Since the bad critic importance only influences the output, the tests with $BCI = 50\%$ or $BCI = 70\%$ make use of the same input algorithm that used $BCI = 60\%$ and are evaluated for 70 bits and 'other movies'.

4.3.2 Criterion Algorithm

The only parameter that can be changed when it comes to the criterion algorithm is the weighting limit. Very early tests show that weighting limits around 10 yield bad results when recommending movies, so only weighting limits below 10 were tested. The results of these tests can be found in Figure 4.2.

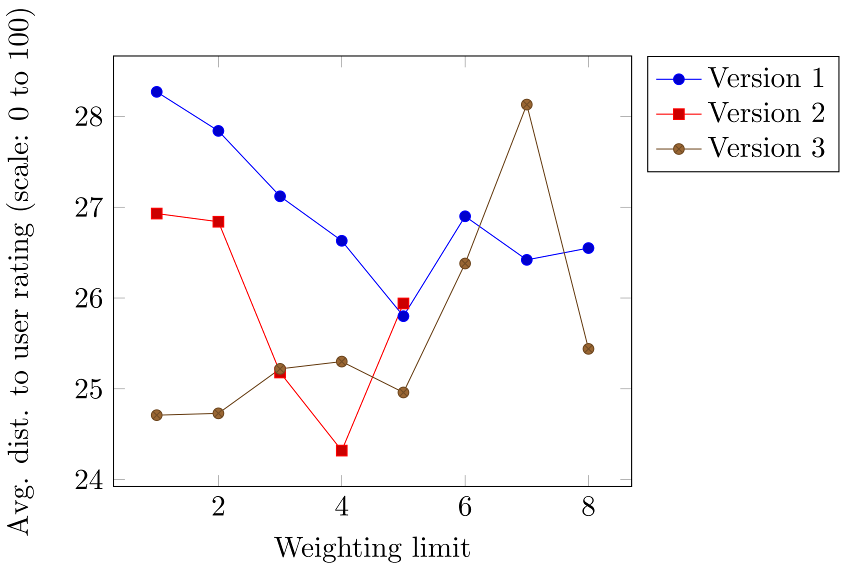


Figure 4.2: Evaluation of ‘Other movies’ for the criterion algorithm

From the tested weighting limits, the ones chosen for the main test (in the order they’re being tested) are:

Version 1: 5, 4, 6, 1

Version 2: 4, 3, 5, 1

Version 3: 5, 4, 6, 1

The first choices for each version are the ones with the best results, followed by their two neighbours. Weighting limit 1 has been added because the results for recommendations after 50/70 bits are impressively close to the user’s rating.

The curve of version 3 is a bit misleading, since not all participants of the preliminary test have tested weighting limits 1 through 3, and the results of these participants were generally worse for version 3 than those of the participants that did test weighting limits 1 through 3.

Chapter 5

Results and Evaluation

A total of 11 people have run the test, 6 of which have done the whole test, 5 of which have done the first set of parameters to be tested.

Each version's results are first presented and evaluated on their own, then against the other versions of the same algorithm. At the end, the two algorithms as a whole are compared to one another.

Since the algorithms only recommend 10 movies, of which the users might not know all, the average of all the users' results of the same testrun is normed to the number of ratings. For example, if A knew 4 of the recommendations and B 7, A's results account for $\frac{4}{11}$ of the final average distance of the testrun and B's results $\frac{7}{11}$.

5.1 Criterion Algorithm

The movies the criterion algorithm recommends are usually the same going from 50 to 70 bits, which makes it ideal to check if its ratings are improving with more information.

5.1.1 Criterion Algorithm Version 1

The results for the recommendations after 50/70 bits and for 'other movies' are plotted in Figure 5.1.

For rating scale 3, the plots for 50/70 bits show how the algorithm improves the recommendations upon being fed more information. For rating scale 4, however, the exact opposite is the case, the algorithm's recommendations are worse with 70 bits of information.

A closer look at the data reveals that with rating scale 3, the users usually knew less of the movies that are recommended after 70 bits than after 50 bits. For rating scale 4, the users knew overall more of the recommendations with 70

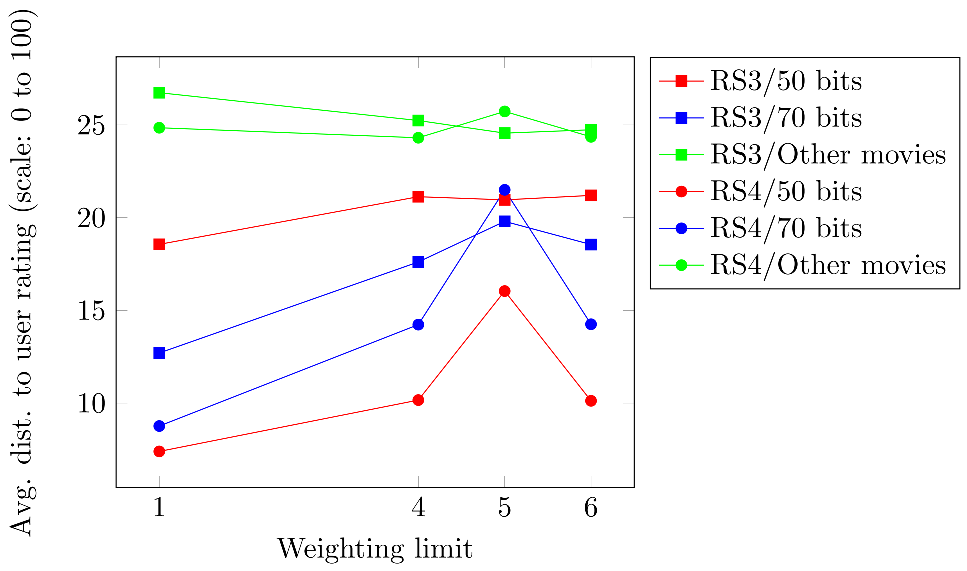


Figure 5.1: Results of criterion algorithm version 1. RS: Rating scale.

bits than with 50 bits. With more movies known, the probability is higher that one of the ratings might be off.

The recommendations with weighting limit 1 deliver the best results in comparison to other weighting limits, but the predictions for 'other movies' are all about on the same level. Overall, rating scale 4 appears to do better than rating scale 3.

'Other movies' seems to correlate with the 50/70 bits curves, aside from weighting limit 1. This is especially showing for rating scale 4. For rating scale 3, the recommendations after 70 bits show a little anomaly compared to the other curves.

In comparison to the preliminary test results, weighting limit 1 has prevailed as being the best choice for recommendations, whereas weighting limit 5 now doesn't exclusively deliver the best results for 'other movies'. This is probably explained by there being more participants who have done the test for weighting limit 5 (which is the first parameter to be tested) than for the others. It is difficult to determine a trend for the results, though.

5.1.2 Criterion Algorithm Version 2

The second version is tested for different parameters than the other because of the preliminary tests. Its results are displayed in Figure 5.2.

When comparing 50 to 70 bit, the results are more intertwined for rating

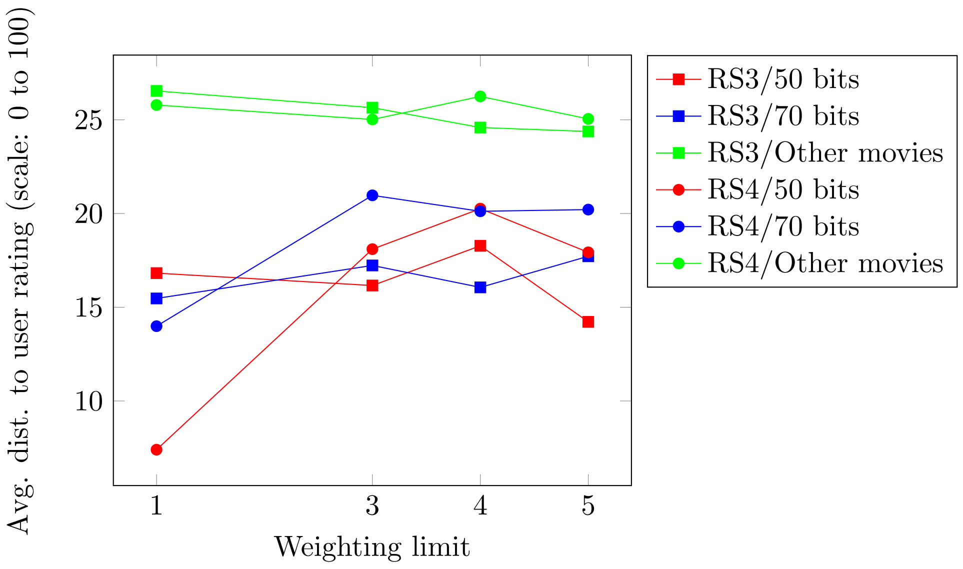


Figure 5.2: Results of criterion algorithm version 2. RS: Rating scale.

scale 3 this time, but with the other scale, the recommendations after 70 bit are generally worse again than after 50 bit.

Again, the users knew more of the movies that are recommended after 70 bits than after 50, which would explain this behavior. It was the same for rating scale 3 this time, however, which would explain why it didn't improve itself this time.

'Other movies' is more or less on the same level again for all parameters. Weighting limit 1 does still deliver good results, but rating scale 3 shows that weighting limit 1 isn't alone in being the best option. This time, it is less clear if rating scale 3 or 4 is to be favored: The best recommendations did come with rating scale 4 (and weighting limit 1), but rating scale 3 beats it on the other weighting limits.

The forms of the curves have less in common with one another this time, with RS3/50 bits, RS4/50 bits and RS3/other movies taking after one another, and the same for the other three (ignoring weighting limit 1).

Again, while weighting limit 4 delivers the best result for 'other movies' in the preliminary tests, it is not the case here; but again, there are more test results for weighting limit 4. Weighting limit 1 still proves to be a good choice, as it does in the preliminary tests.

5.1.3 Criterion Algorithm Version 3

The results for Criterion Algorithm 3 are shown in Figure 5.3.

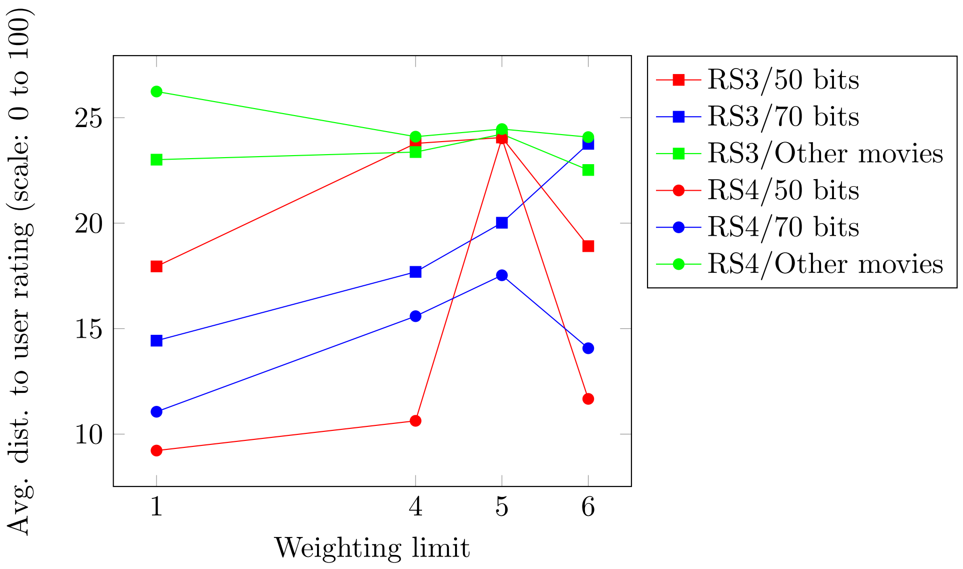


Figure 5.3: Results of criterion algorithm version 3. RS: Rating scale.

As with the first algorithm, using rating scale 3 shows improvement when going from 50 to 70 bits, whereas rating scale 4 does the opposite again, both with one exception. Again, the number of ratings is higher after 70 bits than after 50 bits.

Weighting limit 1 clearly takes the lead again for the recommendations, while for ‘other movies’, the result for rating scale 4/weighting limit 1 is a bit higher than the others. Rating scale 4 has 3 beat when it comes to recommendations, but the opposite is the case for ‘other movies’. When taking into account that the criteria get rated on the same rating scale for this version, rating the criteria on a higher scale seems to do better.

The curves are again not very similar to one another, but weighting limit 1 still performs like it does in the preliminary tests.

5.1.4 Criterion Algorithm overall

The results show that the average distances fluctuate heavily for the recommendations, while with ‘other movies’, the averages are more stable, all of them being around 25. This is probably due not being a lot of data available in the recommendations. For all recommendations, there are however at least 7 ratings each calculated average.

It would be very interesting to see how the algorithms evolve with an additional testrun, say with 90 bits of information; or having more recommendations after one testrun, since they don't seem to reach a saturation level.

Overall, the best results for recommendations are achieved with weighting limit 1 and rating scale 4 for every version, both for 50 and 70 bits. Since they are all similarly good, it is a strong indication of these parameters being a good choice. Figure 5.4 shows the results for the different versions plotted against each other.

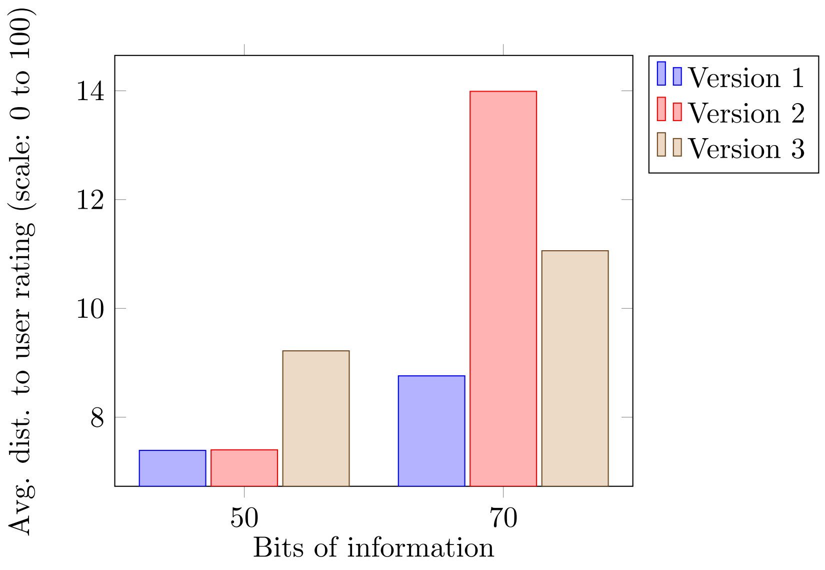


Figure 5.4: Results for rating scale 4 and weighting limit 1.

When it comes to recommendations, version 1 has the others beat, at least with the most favorable parameters. 'Other movies' is about on the same level for all the versions/parameters, which makes it difficult to draw conclusions from there.

Looking at the recommendations including the other parameters shows that for rating scale 4, version 1 beats the others overall, but for rating scale 3, version 2 performs better than the others. Considering that rating scale 4 performs better than 3 in version 3 of the algorithm, it seems safe to say that rating the criteria on a higher scale is better than a lower one.

5.2 Critic Algorithm

From the critic algorithm's recommendations, the users generally know less of the movies than from the criterion algorithm's recommendations. Since some of the averages can be very misleading when they don't have a lot of results, any calculated average distances with 2 or less average ratings or less than 5 total ratings are not shown in the following results.

5.2.1 Critic Algorithm Version 1

The results show that the number of critics that are used doesn't influence the rating much. The averages overall don't change a lot with the number of critics, not even when examining a single user's results. It is therefore advisable to use 20 over 30 critics, since this speeds up the algorithm (if only slightly).

Analyzing different critic distributions leads to more distinctive results. Figure 5.5 shows the average values for different parameter configurations and rating scales that share the same critic distribution parameter. The results all use a bad critic importance of 60%, since that is the BCI that is tested with. The recommendations shown are both after 50 bit and 70 bit.

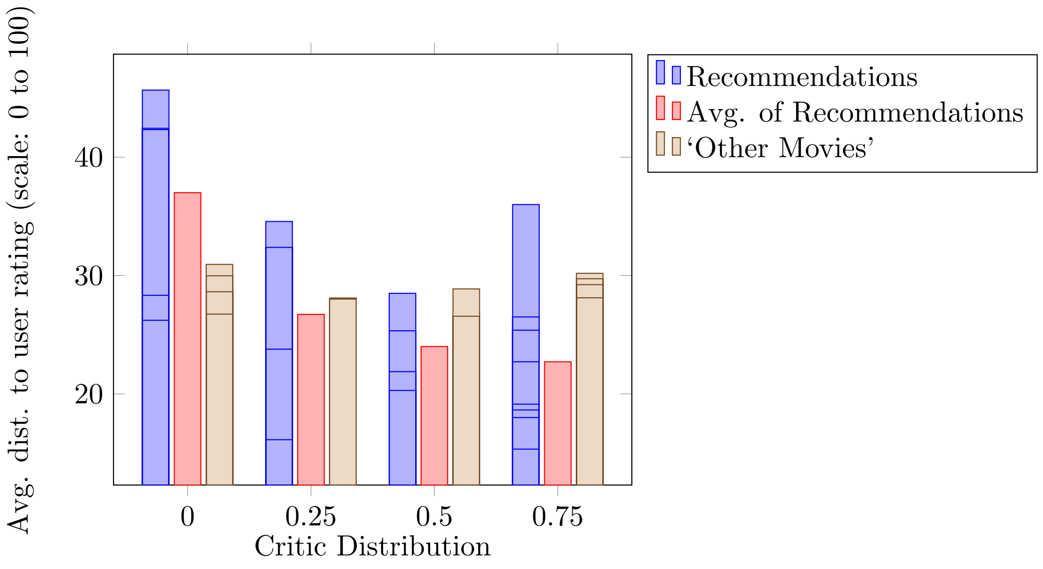
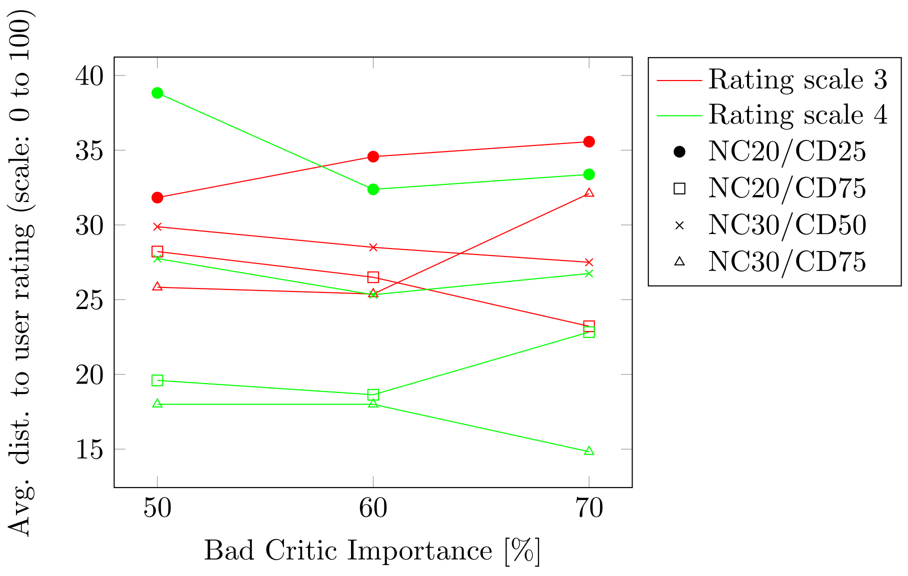


Figure 5.5: Average values of different parameter configurations, sorted by critic distribution.

Using a critic distribution of 0 leads to bad results in both recommendations and 'Other movies', contrary to the preliminary tests. A critic distribution of 0.25 shows spread results in the recommendations, whereas 0.5's results are closer together, on average, 0.5 beats 0.25. 0.75 has one rather bad run, but the others are pretty low, which shows in its average.

Figure 5.6 shows the results displayed by their different bad critic importances. Not shown are the average values when the critic distribution is set to 0, because then the BCI does not matter.



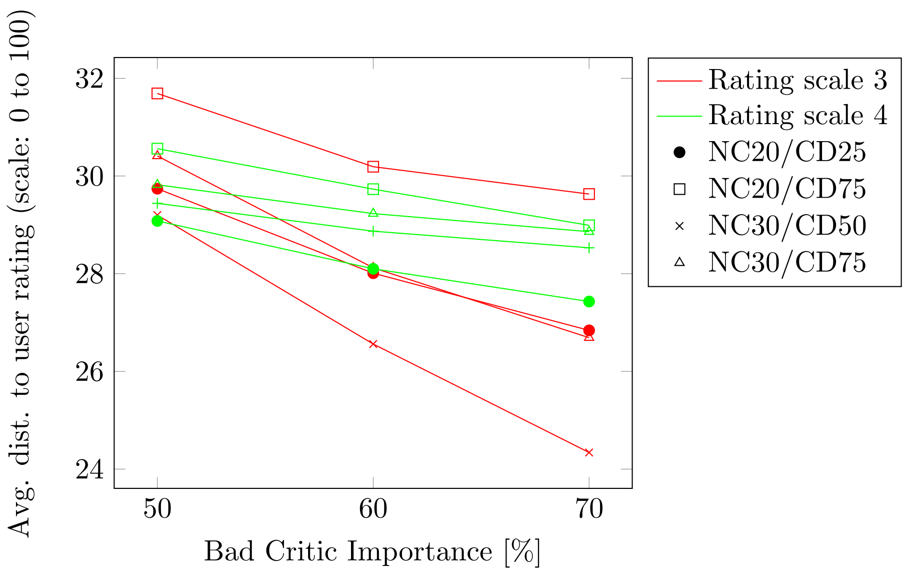


Figure 5.6: Average values of different parameter configurations, sorted by BCI. Top: recommendations, bottom: 'other movies'

The results for 'other movies' all show the trend that the higher the bad critic importance is, the better the results are. This is not always true with the recommendations, however. Most of the recommendations with $BCI = 50\%$ do have worse results than the others, but it can be said that a BCI between 60% and 70% probably yields the best results overall, not higher.

When it comes to ratingscale 3 vs 4, Figure 5.6 shows that rating scale 4 achieved the best results when it comes to the recommendations, but seems to do worse with ‘other movies’.

Compared to the preliminary test results, the critic distribution takes a surprising turn: the best becomes the worst, and the worst is among the best. The number of critics’ and the bad critic importance’s results are confirmed though: it is unknown whether 20 or 30 critics is better, and the bad critic importance of 60 (or maybe a bit higher) leads to better results in the recommendations than other values.

5.2.2 Critic Algorithm Version 2

Unfortunately, most of the testers did not know many recommendations of this algorithm, leading to many averages having less than 5 total ratings. The results for this test are therefore very inconclusive.

What can be said is that again, a critic distribution of 0 leads to very bad results, with average distances of 45 to 49. And the critic distribution of 0.5 seems to do better than the critic distribution of 0.25.

In ‘other movies’, the same results are shown for the bad critic importance as with version 1, the higher it is the better the rating. Most of the distances there are right under 30, with the exception of critic distribution 0.75. There, the distances are closer to 40 than 30.

5.2.3 Critic Algorithm Version 3

The number of critics makes a little greater a difference than in version 1. With critic distribution set to 0, the results are better with 20 instead of 30 critics. On the other hand, they seem to be worse with 20 than 30 when the critic distribution is set to 0.75. The second is less sure because the results for the testrun with rating scale 3, 20 critics and critic distribution 0.75 have not enough ratings.

Figure 5.7 shows the results for different critic distributions. As with the other versions, a critic distribution of 0 seems to produce the worst results, but not by as far. It is also more difficult to say whether 0.25, 0.5 or 0.75 brings the best results. 0.75 definitely has the worst result as far as ‘other movies’ is concerned, but it also makes for the single best prediction of all other parameters. 0.25 might be doing better because there are less results.

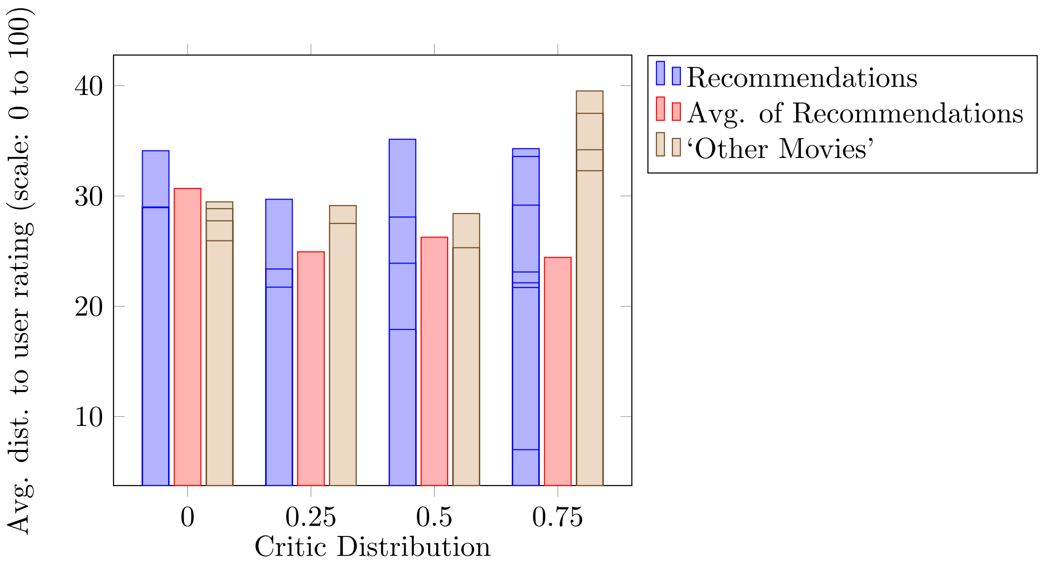
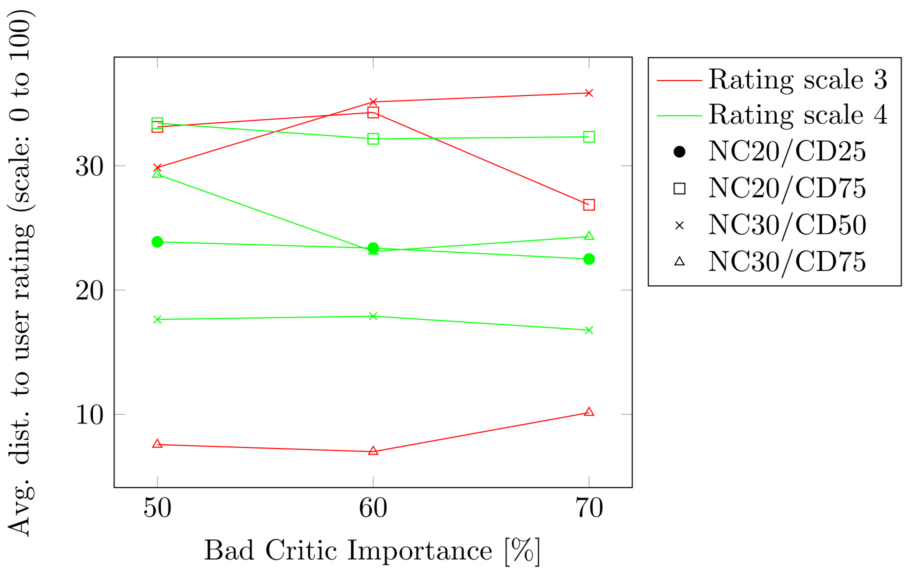


Figure 5.7: Average values of different parameter configurations, sorted by critic distribution.

The results sorted by bad critic importance are shown in Figure 5.8. As with the other versions, the results for 'other movies' are better the higher the BCI is set. However, they are more mixed for the recommendations than in version 1. It does not seem as though the BCI depends on the algorithm, because the same two algorithms with different rating scales don't show a lot of resemblances in their curves.



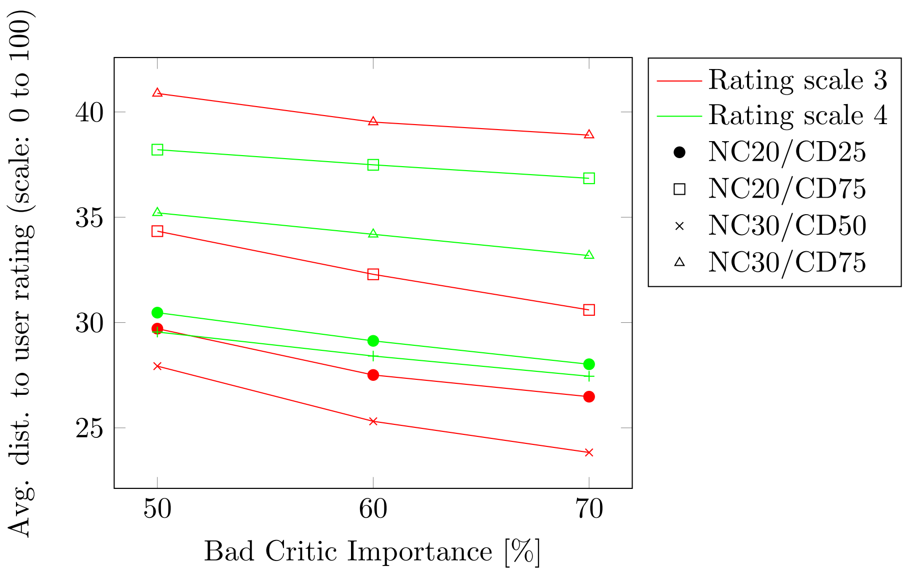


Figure 5.8: Average values of different parameter configurations, sorted by BCI. Top: recommendations, bottom: ‘other movies’

The rating scales seem to produce results equal to one another, although the results of rating scale 3 are wider spread.

5.2.4 Critic Algorithm overall

All three versions have more or less the same behavior when it comes to the number of critics and the bad critic importance.

The fact that the results are better with higher BCI-values in ‘other movies’ can be explained as follows: There can be a lack of information about certain movies that the algorithm has to predict a rating for (i.e. the critics that have rated the movie do not have a lot of ratings). A higher bad critic importance can therefore lead to lower-rated predictions, because the critics with matching taste have less part in the rating, and critics with opposite taste who have rated a movie positively will lower the prediction even further. This in turn leads to closer predictions because ‘other movies’ generally has more negative user ratings than the recommendations.

The results are unfortunately not as good as to be expected from the preliminary results, which indicates that the parameters may not have been chosen optimally.

The best overall performance seems to be achieved if you query the user’s favorite genres first, which also yields the best prediction. The parameters of that prediction are 30 critics, 0.75 of which match the user’s taste, and a bad critic importance of 60. An explanation for the third version’s success is that the users are bound to like movies belonging to their favorite genres.

5.3 Comparison of the two Algorithms

When looking at the recommendations, the criterion algorithm’s results are more satisfying and consistent than those of the critic algorithm. Its recommendations’ results don’t vary as much as of the critic algorithm. The critic algorithm might benefit from having less parameters to tweak, which allows for testing them more in depth. The number of critics for does not seem necessary to change, for one.

From the additionally tested generic algorithms (mentioned in section 4.1), the best results are achieved by the algorithms ‘Always guess 2/3 or 3/4’. This is due to a lot of users rating a lot of movies with 2/3 or 3/4, respectively. However, it beats most results of the criterion- and critic algorithm for ‘other movies’, as shown in Table 5.1.

Then again, the algorithm’s primary purpose is to recommend movies to the user, and if there’s not enough information on some of the movies, the predictions are bound to become worse than those of the recommendations. It would be interesting to see if/how much the results of ‘other movies’ can be improved by defaulting to a generic algorithm if there is not enough information on a movie.

A closer look at individual user data shows that in most of the results, the average distance of only movies with negative user ratings (mentioned in section 4.1) is worse than the average distance of movies with positive user ratings. This means that the algorithms generally rate the movies too high.

If this is the result of a lack of information, it can be fixed by defaulting to

Algorithms, rating scale 3	Average distance to user rating
Criterion V3, WL6	22.52
Always guess 2/3	23.06
Always guess audience rating	23.79
Critic V3, NC30/CD50/BCI70	23.83
Critic V1, NC30/CD50/BCI70	24.34
Criterion V2, WL5	24.38
Criterion V1, WL5	24.56
Always guess tomatometer	25.19
Critic V2, NC20/CD25/BCI70	26.49
Always guess random	35.47
Algorithms, rating scale 4	**Average distance to user rating**
Always guess 3/4	21.17
Always guess audience rating	22.08
Criterion V3, WL6	24.08
Always guess tomatometer	24.11
Criterion V1, WL4	24.31
Criterion V2, WL3	25.02
Critic V3, NC30/CD0	25.94
Critic V2, NC30/CD50/BCI70	26.54
Critic V1, NC30/CD50/BCI70	28.53
Always guess random	37.9

Table 5.1: Generic algorithms vs the best of each criterion/critic algorithm version in regards to ‘other movies’.

‘always guess 1/3 or 1/4’, which yields the lowest average distance of only movies with negative user ratings.

It would also be interesting to see if the algorithm’s ratings improve when rounding the predictions to the nearest possible rating on the rating scale.

In the end, it is not possible to make definitive statements on the performance of the algorithms, because of the low number of tests and the different numbers of testruns. There’s a high variance when it comes to the recommendations both in the number of ratings and the distances to the user’s rating, and it is unknown whether the recommendations that the user does not know are any good. Therefore, the recommendation’s results are to be taken with a grain of salt, and the results for ‘other movies’ don’t say too much about the performance because it is not possible to determine whether the algorithms had enough information to rate the movies unknown to them.

Chapter 6

Future Work

As a direct continuation of this thesis, the testing environment could be improved by making it run the tests in parallel, in order to decrease the time between two questions. This allows for testing more configurations in potentially less time.

The algorithms would also benefit from a user-friendly interface and distribution. An Android app or web application could be developed, and even be used for feedback on parameter configurations on a large scale.

One could also test the IMDb data against the Rotten Tomatoes data, to check among other things if using critics over a normal audience is desirable.

6.1 Other Algorithm Ideas

Two other ideas for algorithms have been considered during this thesis. The first is to directly ask the user about his favorite directors/writers/actors/production companies, with the assumption that the user likes every movie involving those favorites. If some few of their movies disagree with the user, he is prompted to mention them as well. Every non-favorite director/writer/actor/production company that has worked on one of the favorite's movies would then get points. Every non-favorite that has worked on one of the favorite's movies that the user doesn't like gets a severe penalty. Then, the algorithm searches for movies of the highest rated non-favorites that the favorites did not work on.

The second idea is to use PLSA, Probabilistic Latent Semantic Analysis[6]. PLSA is normally used to assign documents to an arbitrary set of classes based on the words they contain, its goal being to group similar documents using probability theory. The idea of applying the concept of PLSA to movie recommendation is derived from Jukefox[7], where PLSA is applied to classifying music.

For movies, PLSA can be used in a multitude of ways, it is possible to classify movies based on what critic liked which movie, treating the movie's written descriptions or the critic's reviews as documents or a combination of them.

Bibliography

[1] SQLite Home Page. https://www.sqlite.org/

[2] Overview (Java Platform SE 7). http://docs.oracle.com/javase/7/docs/api/

[3] Rotten Tomatoes: Movies — TV Shows — Movie Trailers — Reviews. http://www.rottentomatoes.com

[4] About Critics - Rotten Tomatoes. http://www.rottentomatoes.com/help_desk/critics.php

[5] Markus Frei: "Find your Personal Movie Critic". ETH Zürich, 2013.

[6] Ayman Farahat and Francine Chen: "Improving Probabilistic Latent Semantic Analysis with Principal Component Analysis". Palo Alto Research Center, 2006. http://www.fxpal.com/publications/fxpal-pr-06-372.pdf

[7] Michael Kuhn, Roger Wattenhofer and Samuel Welten: "Social Audio Features for Advanced Music Retrieval interfaces". October 2010. http://www.tik.ee.ethz.ch/file/30a4d2c6c579e13de4251b40f5490ad8/mmfat11301-kuhn_221.pdf

Appendix A

Keywords for the Criterion Algorithm

If any of the following keywords have been found in a written review, the score for the corresponding criterion gained +1 or -1 depending whether the ‘critic tomato’ of the review is fresh or rotten.

Story: story, storytelling, story telling, story-telling, story-teller, storyline, writing, written, plot, plotted, plotting, narrative, scripted, script.

Soundtrack: score, songs, soundtrack, music, audio.

Imagery: visual, visually, visuals, special effects, cgi, lighting, camerawork, camera work, camera-work, “to look at”, imagery, animated, animation.

Acting: acting, performance, performances, actor, actress, actors, actresses.

Impact: intense, intensely, tense, tensely, emotional, emotionally, impact, invigorating, invigoratingly, impressive, influential, exciting, touching, touchingly.

The keyword “sound” is used in reviews more often as a verb than to describe the soundtrack of the movie and is therefore not used.

Since some keywords appear in longer words that do not correspond to the criterion (e.g. ‘story’ and ‘history’), the keywords need to appear in one of the following SQL formattings (‘w’ is used as a placeholder for the keyword): “% w”, “% w...%”, “% w %”, “% w_”, “% w_ %”, “% w’s%”, “w %”.

The percent sign and the underscore character are wildcards in SQL. A percent sign is a placeholder for any arbitrary sequence of characters, the underscore is a placeholder for one single arbitrary character.

Appendix B

Sequel Identification

Sequels have a lot of different ways to be named, one of which is the use of numbers. But they are not used uniformly, some movies use arabic numerals, written numbers or roman numerals (i.e. ‘2’, ‘Two’, ‘II’).

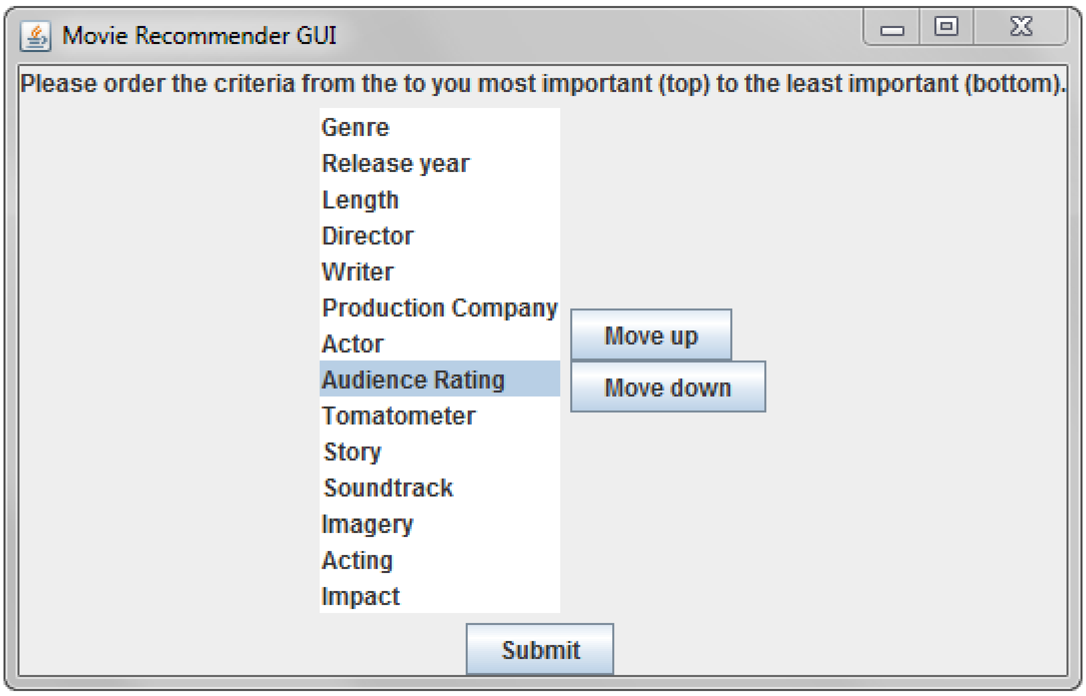
There are the easy cases where the title of the sequel is the title of the original movie with ‘[number]’ appended (“Toy Story”, “Toy Story 2”). Sometimes, the first movie is already numbered itself, which is why any movie ending with ‘one’/’1’/’I’ is checked for sequels by replacing the number (“Kill Bill Vol. 1”, “Kill Bill Vol. 2”).

Some titles have more text following after the appended number, separated by a colon or a hyphen (“Crank”, “Crank 2: High Voltage”; “101 Dalmatians”, “101 Dalmatians II - Patch’s London Adventure”).

There are titles without numbers that can still be recognized as sequels if they share the same prefix. A prefix needs to be followed by a separator to be recognized, namely a colon, a hyphen or the words “ and the ” (“Beauty And The Beast - Belle’s Magical World”, “Beauty and the Beast - The Enchanted Christmas”; “Resident Evil”, “Resident Evil: Apocalypse”; “Harry Potter and the Philosopher’s Stone”, “Harry Potter and the Chamber of Secrets”).

Appendix C

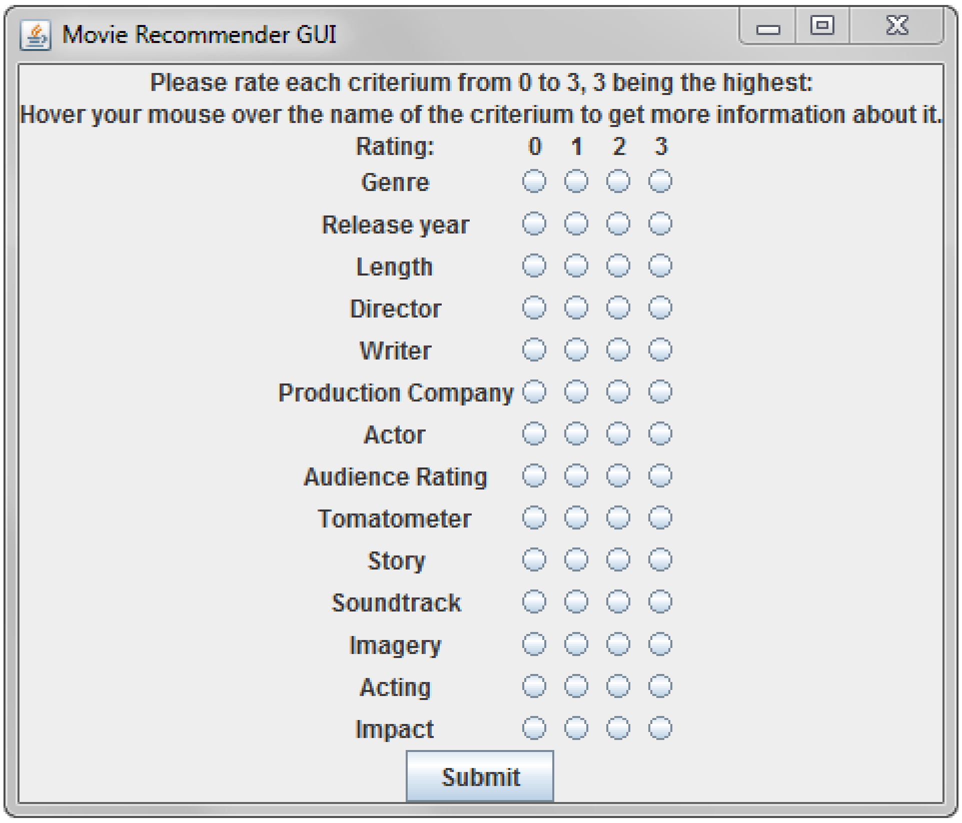
Screenshots of the User Interface


Movie Recommender GUI

Please order the criteria from the to you most important (top) to the least important (bottom).

Genre
Release year
Length
Director
Writer
Production Company
Actor
Audience Rating
Tomatometer
Story
Soundtrack
Imagery
Acting
Impact

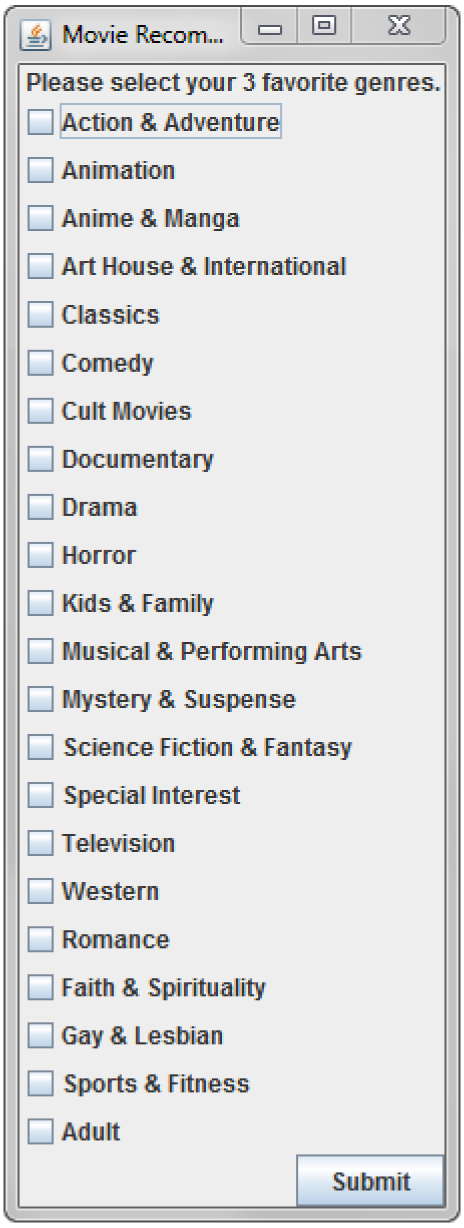
Figure C.1: The initial query of the criterion algorithm version 1 and 2.



Please rate each criterium from 0 to 3, 3 being the highest:
Hover your mouse over the name of the criterium to get more information about it.

Rating:	0	1	2	3
Genre	○	○	○	○
Release year	○	○	○	○
Length	○	○	○	○
Director	○	○	○	○
Writer	○	○	○	○
Production Company	○	○	○	○
Actor	○	○	○	○
Audience Rating	○	○	○	○
Tomatometer	○	○	○	○
Story	○	○	○	○
Soundtrack	○	○	○	○
Imagery	○	○	○	○
Acting	○	○	○	○
Impact	○	○	○	○

Figure C.2: The initial query of the criterion algorithm's third version. Radiobuttons are employed to get the appropriate user input.



Please select your 3 favorite genres.

- [ ] Action & Adventure
- [ ] Animation
- [ ] Anime & Manga
- [ ] Art House & International
- [ ] Classics
- [ ] Comedy
- [ ] Cult Movies
- [ ] Documentary
- [ ] Drama
- [ ] Horror
- [ ] Kids & Family
- [ ] Musical & Performing Arts
- [ ] Mystery & Suspense
- [ ] Science Fiction & Fantasy
- [ ] Special Interest
- [ ] Television
- [ ] Western
- [ ] Romance
- [ ] Faith & Spirituality
- [ ] Gay & Lesbian
- [ ] Sports & Fitness
- [ ] Adult

Figure C.3: The initial query of the critic algorithm's third version looks identical to the first version of the algorithm apart from the instruction.