EXAMENSARBETE INOM TEKNIK,
GRUNDNIVÅ, 15 HP
STOCKHOLM, SVERIGE 2017

Comparison of User Based and Item Based Collaborative Filtering Recommendation Services

PETER BOSTRÖM

MELKER FILIPSSON

KTH
SKOLAN FÖR DATAVETENSKAP OCH KOMMUNIKATION

Abstract

With a constantly increasing amount of content on the internet, filtering algorithms are now more relevant than ever. There are several different methods of providing this type of filtering, and some of the more commonly used are user based and item based collaborative filtering. As both of these have different pros and cons, this report seeks to evaluate in which situations these outperform one another, and by how big of a margin. The purpose of this is getting insight in how some of these basic filtering algorithms work, and how they differ from one another. An algorithm using Adjusted Cosine Similarity to calculate the similarities between users and items, and RMSE to compute the error, was executed on two different datasets with differing sizes of training and testing data. The datasets had the same amount of ratings but the second had less spread in the number of items in the set. The results were similar although slightly superior for both user and item based filtering on the second dataset compared to the first one. Conclusively, when dealing with datasets that are large enough for practical use, user based collaborative filtering proves to be superior in all reviewed cases.

Sammanfattning

Med en markant ökning av information och data på internet har algoritmer som filtrerar detta fått större relevans än någonsin. Det finns en mängd olika metoder för att förse den här typen av tjänst, och några av de mest använda är föremåls- och användarbaserade filtreringsalgoritmer. Båda dessa metoder har för- och nackdelar i olika grad, vilka denna rapport har som syfte att undersöka. Målet med detta är att få insikt i hur dessa filtreringsalgoritmer fungerar och hur de skiljer sig från varandra. En algoritm som använder sig av “Adjusted Cosine Similarity” för att beräkna likheter mellan användare och föremål, och “RMSE” för att beräkna felmarginalen, exekverades på två olika dataset med olika storlekar på tränings- och testdatan. Dataseten skiljde sig i spridningen mellan föremålen och hade olika antal användare, men var för övrigt lika gällande antalet betyg. Resultaten var liknande mellan de båda databaserna, men testet på den andra gav ett bättre utfall. Sammanfattningsvis, vid hantering av dataset stora nog att se praktisk användning var användarbaserad filtrering överlägsen i alla berörda fall.

Table of contents

1 Introduction

Lately, the demand for recommendation services have severely increased due to to the massive flow of new content on to the internet. In order for users to find the content they desire, competent recommendation services are extremely helpful. Finding the right movie or book among thousands of others that get released every year can be difficult. Therefore, automated recommendation services have been implemented to ease this task. There are however plenty of ways to implement these systems and enterprises want to make sure they implement the ones that best fit their business.

Recommendation based algorithms are used in a vast amount of websites, such as the movie recommendation algorithm on Netflix, the music recommendations on Spotify, video recommendations on Youtube and the product recommendations on Amazon. With the amount of content only increasing, research in this subject and implementations of it are further in demand, and in 2006 Netflix handed out an award of one million dollars to whoever could implement the best movie recommendation software for them to use on their service[9].

It is crucial that services recommend the correct items, as it leads to increased consumption, increased user satisfaction, increased profit, and is beneficial to everyone. Collaborative filtering is an effective and easy approach to solve this problem, as it evolves and learns from the user's preferences in order to further fulfill them in the future.

1.1 Problem Statement

The goal of this thesis is to compare the approaches of Collaborative Filtering, mainly User-based Collaborative Filtering and Item-based Collaborative Filtering, on datasets provided by the MovieLens database. This, in purpose of seeing their performances, equalities and differences. The thesis aims at investigating the following:

- Based on database sparsity, size of training and testing data, in which situations are the different approaches to *Collaborative Filtering* superior to one another?
- What are the main equalities and differences between the different algorithms?

1.2 Scope

Two datasets will be used to create subsets of datasets to train the program. The aim is not to reach high performance, but to compare the different approaches to one another. User-based filtering is expected to be superior when dealing with big amounts of data, whereas item-based collaborative filtering is expected to perform better on smaller datasets.

2 Background

There are two major different approaches to collaborative filtering, *item based* and *user based*. Item based filtering uses similarity between the items to determine whether a user would like it or not, whereas user based finds users with similar consumption patterns as yourself and gives you the content that these similar users found interesting. There are also hybrid approaches, which seek to utilise the strengths of both of these approaches whilst removing each of their weaknesses[3].

There are two main approaches to collaborative filtering: Model Based and Memory Based. This paper will discuss Memory Based collaborative filtering, as user based and item based filtering fall under this category. These two are mainly different in what they take into account when calculating the recommendations. Item based collaborative filtering finds similarity patterns between items and recommends them to users based on the computed information, whilst user based finds similar users and gives them recommendations based on what other people with similar consumption patterns appreciated[3].

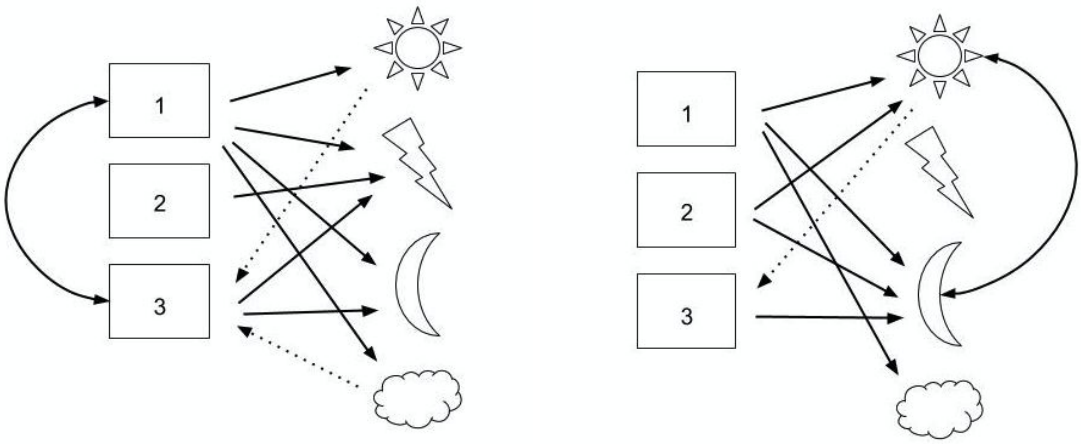

Fig 1: The picture depicts the different approach that user based and item based collaborative filtering takes. The half dotted lines represent recommendations based on the users preferences and similarities to the left, and based on similar items on the right.

Collaborative filtering is one of the most widely used algorithm for product recommendation, and it is considered effective[7].

Hybrid solutions can be useful in order to recommend content to users with unique or wide tastes, as it will be hard to find a “close neighbour” or someone with a similar consumption

pattern to that user. A regular item based or user based solution may prove to be unsatisfactory in this situation[3].

2.1 User-based Collaborative Filtering

The report is focusing on the “nearest neighbour” approach for recommendations, which looks at the users rating patterns and finds the “nearest neighbours”, i.e users with ratings similar to yours. The algorithm then proceeds to give you recommendations based on the ratings of these neighbours[2].

In a fixed size neighbourhood, the algorithm finds the X most similar users to you and use them for a basis of recommendation. In a threshold based neighbourhood, all users that fall within the threshold, i.e are similar enough are used to provide recommendations[8]. This report will use the threshold based neighbourhood as it makes more sense to use data that are similar enough, and not give bad recommendations to certain users simply because the closest neighbour was really far away. This will lead to some users getting better recommendations than others (as they have more similar users for the algorithm to work with), but it will at least not give bad recommendations where no recommendations might have been preferred. It will also not neglect similar users just because some users are even more similar, and it makes sense to use all good data we have at our disposal.

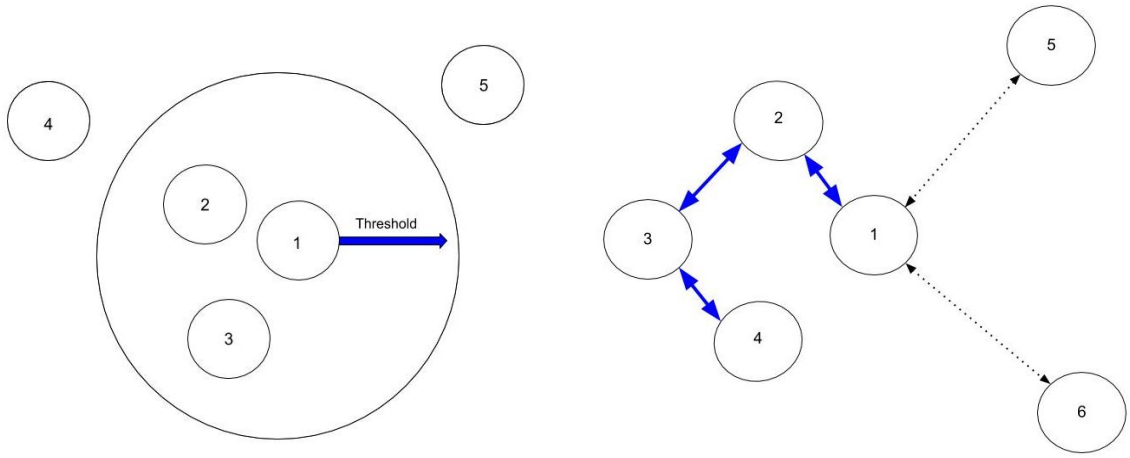

Fig. 2: the image on the left depicts a threshold based neighbourhood. User 1 would get recommendations from users 2 and 3, but not from 4 and 5 as they are outside the threshold. The image on the right depicts a fixed size neighbourhood. User 1 would get recommendations from users 2, 3 and 4, but not from 5 and 6 as it in this example uses the three closest neighbours for recommendations[8].

2.2 Item-based Collaborative Filtering

Item based collaborative filtering was introduced 1998 by Amazon[6]. Unlike user based collaborative filtering, item based filtering looks at the similarity between different items, and does this by taking note of how many users that bought item X also bought item Y. If the

correlation is high enough, a similarity can be presumed to exist between the two items, and they can be assumed to be similar to one another. Item Y will from there on be recommended to users who bought item X and vice versa[6].

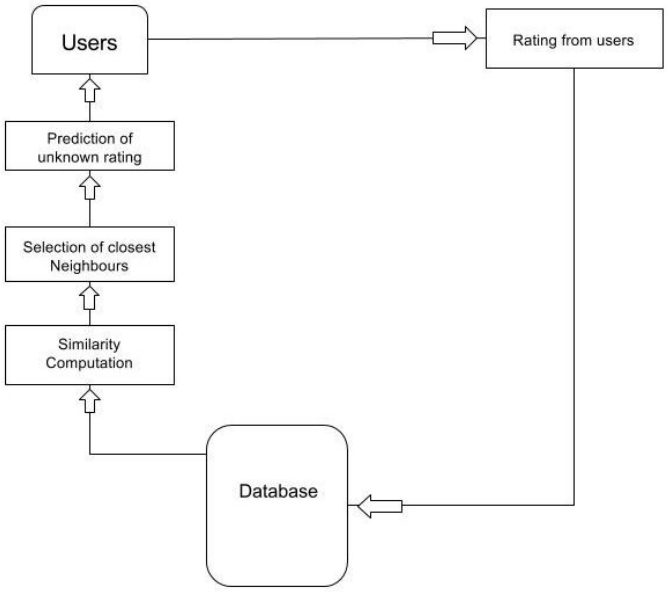

Fig. 3: the picture depicts a graph of how users ratings affect their recommendations[5]

2.3 Collaborative Filtering-System Problems

Problems that commonly occur with collaborative filtering include (but are not limited to) the following:

The *early rater problem* occurs when a new user is introduced to the system and has yet to rate a selection of items significantly large enough for the service to start suggesting similar items. A simple solution to this would be to have the user rate content they may have consumed on another site or platform in order to let the algorithm have a sufficiently large amount of data to start recommending products with an acceptable probability of success. It is of course harder to determine what for example the closest neighbour is if there are a vast amount of users who rated content similarly to this user, but rated other content severely different from one another[10].

The *sparsity problem* occurs when there are too little information to work with in order to provide the user base with decent approximations as to which products they would likely prefer. The sparsity problem occurs if the data is too spread, and while users might have rated a fair amount of movies, too few have rated the same movie to make accurate predictions[10].

Cold start occurs when one or several users or products is added to the system, and not enough data is recorded to provide optimal recommendations. This can affect an entire recommendation algorithm if the service is newly established because it can not provide

acceptable recommendations to any one user as of yet. This is a subject that has been researched for a long time and is still a problem in many recommendation systems[11].

A *gray sheep* is a user which has no obvious closest neighbour and seems to rate content in an unpredictable pattern unlike that of any other user. A gray sheep is a problem as it can be hard to estimate what the user might like or dislike when there are no similar consumption or rating patterns[10].

It takes a certain amount of time and amount of ratings for this new element to be introduced into the system in a way such that it works similarly to the rest of the objects within it. It is simply the process that leads up until the time when element has been correctly established within the environment.

A *shilling attack* is when a person or group of people create multiple accounts in order to promote certain content and take away user's interests in other, in an attempt to promote their own products and hurt their competitors, and is an attempt to manipulate users into buying, watching or subscribing to a certain type of content based on a hidden agenda[14].

2.4 Adjusted Cosine Similarity

Adjusted cosine similarity calculates similarity between users and items by calculating the difference in the vector angles between the ratings. It is calculated according to the following formula:

$$AC(i,j) = \frac{\sum_{U \in u_{ij}} (r_{ui} - \bar{r}_u)(r_{uj} - \bar{r}_u)}{\sqrt{\sum_{U \in u_{ij}} (r_{ui} - \bar{r}_u)^2 \sum_{U \in u_{ij}} (r_{uj} - \bar{r}_u)^2}}$$

Fig. 4: The image depicts the formula for adjusted cosine similarity *for computation of similar weights of items [12].*

Where r_{ui} represents the rating that is given by the user u to the item i and r_{uj} represent the rating that is given by user u to item j, $\bar{r}_u$ indicate the average ratings that are given by user u[12].

Adjusted cosine similarity also takes into account that users might rate items differently high in general. One user could have 3 as their average rating and consider 4 a high rating, whilst another might give items a 4 on average, and consider 3 to be a really bad score. We adjust for this by subtracting the average rating for each user from that user's ratings[15].

Adjusted cosine similarity slightly outperforms pearson correlation when dealing with item based recommendations, and is as effective regarding user based collaborative filtering[13].

2.5 Root Mean Square Estimation (RMSE)

Root mean square estimation is an estimate for the average difference between the predicted value and the actual value. Root mean square estimation is used as a protective measure to reduce the individual impact from the errors of each individual measurement, so that no one majorly faulty estimation will skew the result too much. Root mean square estimation is calculated according to the following formula:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Fig. 5: the image depicts the formula for RMSE, root mean square estimation. In this formula, n is the number of estimations, y_j is the current estimation, and $\hat{y}_j$ is the average value of the estimations.

3 Methods

3.1 Software and Hardware

A program was written in the programing language Python using the libraries *pandas*, *numpy* and *sklearn*. We chose this language as our experiments requires many matrix computations and python appears to be the best and fastest option when it comes to that certain aspect.

- http://pandas.pydata.org/
- http://www.numpy.org/
- http://scikit-learn.org/stable/

These tools were all strongly recommended for machine learning in python and had a lot of practical functions that made the implementation easier.

The results were computed on an *Acer aspire E 15 521 8453* laptop with an *AMD radeon quad-core processor A8-6410* with up to 2.4 gigahertz capacity and an *8 gigabyte DDR3 L* random access memory.

3.2 Datasets

The first tests were performed on the “ml-100k” database which contains over one hundred thousand ratings from 671 different users across 9066 different movies. We chose this dataset as the movieLens database suggested this one for education and research. It was

also large enough to provide reasonable results without one execution taking an unreasonable amount of time.

The second batch of tests were performed on an equally large database of one hundred thousand ratings, with an equal amount of ratings and roughly the same amount of users, but only a quarter as many movies as in the other database. This resulted in each person having watched similar movies to each other person in the database, meaning there were more ratings and data on each individual movie. It also meant the users were more “similar” to one another as an equal number of people were distributed on a smaller set of movies. This was therefore expected to yield better results.

The reason for using multiple datasets is to verify that the theories are correct, and that a database with more people comparable to one another provides more data and yields better results. It is also interesting to see if user based or item based is more heavily affected than the other when the data is more compact.

The mentioned databases can be found on the grouplens website (https://grouplens.org/datasets/movielens/) and the names of the first and second databases are *ml-latest-small.zip* and *ml-100k.zip*.

3.3 Implementation

Two matrices were created from the dataset, one for training and one for testing. Each row in both of the matrices represent a unique user whilst each column represented a unique item. The algorithm then created two new matrices with their estimated similarities calculated using the cosine similarity formula (based off of the training matrix), one being for user based filtering and the other for item based. The algorithm then tried to predict how certain users would rate certain items based on the two previously calculated matrices, and used root mean squared estimation to calculate how far off the estimations were based on the test matrix. The program was then executed one hundred times per test, writing its result for both user based and item based filtering after each iteration. The results were then put into an excel sheet, where diagrams were created based of the data from the graphs.

In the initial test we used 75% of the database as the training data and 25% as the testing data. The data was put together and was used to make a graph presenting the results. This initial test took roughly 2 hours for the computer to complete.

The second test was performed yet again on the complete database of one hundred thousand ratings, but this time with 90% of it as training data and 10% as testing data. This test took roughly the same amount of time as the first test.

The third test was performed on the complete database for the third time, but this time with half (50%) of the database as the training matrix and the other half of it as the testing one. This test took roughly the same amount of time as the first two tests.

The same three tests (same percentages of training and testing data) were then performed on the second database, which had more compact data. These three tests took significantly less time than the ones performed on the other database, and were all individually done in approximately twenty minutes.

4 Results

When discussing values in this chapter we are often referring to how far off the estimated ratings were compared to the actual ratings.The values are calculated using Adjusted Cosine Similarity and Root Mean Square Estimation. A lower value is therefore a better result, as the program provided us with estimates closer to the actual values provided by the database.

4.1 Dataset 1

4.1.1 75/25

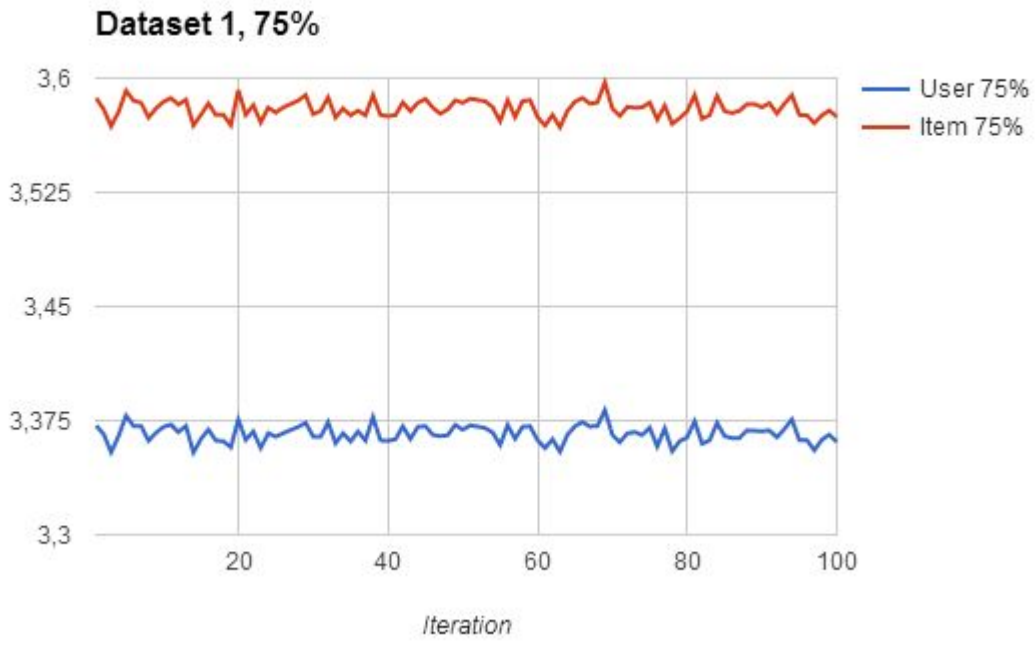

Diagram 1: Depicts the result of the 100 executions in test 1. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result from the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

The x-axis is the iteration from which the result was gathered, and the y-axis is the distance from the Root Mean Square Estimation to the actual user score. We can conclude that the user based algorithm performed better in this scenario, as the RMSE from the real values were significantly smaller in all 100 iterations. Each iteration for user based and item based was run on identical training and testing data, which resulted in similar patterns in the two functions.

Which data which was used in each iteration was taken randomly from the database, resulting in smaller fluctuations between the results of each iteration.

The fluctuations appeared to be minimal, ranging from 0.83% for item based, and 0.5% for user based, suggesting that which data used for training and which was used for testing had a minimal impact on the results.

4.1.2 90/10

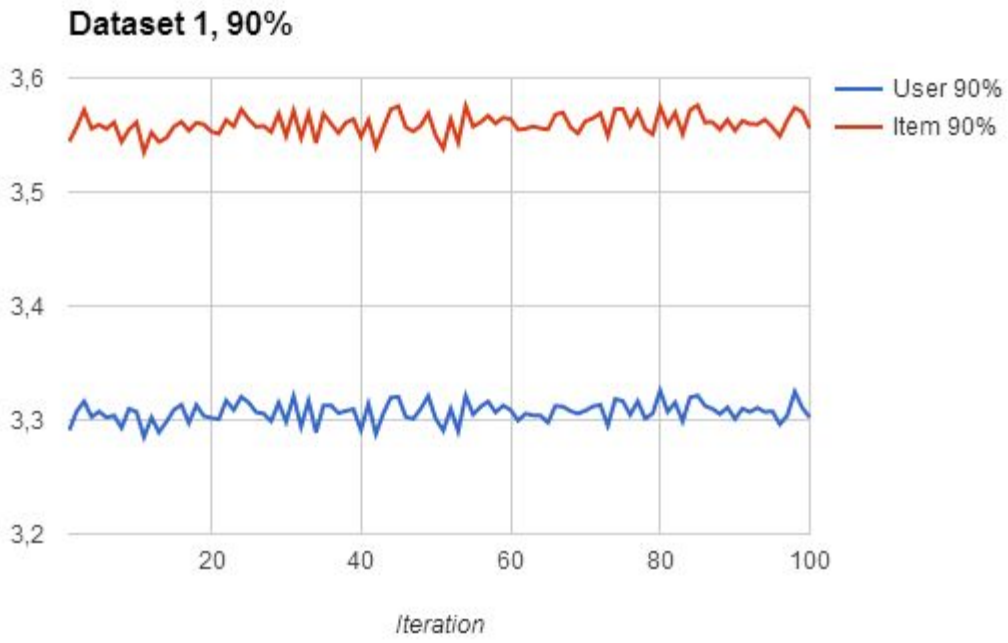

Diagram 2: Depicts the result of the 100 executions in test 2. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result from the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

When increasing the size of the training data from 75% to 90% (and as a side effect downscaling the size of the testing data from 25-10% of the complete size of the database) we got slightly improved results, and very slightly increased sizes on the fluctuation of values between each iteration. This is logical as the data on which the results are tested is significantly smaller than in the last experiment, but the fluctuations still have hardly any impact on the results.

The fluctuations are in this case 1.1% for item based, and 0.9% for user based, which is still minimal, suggesting that 10% is a sufficient amount of testing data while working with a database of 100,000 ratings.

4.1.3 50/50

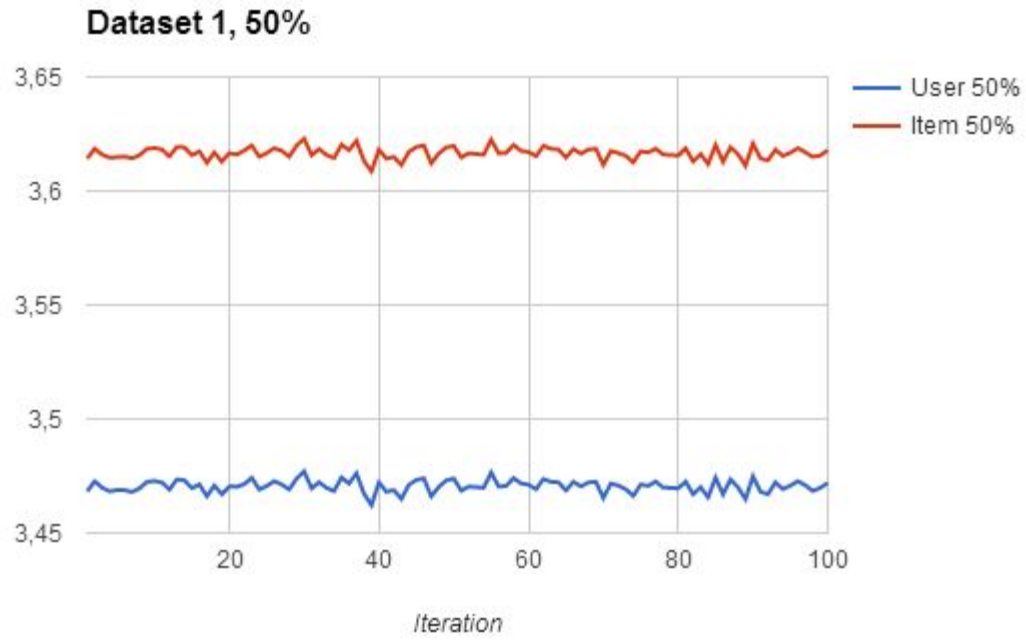

Diagram 3: Depicts the result of the 100 executions in test 3. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result form the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

Using merely half of the database for training and half of it for testing resulted (as expected) in the worst result as of yet. Since the testing data matrix was bigger we had smaller fluctuations, although hardly noticeable from the first test, but way worse results than in the previous two experiments.

4.1.4 All Tests Dataset 1

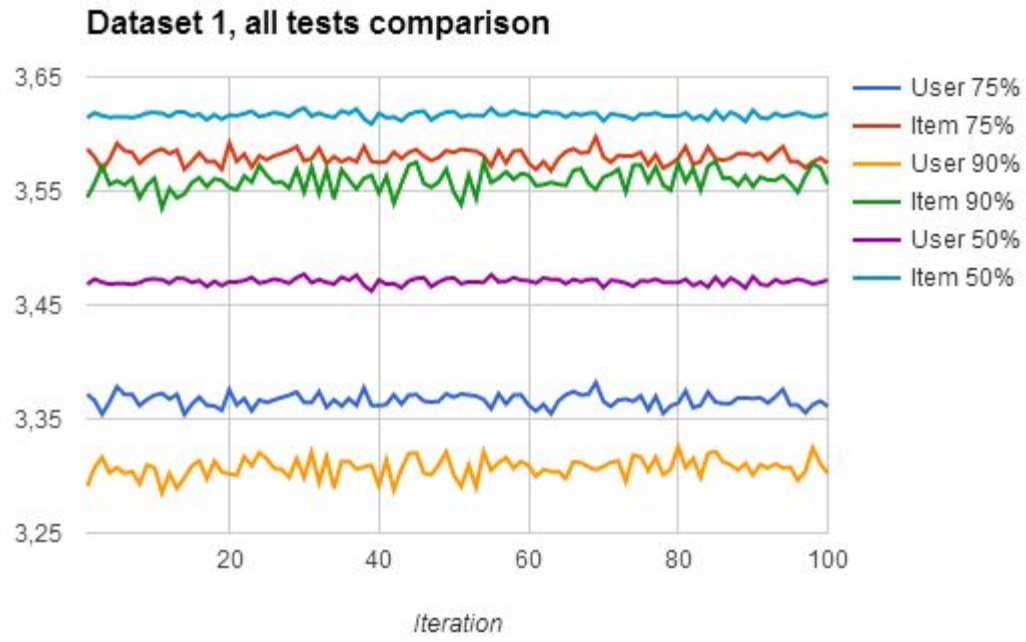

Diagram 4: Depicts the results from the first 3 tests next to each other for comparison. The x-axis is the current iteration of the program, simply there to provide 100 different results from the program execution, and the y-axis is the RMSE from the actual database value to the value that the algorithm provided in this current iteration.

From the previous three experiments, 90% training data proved superior to 75% training data, which was superior to 50% training data, for both the user and item based algorithms. Note that only item n and user n was run on the exact same training data. For example: item 1 and item 2 are not being trained on identical data sets, and are therefore not directly correlated to each other on each individual value on the x-axis. The fact that Item 1 sometimes outperforms Item 2 can have had to do with it receiving better training data and not necessarily it being the better algorithm for that individual situation, as the training data is not necessarily the same. The graph is mostly there to be used as an overall measurement of which parameters provided the best result in the span of one hundred executions.

4.2 Dataset 2

4.2.1 75/25

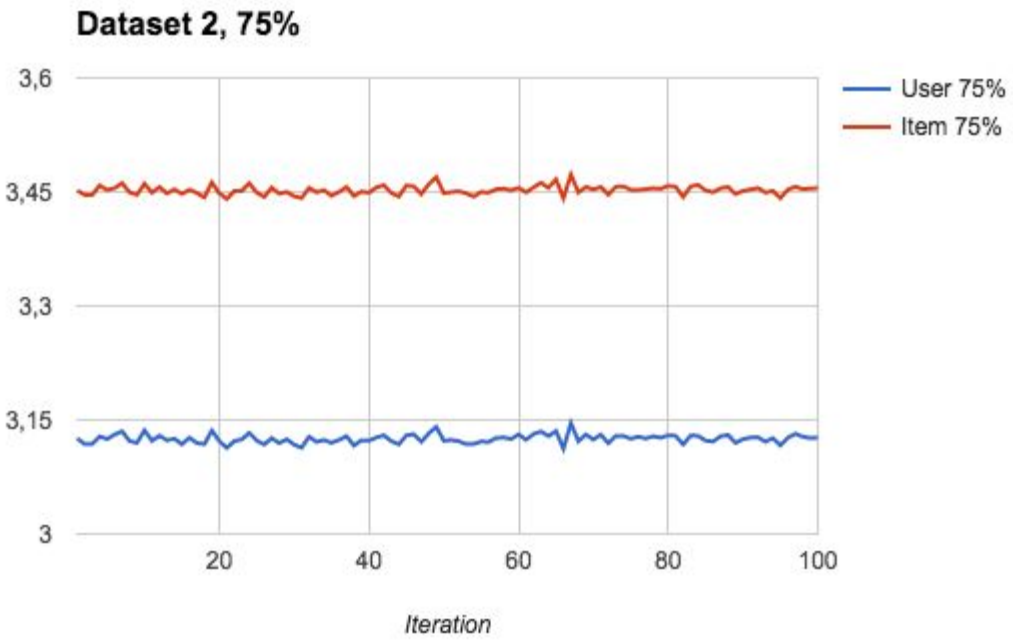

Diagram 5: Depicts the result of the 100 executions in test 1 of Dataset 2. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result from the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

Just as in dataset 1 user based collaborative filtering outperformed item based collaborative filtering in all one hundred executions. The fluctuation is within similar range to that of the corresponding test on the first dataset.

4.2.2 90/10

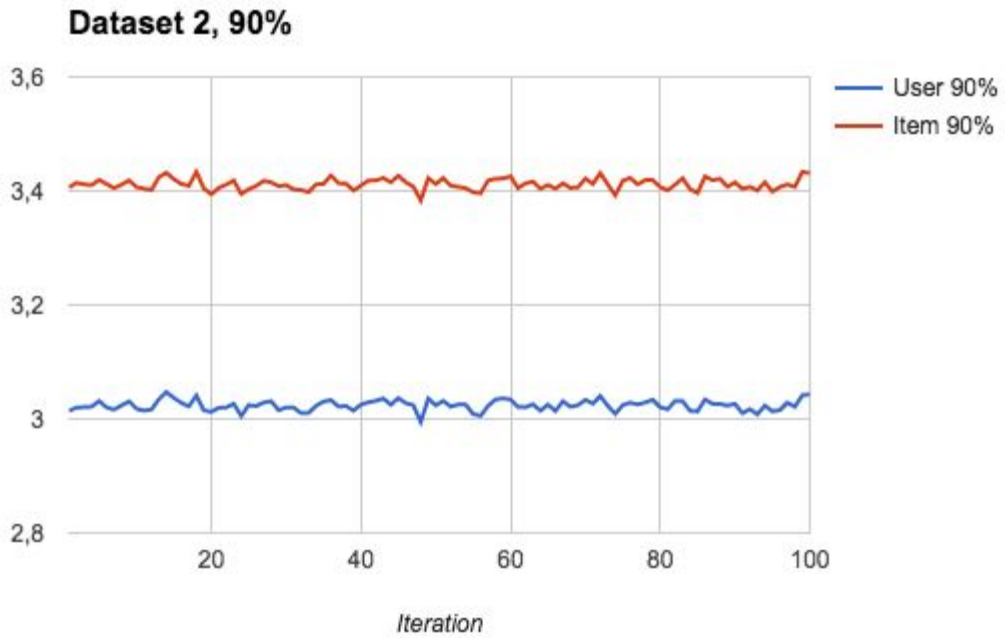

Diagram 6: Depicts the result of the 100 executions in test 2 of Dataset 2. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result from the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

Just as in dataset 1 user based collaborative filtering outperformed item based collaborative filtering on all one hundred executions, and just as before 90% training data was superior to 75% training data.

4.2.3 50/50

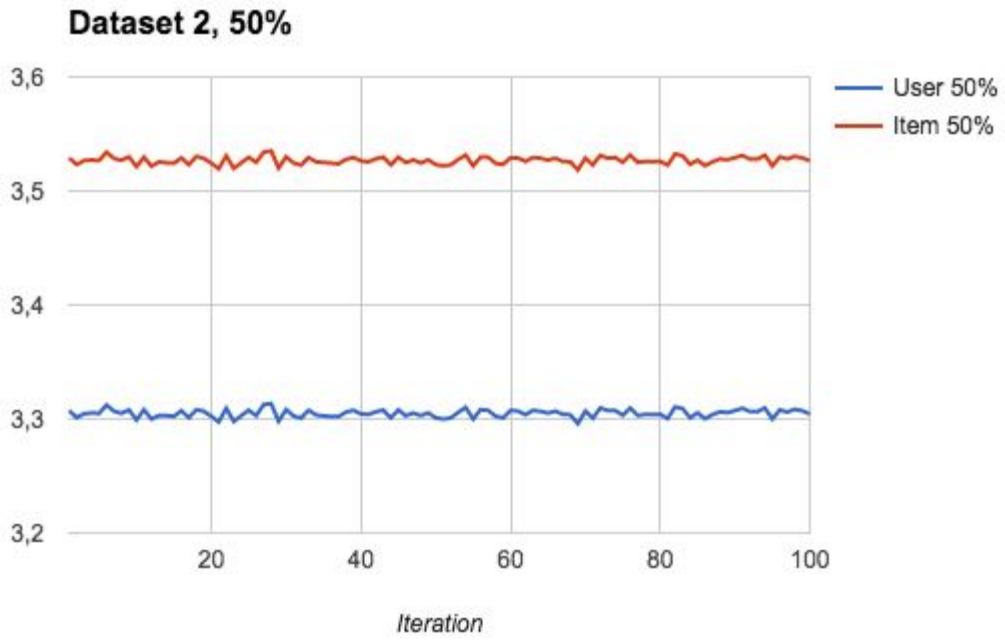

Diagram 7: Depicts the result of the 100 executions in test 3 of Dataset 2. The x-axis is the number of the current iteration and the y-axis is the RMSE value. The red line represents the result from the item based estimation, and the blue line the user based. The user based prediction proved superior in this example.

Just as in dataset 1 user based collaborative filtering proved superior to item based collaborative filtering on all one hundred executions, and once again the results were worse, but more consistent, than on the tests with 75% and 90% training data.

4.2.4 All Tests from Dataset 2

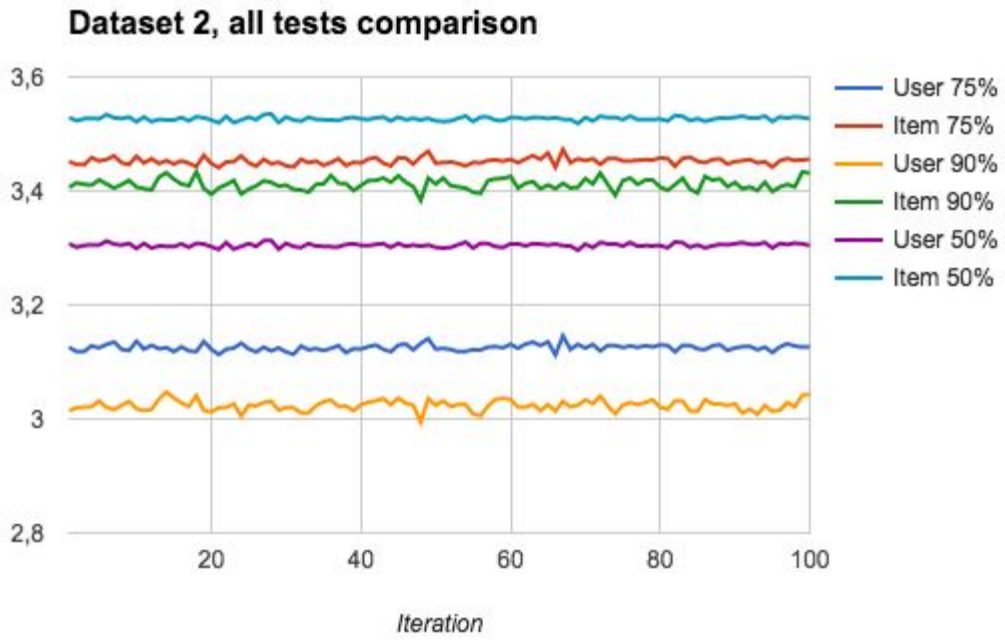

Diagram 8: Depicts the results from all executions in test 1,2 and 3 of Dataset 2.The x-axis is the current iteration of the program, simply there to provide 100 different results from the program execution, and the y-axis is the RMSE from the actual database value to the value that the algorithm provided in this current iteration.

Once again, user based collaborative filtering proved superior in all cases, and yet again there is a positive correlation between the size of the training data and the quality of the results. Item based collaborative filtering was once again more heavily affected by the size of the training and testing data, which had a way smaller impact on the results coming from the item based collaborative filtering.
There was also a smaller fluctuation in the 50% training data test, just as on dataset 1.

5 Discussion and Conclusion

5.1 Discussion

5.1.1 Datset 1

In each of the tests on the entire database, user based collaborative filtering proved to be superior, with estimations ranging from roughly 1.25 to 2.5 closer to the actual ratings. The user based estimations had bigger impacts from smaller training sizes than the item based estimations by nearly 250% in difference compared to the user based ones, implying user

based filtering performs much better the larger the training data set is. The item based filtering is also slightly improving with bigger training data sets, but by a significantly smaller amount.

Test number 3 for both item and user based collaborative filtering, using 50% of the dataset as training data had the most consistent results. This is likely due to the testing matrix being larger, resulting in each deviation therein having a smaller impact on the larger picture. This in turn decreases the misrepresentations stemming from gray sheep. It is also logical that it performs worse, as it had less data to train on, and deviant ratings in the testing matrix have a bigger impact on the estimations that this algorithm provided than the other two.

As the curve's spikes and valleys for item based and user based collaborative filtering seem to be nearly identical for each individual test, we can assume that the algorithms work the same for each indata, and that they are simply superior to one another based on the size of the training and testing matrices, and that the individual values have little to no impact, at least for big enough data sets.

We can from this conclude that more training data provides better results while larger testing data provides smaller fluctuations. The size of the fluctuations between having 50% test data and 25% hardly had any impact on the results in this case anyway, and we conclude that for databases this large, a higher amount of training data is superior to a higher amount of testing data, as long as the testing data is large enough to not skew the result too much by one rating being an oddity in comparison to the others. 10% testing data proved to be far enough in this case.

5.1.2 Dataset 2

The results yielded from the tests on the second database gave similar results to those of the first one. The results were in all cases better, if only slightly, but the same patterns appeared and a larger size of the training matrix always proved to perform better. The test with 50 % training data yet again gave smaller fluctuations between iterations, once again due to there being gray sheep or odd users in the testing matrix whose data was used for confirmation, had a smaller impact on the result as they are a smaller percentage of all users being used as correction in this case.

The reason for dataset 2 giving better results than dataset 1 is probably due to a lack of the cold start problem and the sparsity problem, as there are fewer items and users but still the same amount of ratings, which gives us a more compact matrix with users more likely to be similar to one another. We also do not have problems with gray sheep, as there are fewer movies in the dataset, and therefore harder to find users who are that different in their ratings. The early rater problem is also less severe, as the algorithm will find a similar user to the early rater faster in comparison to dataset 1, since the users are more homogenous in this dataset. These points combined will in turn lead to improved outcome, as demonstrated in the results section.

The fact that the tests were completed over five times as quickly on the second database is likely due to the size of the matrix that represents the database is significantly smaller, as there is only a quarter of the amount of movies in this database compared to the first one. One of the matrix's dimensions will only be a quarter of that of the other database's matrix, which will be significantly faster to compute.

5.2 Method Evaluation

We chose to use Adjusted Cosine Similarity instead of Pearson correlation in our algorithm. This because when comparing two users that are different from each other Pearson correlation ignores the items which only one or neither of the users rated, meanwhile Adjusted Cosine Similarity keeps them but sets the rating to 0. Therefore Adjusted Cosine Similarity will look not only on the data that is the same, but also the data that is different, which will in turn give us a more accurate result[13].

5.3 Conclusion

Conclusively, the study shows that datasets large enough to see practical usage, user based collaborative filtering is superior on all of the tested cases. Larger amounts of training data lead to better results, and the amount of testing data does not need to be needlessly large to achieve a decent outcome. Datasets with less spread in the amount of items, and therefore more homogenous users leads to superior results, which is why the tests in dataset two gave better results than the ones in dataset one.

User based and Item based collaborative filtering provide similarly looking results, but user based is better on the sizes tested in this report. User based collaborative filtering also improves faster as the amount of training data is increased.

In the second database the user based results were also more improved (by roughly 0.1) than item based (which was improved by roughly 0.05).

6 References

[1] B. Sarwar, G. Karypis, J. Konstan, J. Riedl (2016) Item-based Collaborative Filtering Recommendation Algorithms (http://www-users.cs.umn.edu/~sarwar/sdm.pdf) (Accessed 20-02-2017)

[2] R. Devooght, H. Bersini (2017) Collaborative Filtering with Recurrent Neural Networks (Accessed 15-02-2017)

[3] Peng Yu (2015) Collaborative Filtering Recommendation Algorithm Based on Both User and Item(Accessed 21-02-2017)

[4] Movielens database, https://grouplens.org/datasets/movielens/ (Accessed 15-02-2016)

[5] L. T. Ponnam, S. N. Nallagulla, S. D. Punyasamudram, S. Yellamati (2016) Movie Recommender System Using Item Based Collaborative Filtering Technique (Accessed 21-02-2016)

[6] https://www.google.com/patents/US6266649 (patent of item based collaborative filtering)

[7] Y. Guo, M. Huang, T. Lou (2015) A Collaborative Filtering Algorithm of Selecting Neighbors Based on User Profiles and Target Item (Accessed 22-02-2016)

[8] U. Farooque (2016) Implementing User Based Collaborative Filtering to Build a Generic Product Recommender Using Apache Mahout (Accessed 22-02-2016

[9] X. Su, T. M. Khoshgoftaar (2009) A Survey of Collaborative Filtering Techniques (Accessed 24-02-2016) https://www.hindawi.com/journals/aai/2009/421425/

[10] Recommender Systems (KORREKT FORMATERING PÅ DENNA KÄLLA SAKNAS!) http://recommender-systems.org/hybrid-recommender-systems/ (grey sheep, sparisty osv osv)

[11] Anna-Karin Evert , Alfrida Mattison, (2016) Rekommendationssystem med begränsad data (Accessed 21-02-2017) https://www.kth.se/social/files/588788adf2765412b6e8ec74/A-KEvert_AMattisson_dkand16.pdf

[12] Lakshmi Tharun Ponnam, Sreenivasa Deepak Punyasamudram, Siva Nagaraju Nallagulla, Srikanth Yellamati (2016) Movie Recommender System Using Item Based Collaborative Filtering Technique (Accessed 19-02-2017)

[13] Michael Ekstrand (2013) Similarity Functions for User-User Collaborative Filtering https://grouplens.org/blog/similarity-functions-for-user-user-collaborative-filtering/ (Accessed 03-05-2017)

[14] Alper Bilge, Zeynep Ozdemir Huseyin Polat (2014) A novel shilling attack detection method http://www.sciencedirect.com/science/article/pii/S1877050914004347 (Accessed 20-02-2017)

[15] http://www.cs.carleton.edu/cs_comps/0607/recommend/recommender/itembased.html