Collaborate Filtering for Digital Publishing

Rasmus Theodorsen

Copenhagen 2013
M.Sc.-2013-60

Technical University of Denmark
DTU Compute
Building 303 B, DK-2800 Kongens Lyngby, Denmark
Phone +45 45253031
compute@compute.dtu.dk
www.compute.dtu.dk M.Sc.-2013-60

Summary (English)

Recommender systems provide users with personalized suggestions for products or services. The recommendations can be based on the nature of the products or it can be based on collaborate filtering. Collaborate filtering takes the behaviour and patterns of users into account in order to make recommendations that reflects the preferences of the users, inferred by their past behaviour.

The goal of the thesis is to test collaborate filtering models which could be used as part of a recommender system for the publication portal, Issuu. To do so two models have been tested on a data set provided by Issuu. The data set contains the publications read by a user as well as the amount of time the user has spent on the publication. By applying the read time as an implicit rating of a publication the models were tested for their ability to predict how a user would rate an item.

Finally it is shown how each of the models can be adapted into a live recommender system, where the models are to be able to output recommendations for a given user. It is investigated how the models can handle new data entries as they are generated on the website, and how the models should treat entries as they age.

Summary (Danish)

Anbefalingssystemer giver brugere personlige forslag til produkter eller servicer. Anbefalingerne kan foretages på baggrund af produktets art og indhold, eller det kan basere sig på collaborate filtering. Collaborate filtering benytter brugernes opførsel og interaktion til at give anbefalinger, der reflekterer de præferencer brugerne har givet udtryk for igennem deres opførsel.

Målet med denne afhandling er at teste hvilke modeller baseret på collabote filtering, der kan bruges som en del af et anbefalingssystem til publikationsportalen Issuu. For at gøre dette er to modeller blevet testet på et datasæt, som Issuu har fremskaffet. Datasættet indeholder en liste over brugere og hvilke publikationer de har læst. For hver publikation er det angivet hvor lang tid brugeren har læst publikationen. Ved at anvende læsetiden som en implicit rating af en publikation, kunne modellerne testes på deres evne til at forudsige hvordan en bruger vil rate en publikation.

Til slut vises det hvordan modellerne kan inkorporeres i et live anbefalingssystem, hvor modeller skal være i stand til at komme med anbefalinger til en given bruger. Det er undersøgt hvordan modellerne kan håndtere nye datapunkter efterhånden som de bliver genereret, og hvordan modellerne kan håndtere datapunkter, efterhånden som de forældes.

Preface

This thesis was prepared at the department of Applied Mathematics and Computer Science at the Technical University of Denmark in fulfilment of the requirements for acquiring an M.Sc. in Digital Media Engineering. The work was carried out in the period January 2013 to July 2013.

I would like to thank Morten Arngren from Issuu for extracting the data, which made this thesis possible, and Issuu for spending time and resources on having me around.

Copenhagen, 5-July-2013

Rasmus Theodorsen

Contents

Chapter 1

Motivation

Issuu is an international digital publishing company, where anyone can upload publications and have them converted to web based online publications. The content is highly diverse with magazines, newspapers, manuals, books, advertisements and more. They have over 10 million publications, growing at about 200.000 publications per month, and they have more than 70 million monthly readers worldwide.

Publications are uploaded as pdf or text files by the publishers and transformed to a user friendly E-paper readable on computers and mobile devices. Publications can be embedded on the publisher's own website and it can be viewed on Issuu's website. Issuu's website is designed as an inspirational portal of publications, where users can browse through publications, filter by topics and get related publications. Issuu is in many ways for textual publications what YouTube is for videos. It is user generated content made easily available. You can follow publishers as they publish new publications, create favourites and create your own stacks of publications.

Issuu's business case is to make money from the publishers, while it is free for the readers to use. Publishers are given premium options to customize the reading experience of their publications and have them embedded on their own website. From the publisher's viewpoint, they are typically interested in as much exposure as possible, so the more readers Issuu can gather, the more publishers

will be inclined to use it as their platform. Consequently Issuu is interested in making the user experience as good as possible and get users to read as many publications as possible.

1.1 Current issues

The vast amount of publications available makes it impossible for users to browse through all publications. To keep users on the website it is crucial to display publications that are as interesting to the users as possible, and to do so content must be filtered so that the users see publications that matches their preferences. To help users find relevant magazines Issuu currently uses a content filtering approach based on a topic distribution model (LDA) to find similarity between publications in terms of topics. This model works well for finding publications that are similar in content, but it lacks the feature of detecting connections between topics. If a user reads about one topic it will keep suggesting publications about that topic. Some topics could however be related even though their content differs significantly and users are most likely interested in more than just one topic. Another issue is language, currently the same LDA is used for all publications, but if only a small amount of publications are published in a given language the topic distribution model can have a tendency to cluster those magazines together regardless of their content. Finally the LDA has no way of determining the quality of the content in a publication and taking that into account when recommending publications.

1.2 A new approach

An alternative approach is to harvest the huge amounts of data on user behaviour on the website and use that to create recommendations. Issuu collects large amount of user traffic data which is not currently being used for recommendations. Issuu has data about when and for how long a user has read a given publication, data that could be used for a collaborate filtering model. With over 70 million monthly readers it is more than likely that there are other users with the same preferences as any one user.

By analysing the user behaviour in general and comparing it to the active user, collaborate filtering can find preferences of users and characteristics of publications based on the patterns of other users and the read history of the active user. By finding publications that match the user's preferences, collaborate filtering can recommend publications.

A model based on collaborate filtering would pick up on interconnected topics, it would favour quality publications and it would not be limited by the different languages of the publications. A collaborate filtering model would be a good addition to the existing content filtering approach and would likely improve the quality of the recommendations in terms of relevancy for the user.

1.3 Problem statement

By comparing existing recommendation systems based on collaborate filtering, it is investigated which collaborate filtering models are best applicable to the data set available from Issuu.

Potential collaborate filtering models are analysed and evaluated on a data set provided by Issuu, in order to compare the accuracy of the predictions they make.

Prototypes of the most suited collaborative filtering algorithms are to be implemented as models that could be extended to run in a live environment based on attributes such as ability to make quality recommendations and the ability to adapt to the increasing size and growth of Issuu's data set. It should be able to deliver recommendations immediately on request and continuously evolve and improve as new data about users and publications becomes available.

Chapter 2

Related Work

Collaborate filtering and recommender systems are widely used in companies today. As more and more data is digitalised there is an increasing interest in finding meaningful patterns in the huge amounts of data found on the internet. Many online service providers have a huge amount of products which makes it next to impossible for users to get a full overview. A well functional recommender system can be an important tool in keeping users on the site and increasing sales. In the following sections, different approaches by other companies are discussed, to help make the decision as to which approaches to use for Issuu's data set.

2.1 Nearest neighbour based collaborate filtering

The first collaborate filtering algorithms that emerged were based on finding users that represented the nearest neighbours of the active user and giving recommendations based on the items rated highly by those similar users [HKBR99]. This user based collaborate filtering approach has proved successful for smaller data sets, but it does have some draw backs. First of all it can be hard to find relevant nearest neighbours, especially since the amount of ratings by a single

user can be quite limited. Secondly there will almost always be far more users than there are items, and calculating similarity between all users can therefore prove to be a very time consuming task. Because of the size of Issuu's data set, and the past experiences of others [LSY03], user based collaborate filtering will not be tested be on the data set.

Instead of looking at nearest users, a more popular approach is looking at nearest items. Item based collaborate filtering has two strong advantages over user based. Firstly items tend to have a larger amount of ratings than users, so when looking at two items it is likely that there is a larger overlap in ratings than when looking at two users. A larger overlap means that the computed similarity between two items is more accurate. Secondly since there are more users than there are items, it is more efficient to calculate similarity between all items than between all users.

2.2 Amazon

One of the most famous recommender systems is the one created by Amazon. Amazon has a wide array of different recommendation techniques, the most obvious being "Frequently bought together" and "Customers who bought this item also bought". But they also customise their entire website and the showcased products based on the user's profile, interests and purchase history. According to Amazon the click-through rate of these targeted products vastly exceeds traditional approaches like banner advertisements and top-seller lists. The recommendation engine behind this is based on item-to-item collaborate filtering where each item, that the user has shown interest in, gets associated with a number of related items which constitute the recommendations. Amazon outlines the general approach of their algorithm, but the details of the model are very sparse. They do however state that the choice of item based collaborate filtering was made mainly due to the fact that it scales well and that it can make the online recommendations very fast because a lot of computations can be made in advance. At the same time it provides good recommendations even for users with few purchases. Amazon's description of the recommendation system was written in 2003, and they have not released any subsequent articles about the subject, so it is unknown which improvements they have made since then. [LSY03]

2.3 YouTube

Google published a paper about the YouTube recommendation engine where they describe in detail how their recommendation engine works [DV10]. According to Google recommended videos account for about 60 % of all their video clicks on YouTube's website. Their approach is very similar to Amazon's, but they build a weighted, directed graph of related items, where the distance between items (the similarity) is based on associate rule learning [TSK05].

YouTube suggest different ways of calculating the similarity, but the simplest is based on the confidence association rule. For each video i they calculate the similarity to other items as the confidence that any other video j leads to watching i within the same session, equivalent of calculating the confidence

$$similarity(i, j) = conf(j \rightarrow i) = \frac{c_{ij}}{c_j}$$

Where c_{ij} is the amount of users that have watched i and j in the same session in the past 24 hours and c_j is the global popularity score of j, i.e. the amount of users that have watched j in the past 24 hours across all sessions. By dividing by the global popularity score they basically favour less popular videos over more popular ones.

When making recommendations they look at the seed videos the user has watched and collect a short list of candidates by adding all videos within a given distance on the related items graph. This can include videos which are not directly linked to the seed videos, which helps improve the diversity of the short list. The final recommendations based on the short list are made by taking into account the quality of the videos, user specificity and diversification. Like Amazon they do the heaviest computations offline, in order to allow fast recommendations to the users. This means that recommendations are not necessarily updated as the user browses the site, but they do update the relatedness graph several times a day.

YouTube and Issuu have many things in common since both websites rely on user generated content with very limited metadata, so it is likely that YouTube's approach would work well for Issuu as well.

2.4 The NetFlix Prize

In 2006 Netflix started a competition to beat their own recommendations system, Cinematch. They released a training data set of about 100 million entries

and asked the contestants to train a model that would lower the root mean square error of predicted ratings on a test set by 10 % compared to their own system. The first team to reach the goal would receive 1 million USD. The competition became very popular and resulted in more than 20.000 submissions from universities and companies throughout the world. Numerous papers have been published on that account and a lot of collaborate filtering models have been developed and tested against the data set.

In the end there was no single model that accomplished the goal of a 10 % improvement, instead the winning team used a blend of hundreds of models to win the prize. This approach is feasible on a static data set where scalability and performance is not an issue, but it is not very suitable for a dynamic, growing data set such as Issuu's.

In a blog post Netflix evaluated on the Netflix prize stating that two distinct models developed by the winning team had been implemented into Netflix' recommender system, namely a Matrix Factorisation model and a Restricted Boltzmann Machine [KBV07][Net12]. When evaluating the single individual models these two proved to deliver the best predictions on their own with Matrix Factorisation being the best of the two. Netflix reasoned that the rest of the models used in the winning submission [Kor09] constituted an improvement in accuracy not worth putting into production because of the added complexity of the solution.

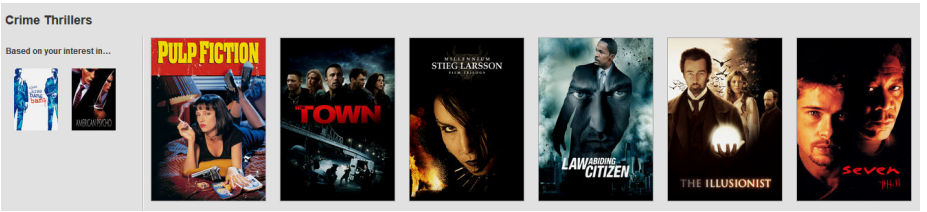

Figure 2.1: Netflix recommendations based on an inferred genre preference

Netflix has evolved a lot since the competition started moving from DVD rentals to internet streaming. This has enabled Netflix to gather a lot more information about user behaviour than just ratings and has also evolved the way recommendations are given. Netflix state that the ability to explain recommendations, as shown in figure 2.1, is an important factor in order for the users to accept the proposals and give valid feedback. The newest approach by Netflix that proves this point, is a recommendation service called Max [Net13]. Max (figure 2.2) asks questions to determine what kind of movie the user wants to see, and gives recommendations based on the answers. According to Netflix 75 % of what people are watching is chosen from some sort of recommendation.

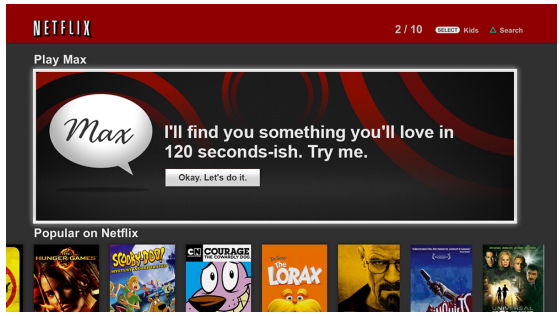

Figure 2.2: Netflix' new feature called Max

2.5 General conclusions

Based on the prior work by other large companies it is clear that a lot of work has come into designing and optimising recommendation systems and that their impact is significant. The most important attributes of recommender systems seem to be:

- Quality of recommendation, even for items with few ratings
- Scalability and run-time of model
- Ability to explain why certain items have been recommended
- Diversity in the recommendations

Item-to-item based models and models based on matrix factorisation are currently among the most popular collaborate filtering models. Item-to-item because of its good scalability and performance, and because it is easy to explain the recommendations, and matrix factorisation because it has proved to serve the most accurate predictions. Both methods are suitable for continuous values and are therefore applicable for Issuu's data set.

Based on these experiences those two methods are the ones that are implemented and tested on Issuu's data set.

Chapter 3

Data Preprocessing

Before the collaborate filtering models can be tested, the data set they rely on is subject to preprocessing in order to ensure that the data is suitable for collaborate filtering.

A data set was provided by Issuu containing user data of all registered users on Issuu. The data set contains a list of entries for each user, where each entry corresponds to a publication read by the user. The entry contains the id of the publication, the timestamp, the reading time and the amount of pages of the publication. The entries are from October 2012 to April 2013. Since Issuu currently only log usage of registered users the data set only contains a fraction of the actual traffic on the website. Visitors on the website are not forced to log in, and the advantages of creating a user have been quite small in the past. Mostly registered users get the benefit of being able to upload publications and can choose to disable the 'safe mode' to read explicit content. The data set does however contain about 2.7 million entries spread across almost 135,000 users and 900,000 publications.

3.1 Implicit feedback

The entries from the data set provided by Issuu is a result of implicit feedback. The users have not actively rated the publications, but their behaviour has been monitored and the amount of time they spent on a publication can be seen as an implicit rating of the publication. This kind of feedback is less accurate than explicit feedback, such as a rating, and more prone to noise, but it does compensate by being far more abundant.

One issue with implicit feedback is that it is geared towards positive ratings [HKV08], since having read an item will always result in a more positive rating than if the user had not read the item at all. Because of this, read time cannot be perceived as a true user rating, and there will always be the risk of recommending items based on items that the user actually did not like.

Compared to other kinds of media such as movies and television shows, the time spent is likely a much better estimate for written publications. Unlike videos and television where users might be inclined to watch the ending, even if they do not particularly like the show, or watch the next show simply because it is automatically on, written publications have to keep the users interested at all time in order to keep them reading. Unlike videos it is easy to browse through the content quickly and only stop up if anything of interest shows up, and the users always actively select the publications to read. Because of these characteristics read time is a fairly accurate measure of the users' preferences and can be seen as a reflection of the probability that the user liked the item.

3.2 Characteristics

The data set contains the read time of each item by a user, but since the total read time will rely heavily on the amount of pages in the publication, the read time will be divided by the amount of pages. Unfortunately the data set does not say anything about how many pages the user has actually read, so this approximate read time per page is not entirely accurate. The unit of the read time is supposedly in milliseconds, but looking at the data this does not seem to fit with the amount of time it takes to read. Since it has not been possible to verify the accuracy, read time will have to be treated as a unitless quantity proportional to the actual time the user has read a publication.

Figure 3.1 shows the distribution of the read times. The read time is per page. It shows how the vast majority of entries have a read time per page below

500 while there is a very steep curve of a few entries with a very high read time. This curve is a result of a weakness in the way Issuu measures read time. Unfortunately the read time keeps counting even if the user leaves the computer or has the publication open in the background. This issue will be addressed in outlier detection in section 3.3.

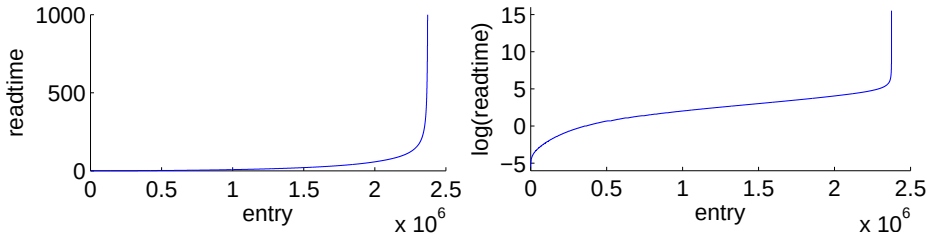

Figure 3.1: sorted read times for all data entries (left) and logarithm of read time (right)

When looking at the density function of the logarithm of the data set (figure 3.2), it appears that the read time is log-normally distributed. Log normal distributions are distributions whose logarithm is normally distributed. This happens when the distribution can be perceived as the product of a series of independent multiplicative identically-distributed variables [Wik13b].

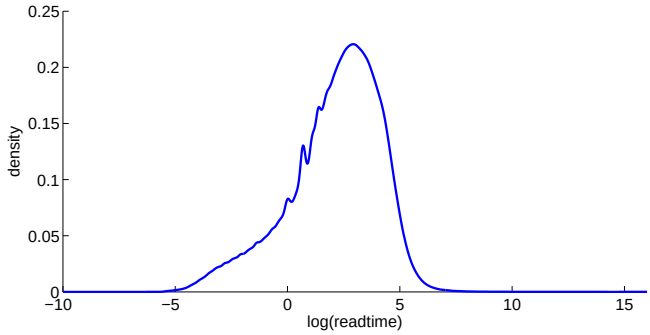

Figure 3.2: Density function of read time

To detect whether the data set is in fact log-normally distributed, the data set is plotted (figure 3.3) against a theoretical log-normal distribution with same variance (σ) and mean (μ). If the data set is in fact log normal there should be a strong linear relationship between the logarithm of the two. The figure confirms the hypothesis and shows quite strong linearity between the two distributions with minor deviations near the boundaries. Since the correlation coefficient is at 0.98, the data will be treated as a log normal distribution, despite the minor deviations. Perceptually the log normal distribution can be explained by the

read time being the product of a series of independent choices by the user as to whether to continue reading or to stop at any given moment in time.

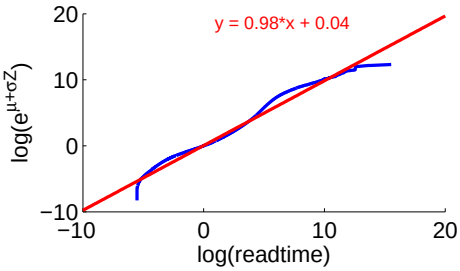

Figure 3.3: Normal probability plot

3.3 Filtering and outlier detection

With an average of only three entries per publication the data set is extremely sparse. For a lot of the lesser known magazines only the publisher has read the magazine as a registered user while the readers of the publication are all anonymous users. This means that the data set is not a true reflection of the actual user behaviour on the website and that there is likely a proportional larger amount of publishers than pure readers, than there should be. For the purpose of testing and comparing different models the quality of the data should be as high as possible and in order to achieve that, the data is going to be trimmed in a number of ways.

The first issue is the users who have left a publication open in the background resulting in very large read times. These read times are a result of a different behaviour than the rest of the data set and some are many magnitudes larger than the rest of the data. There is however no way of accurately defining when an entry is a result of background activity so the outliers will have to be marked at some threshold. Since the logarithm of the read time is normally distributed it is natural to use that for finding outliers. First the log read time (x) is mapped to a standard normal variable

$$z = \frac{x - \mu}{\sigma}$$

Where μ and σ are the average and the standard deviation of the log read times. Outliers will be defined as entries with an absolute z-value of more than 2.5758, equivalent to entries with a probability of less than 0.005. The threshold is chosen empirically because it detects most of the steep slope of very high read

times. This outlier detection will find very small read times which are likely cases where the user has not looked at the publication at all, but maybe clicked it by accident, and it will find the largest entries which are most likely a result of publications being open in the background. By removing the smallest entries it is ensured that users will not get recommendations based on items that they have hardly opened. For the largest entries it might suggest that the item is favoured by the user, since it has been kept open in the background, but the actual read time is impossible to infer and instead of guessing a value and adding more noise to the system, they will be removed as well. In the end it is up to Issuu to prevent these cases, and let the recommendations be a product of the actual read time.

A second issue is the sparsity of the data, and because of this a lot of the publications must be filtered out in order to get a data set from which meaningful information can be extracted. Collaborate filtering on magazines only read by a single user is too inaccurate and prone to overfitting. Because of this publications that have been read by less than three users will be removed from the data set.

After having removed all publications read by less than three users as well as the outliers the data set was reduced to around 1.5 million entries, 148,000 publications and 119,000 users. It is still very sparse, but it is a compromise that does not throw away too much data while still ensuring some degree of data interconnectivity.

The data set after outliers and publications with less than three readers have been removed is shown on figure 3.4.

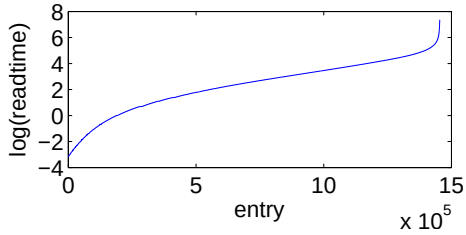

Figure 3.4: Data set after outliers and publications with less than three readers have been removed

The data input for the collaborate filtering models is crucial for the quality of the recommendations coming out. Having taken the logarithm and removed outliers from the data set resulted in a nice almost normal distributed data set. The final step before feeding the data set to the models is remapping it to new intervals. To avoid zero and negative values the data is linearly interpolated to

continuous values in the range one to ten. For future reference these new values in the range one to ten based on the read time will be referred to as ratings, while each publication will be referred to as an item.

3.4 Usage of the data set

The overall goal of this thesis is to design and test collaborate filtering models which can be used on Issuu's website. One of the most important parameters in the choice of model is the ability to precisely predict how much a user will like a given item. To approximate how much a user will like an item, the predicted read time will be used. The higher the predicted read time i.e rating, the more the model predicts that the user will like the item. The aspect of accuracy of the model will be measured by comparing the actual ratings of the user with the predicted ratings. The closer the model can get to guessing existing ratings the more accurate it is assumed to be in predicting user preferences.

The data set will be used to measure the accuracy of the models. To compare the accuracy of predictions between different models they must be able to be benchmarked against each other. This will be done by dividing the data set into a training set and a test set. The training set will account for 90% of the entries chosen randomly from a uniform distribution. Each model will be trained on the training set and will be designed to be able to predict the rating of an item, based on a user. Each model will then predict each entry in the test set and the accuracy of the model will be defined as the average root mean square error between the actual ratings from the test set and the predicted ratings. This method is the same as was successfully used in the Netflix Prize competition.

In the following two chapters, it is described how models based on item based collaborate filtering and matrix factorisation have been designed and implemented in order to test the accuracy of each of the models.

Chapter 4

Item Based Collaborate Filtering

Item based collaborate filtering (IBCF) is the method applied at Amazon and YouTube. It works by comparing similarity between items and uses that to give user recommendations based on the items the user has read. The main component of this method is a similarity matrix which contains the similarity between items. The similarity is based on the correlation in user ratings between two items. The item-item similarity matrix can be used to predict the rating of unread items by weighing the user's item vector against the similar items and based on those predictions, recommend items with the highest predicted rating. The following sections describe how item-to-item collaborate filtering was modelled and implemented with Issuu's data set.

4.1 Similarity matrix

In order to determine items that are most similar to a given item, an item-item similarity matrix is created. In its most general form it consists of a large table with all available items as both rows and columns where each entry corresponds to the similarity between the row item and the column item. However a lot of pairs of items will be so different from each other that they will have no users in

common and hence zero similarity so having all possible pairs in a table would be very inefficient. To take advantage of the sparsity an efficient approach to filling out the matrix is to find pairs of items read by the same user. Instead of calculating the similarity between all possible pairs of items, the computations can be greatly reduced by only considering items that in fact have users in common. The general approach for each item (i) is as follows.

```
For each user u who has read i
    For each other item j that u has read
        register that i and j are related
```

This results in a list of related items for each item and for each related item the similarity is calculated as described in section 4.2. Since items with a very low similarity are very unlikely to be recommended, storage space and computation time can be reduced by only storing the top N related items for each item. The size of N will influence the quality and accuracy of the recommendations, and is subject to testing in order to find a good balance between performance and quality. Since the item-item matrix is extremely sparse it is stored as a list of lists to minimize storage space.

4.2 Calculating similarity

Calculating the similarity between two items is done by extracting the user ratings of each item. Let r_i and r_j represent the vectors of user ratings of the first item (i) and the second item (j). Now the similarity between i and j can be calculated by applying a similarity measure between the two vectors as shown on figure 4.1.

Related works suggest that the similarity between two items should only be calculated from users who rated both items ignoring users who have only rated one of the two [SKKR01]. This approach is reasonable because it calculates similarity purely on similarity in user ratings and does not penalise similarity on behalf of users who have not rated one of the items. On the other hand it does not account for the amount of users the two items have in common, which is also a kind of similarity. If two items each are read by 10.000 users but only have three users in common, it might suggest that the items are not very similar. But if only the intersection is considered those three users could still result in a high similarity if the common users have provided similar ratings. YouTube uses a different approach, the rating of a viewed video is a binary 1, and they calculate the similarity between videos based on the co-visitation counts and the total visitation counts having no actual user ratings of the individual videos

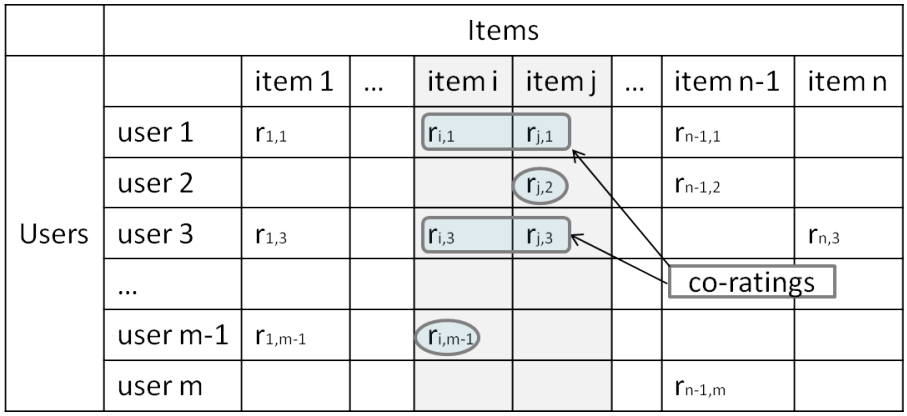

Figure 4.1: Similarity computation of item i and item j is based on co-ratings and ratings only rated by one of the users.

[DV10]. As discussed earlier watch time of a video might not be as good an approximation for a rating as read time is for written publications, so it makes sense for them to use binary ratings. For Issuu's data set it makes sense to make use of both the rating values and the co-visitation degree. To achieve this the similarity measures will not ignore cases where only one of the users has rated, instead they will infer a rating of zero for the other user, resulting in a lower similarity for items with a low co-visitation count.

The similarity between two vectors can be calculated in numerous ways, but the choices can be narrowed down based on a few preferences. First of all since the dataset is very sparse there is going to be a lot of users who have rated neither of two items resulting in a lot of 0-0 matches for each similarity calculation. The amount of 0-0 matches should not alter the similarity since the similarity should only reflect how much two items have in common and not be influenced by other users with completely different preferences. Secondly since the dataset can grow very large, computational effectiveness is also an issue, and preferably the similarity should be relatively efficient to calculate. Common approaches which display these properties and are suitable for continuous values are cosine similarity, extended Jaccard coefficient and Pearson's correlation coefficient.

The cosine similarity measure is simply the cosine angle between two sets. For non negative values the similarity is between 0 and 1. It is defined as:

$$similarity = cos(i, j) = \frac{r_i \cdot r_j}{\|r_i\|\|r_j\|} = \frac{r_i}{\|r_i\|} \cdot \frac{r_j}{\|r_j\|} \tag{4.1}$$

In cases where two sets are already normalised, the cosine similarity simplifies

to just the dot product of the two sets.

The extended Jaccard coefficient (EJ) is an extension of the Jaccard coefficient, but unlike the Jaccard coeffiecient it can be used for continuous data. The Jaccard coefficient is based on the coefficient between the intersection of two sets and the union. It is defined as:

$$similarity = EJ(i,j) = \frac{r_i \cdot r_j}{\|r_i\|^2 + \|r_j\|^2 - r_i \cdot r_j} \tag{4.2}$$

Pearson's correlation coefficient is a measure of the linear correlation between two variables. It takes a value between -1 and 1 where -1 is a perfect negative linear relationship, 1 is a perfect positive linear relationship and 0 means that there is no linear relationship. Pearson's correlation coefficient for a population is defined as

$$similarity = \rho(i,j) = \frac{cov(r_i, r_j)}{\sigma_{r_i}\sigma_{r_j}} \tag{4.3}$$

Where the covariance (cov) between two items is calculated from the co-rated ratings only, while the variance (σ) is calculated from all item ratings.

The choice of the best similarity measure is impossible to make in advance and is to be subject to testing before a preferred method can be decided upon. Test results based on the different similarity measures are shown in chapter 6

4.3 Predicted rating

The predicted rating of an item, is the model's best guess at how the user would rate the item. To make a prediction, the similarity score between the item being predicted and each of the items that the user has already read, is looked up in the item-item similarity matrix. The predicted rating of the item is an average of each of the user's existing ratings weighted by the similarity to the predicted item.

Table 4.1 shows the calculations of the predicted rating of an item based on a user's ratings of item 1 through 4. For each rated item a weight is calculated as the product between the user rating and the similarity to the predicted item. The weighted average rating, which is used as the the predicted rating, is calculated as the sum of the weights (3.9) divided by the sum of the similarities (1.1) so a predicted rating of $3.9/1.1 = 3.55$ is achieved.

Read items	User rating	Predicted item	
		Similarity	Weight
Read item 1	1	0.3	0.3
Read item 2	4	0.6	2.4
Read item 3	6	0.2	1.2
Read item 4	3	0	0
Sum		1.1	3.9
Predicted rating		3.9/1.1 = **3.55**	

Table 4.1: Calculation of the predicted rating of an item

This method is improved a little bit in the implementation to further increase the accuracy of the predicted rating. The predicted rating of the similar item is based on the rating of an existing item. Some items do however generally have a lower or higher rating than others, so by taking into account the average bias of an item compared to the global average the individual weights are adjusted to:

$$Weight(i,j) = similarity(i,j) \cdot (r_{iu} + bias_j - bias_i)$$

i being the rated item by user u and j being the predicted item.

4.4 Implementation enhancements

The item-item similarity matrix is quite time consuming to compute with an estimated run time of O(N^2M) for N items and M users. The runtime is however closer to O(NM) since an average single user has read only a fraction of the total item catalogue. The similarity matrix does however not have to be computed each time a new recommendation is given, but can be computed periodically and stored offline, so as to minimize the amount of computations required when making the actual user recommendations. The similarity matrix should however be recalculated regularly to allow recommendations of new items as they are uploaded and to adjust for the changes in similarity over time as will be discussed in section 8.1.

To further improve performance, calculation of the similarity matrix can be heavily parallelised since each item's similarity list can be computed independently. Most similarity measures also display symmetry, that is the similarity between x and y is equal to the similarity between y and x. This property can

be used to limit the amount of calculations required by storing the similarity between item x and item y in both items' similarity list in one pass.

Chapter 5

Matrix Factorisation Model

Collaborate filtering based on matrix factorisation (MFCF) has proved to be an accurate method of predicting user preferences. It was the single method that performed best on the Netflix data set and it is used in Netflix' current recommendation engine.

Matrix factorisation is a latent factor model that tries to explain users and items by modelling their inferred preferences. The matrix factorisation model works by mapping both users and items to a joint factor space of f features. The amount of features can be adjusted, but generally accuracy of predictions increase with the amount of features used, while computation time increases. The amount of features is typically in the range 10-200 [KBV08].

Matrix factorisation results in two matrices, one containing the features of each item, and one containing the features of each user. Each feature can be perceived as a certain topic or characteristic of an item. The user and item values of the feature represents how much the item possesses the characteristic and how much the user shows preference for the characteristic. The full vector of features is the distribution of preferences, and by considering this preference profile it is possible to find items which have the characteristics that the user shows preferences for and therefore should find interesting.

The inner dot product of an item and a user across all features represents the

total user-item interaction and can be perceived as the extent of the user's overall interest in the item's characteristics. In cases where the matrix factorisation model is modelling all the variation in the training data the inner product is also the predicted rating and can be calculated as follows.

$$\hat{r}_{ui} = \mathbf{q_i} \cdot \mathbf{p_u} \tag{5.1}$$

Where $\hat{r}_{ui}$ is the predicted rating of item i by user u, q_i is the feature vector of item i and p_u is the feature vector of user u.

Research does however suggest that it is preferable to isolate the variation that is not caused by user-item interaction [KBV09], this is discussed in section 5.1.

The greatest challenge of matrix factorisation is factoring a user-item ratings matrix where the majority of the entries are missing. Because of the missing entries it is not possible to apply regular techniques for matrix decomposition. Instead the missing entries are approximated by minimizing a loss function, namely minimizing the difference between the predicted ratings of the model and the actual ratings of the training set. The details of this approach are explained in section 5.2.

5.1 Item- and user bias

Much of the variation in the ratings of the data set, is typically not caused by user-item interaction, but is due to either user- or item biases. Not all users rate items in the same way. Some users have a general tendency to rate items higher because they read slower, or are just more thorough when going through a publication, while others will be more inclined to skim pages and browse quickly through content. Similarly for items, some items are generally rated higher than others. It can be items that are simply considered more interesting in general, or items with a higher density of text. Instead of having the matrix factorization account for these biases, it is favourable to remove them from the factorisation and having the matrix factorization only account for actual item-user interaction, which is the part that accounts for the actual user preferences. The bias will be defined as follows:

$$b_{ui} = \mu + b_u + b_i \tag{5.2}$$

Where b_{ui} is the total bias for item i by user u, μ is the global mean of all ratings and b_u and b_i are the user and item deviations from the global mean.

The bias will give a rough estimate of a given rating. If the global mean of all items is 5, and the user tends to rate 0.5 higher, while the item is generally scored 2 lower than average, then the baseline prediction will be $5 + 0.5 - 2 = 3.5$. Of course this rating does not say anything about whether the user actually prefers items like the one in question, but that will be accounted for by the matrix factors. The actual predicted rating will be the sum of the biases and the matrix factors extending equation 5.1 into:

$$\hat{r}_{ui} = \mu + b_u + b_i + \mathbf{q_i} \cdot \mathbf{p_u} \tag{5.3}$$

5.2 Learning algorithms

Since a lot of the entries in the user-item rating matrix are empty, it is not possible to directly calculate the matrix factors. Instead there are ways of approximating them based on the existing entries. Basically the matrix factors should be filled in such a way that they minimize the mean squared error of the difference between the predicted ratings ($\hat{r}$), and the actual ratings (r) from the set of item-user rating pairs in the training set (K):

$$S = \min_{q*p*} \sum_{(u,i)\in K} (r_{ui} - \hat{r}_{ui})^2 = \min_{q*p*} \sum_{(u,i)\in K} (r_{ui} - \mathbf{q_i} \cdot \mathbf{p_u} - b_{ui})^2 \tag{5.4}$$

Minimizing in accordance to the loss function (S) would be sufficient for a dense data set, but for users and items with only a few ratings it would lead to overfitting. It would be able to perfectly match the preferences of a user or item using only a few features resulting in a training error close to zero. The feature values would however have so much impact that unknown entries with slightly different characteristics would get a very large error. Ideally the feature values should be as small as possible, and in order to achieve that the learning algorithms must be regularised to penalize the magnitude of a feature.

There are two learning algorithms that have been commonly used to solve equation 5.4 with added regularisation, that is stochastic gradient descent [Fun06] and alternating least squares [ZWSP08]. Those two methods have been implemented and tested with Issuu's data set and the implementations are explained in the following sections.

5.2.1 Stochastic Gradient Descent

Stochastic gradient descent (SGD) is an algorithm that stepwise adjusts the individual feature values of an item and a user one feature at a time by following the gradient of the loss function. The gradients of the loss function (S) for user u, item i and feature f are:

$$\begin{aligned}(\nabla S)_{p_{uf}} &= \frac{\partial}{\partial p_{uf}}(r_{ui} - \mathbf{q_i} \cdot \mathbf{p_u} - b_{ui})^2 = -2\, q_{if}\, e_{ui} \\ (\nabla S)_{q_{if}} &= \frac{\partial}{\partial q_{if}}(r_{ui} - \mathbf{q_i} \cdot \mathbf{p_u} - b_{ui})^2 = -2\, p_{uf}\, e_{ui}\end{aligned} \tag{5.5}$$

Where e_{ui} is the training error:

$$e_{ui} = r_{ui} - b_{ui} - \mathbf{q_i} \cdot \mathbf{p_u} \tag{5.6}$$

For each (user, item, feature) triplet, the feature value of the user and the item is adjusted by subtracting a factor of the gradient:

$$\begin{aligned}p_{uf} &:= p_{uf} - \alpha\,(\nabla S)_{p_{uf}} = p_{uf} + \alpha(q_{if}\, e_{ui}) \\ q_{if} &:= q_{if} - \alpha\,(\nabla S)_{q_{if}} = q_{if} + \alpha(p_{uf}\, e_{ui})\end{aligned} \tag{5.7}$$

Where α is the learning rate. The learning rate will be chosen based on testing, but is in the order 0.001.

Now in order to limit the magnitude of each feature a regularisation term is added to the loss function.

$$\min_{q*p*} \sum_{(u,i)\in K} e_{ui} + \lambda(\|q_i\|^2 + \|p_u\|^2) \tag{5.8}$$

The regularisation terms consist of λ, which controls the extent of the regularisation, and the norm of q_i and p_f. By adding the norm to the loss function, a solution than minimises the impact of each feature is encouraged. Based on

the regularised loss function the adjustments to the user and item feature values changes to:

$$\begin{aligned} p_{uf} &:= p_{uf} + \alpha(e_{ui} \cdot q_{if} - \lambda\, p_{uf}) \\ q_{if} &:= q_{if} + \alpha(e_{ui} \cdot p_{uf} - \lambda\, q_{if}) \end{aligned} \tag{5.9}$$

Since the regularisation term grows with the magnitude of the feature value, the adjustments to its value will be contained. The value of λ is variable, though a value of 0.002 has been suggested for the Netflix data set by related research [Fun06]. Different λ values will be tested (section 6), to find an optimal value.

Flow of the algorithm

1. The two feature matrices are initialised to small random non-zero numbers.
2. The algorithm loops through every single known rating, calculates the gradients and adjusts the feature values of the current feature in accordance with equation 5.9, in order to fit the feature matrices to minimise the regularised loss function.
3. Step 2 is repeated until the the average squared error is no longer improving or until it has passed a finite number of iterations.
4. Step 2 and 3 is repeated for each feature.

5.2.2 Alternating Least Squares

Alternating least squares (ALS) is an alternative approach to solving equation 5.4. It works by alternating between solving p_u for each user and q_i for each item. By fixating one of the two feature matrices, the problem can be seen as a matter of approximately solving an overdetermined system of linear equations, which is a linear least square problem that can be solved optimally.

Solving the least squares problem

Finding the optimal solution for the linear least square problem is a matter of finding the point where the gradient of the loss function is zero. In the case

where the item feature vector (q_i) is kept constant, the equation for finding the minimum of the user feature vector (p_u) of user u becomes the following:

$$\frac{\partial}{\partial p_u} \sum_{i \in K_u} (r_{ui} - \mathbf{q_i} \cdot \mathbf{p_u} - b_{ui})^2 = 0 \tag{5.10}$$

Where K_u is the set of items rated by user u. From equation 5.10 the normal function can be derived [Wik13a]:

$$Q_{K_u} Q_{K_u}^T p_u = Q_{I_u}(r_{K_u} - b_{K_u}) \tag{5.11}$$

Where Q_{K_u} is a matrix containing the item feature vector of each of the items rated by the user, r_{K_u} is the vector containing each rating by the user and b_{K_u} is the associated bias of each rating. From the normal equation (5.11) p_u can be solved by solving the linear matrix equation.

The normal equation for finding the item feature vector is derived in the same manner, where the loss function is differentiated with regards to q_i instead of p_u.

As well as for the stochastic gradient descent algorithm, this solution should also be regularised. This is done by adding a Tikhonov regularisation term to the normal equation.

$$(Q_{K_u} Q_{K_u}^T + \Gamma\Gamma^T) p_u = Q_{K_u}(r_{K_u} - b_{K_u}) \tag{5.12}$$

To achieve the same kind of regularisation as for stochastic gradient descent, the Tikhonov matrix will be defined as $\Gamma = \lambda I$, I being the identity matrix with size f (amount of features), and λ being the degree of regularisation. Besides regularisation the solution it also has the side effect of improving the conditioning of the problem, and it helps avoid issues in the unlikely case that Q_{K_u} is singular.

Flow of the algorithm

1. The user feature matrix (P) is initiated to random small numbers
2. P is kept constant while the item feature matrix (Q) is solved one item at a time.

3. Now Q is kept constant while P is solved one user at a time.
4. Alternating between solving P and Q is continued until the average squared error is no longer improving or until a finite number of iterations has passed.

5.3 Learning algorithms comparison

The advantage of ALS compared to SGD is that each row in the feature matrices can be calculated independently. So when updating the user feature matrix, it can be done in parallel by dividing the users into a number of subsets and calculating the user features for each user in the subset on a separate processor. On the other hand stochastic gradient descent is a simpler algorithm, and based on the research done on the Netflix Prize, it gives slightly better results [TJ09]. Both algorithms will be tested on the data set.

Chapter 6

Validation

Two models have been implemented in the Python programming language, and have been tested with Issuu's data set. The matrix factorisation model has been tested with the stochastic gradient descent and the alternating least squares algorithms each with several different amounts of features. The item based collaborate filtering model has been tested with different similarity measures. For comparison, a naive predictor that always predicts the average rating of the predicted item will be used.

6.1 Calibrating MFCB

To test the accuracy of the matrix factorisation models, the first step is to find the optimal values of the regularisation factor (λ). For the Netflix dataset the optimal value ranges from 0.02 to 0.08 [ZWSP08]. The Netflix data set is however far denser than Issuu's with an average of over 5000 ratings per movie compared to Issuu's, that has roughly around 10. The extra sparsity of the data means that features should be regularised even more, since generalising characteristics on behalf of very little data will easily lead to overfitting.

Figure 6.1 shows the resulting test errors (RMSE) from training the ALS algorithm with different lambda values. With ten features the optimal λ value

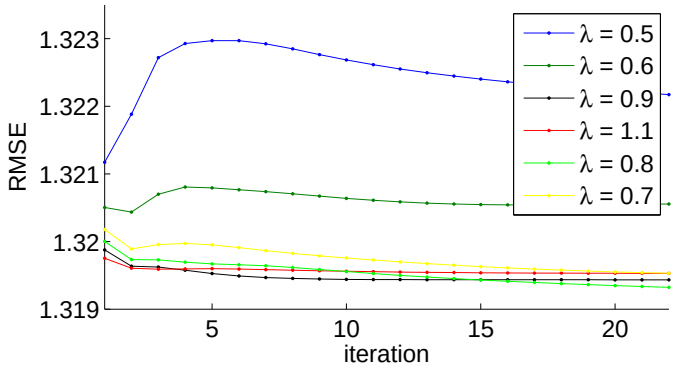

Figure 6.1: RMSE test scores of different regularisation values, using the ALS algorithm with ten features

is around 0.8, while lower values seem to overfit the data. Even though the optimal λ value can change a little bit with the amount of features, 0.8 will be the value used in the model comparison tests.

A similar test was run for the stochastic gradient descent algorithm (figure 6.2). It shows the resulting test errors from training the algorithm with five features over ten iterations. For stochastic gradient descent the optimal regularisation value appeared at 0.2, which is the value that will be used for comparison. The tests were run with a learning rate of 0.05.

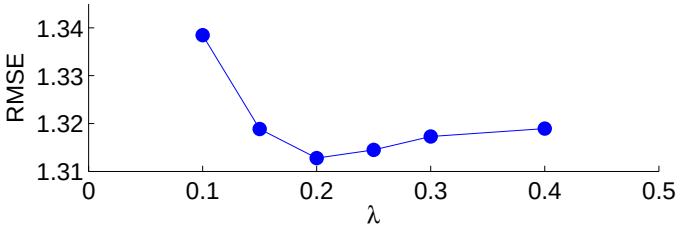

Figure 6.2: RMSE test scores of different regularisation values, using the stochastic gradient descent algorithm with 5 features and 10 iterations

6.2 Test results

Each model has been trained using the training set, and the root mean square errors on the test set are shown on figure 6.3. The IBCF predictors can only predict ratings of items that are in the top similarity list of at least one of the user's rated items, so for these predictors only about 20% of the test set could be predicted.

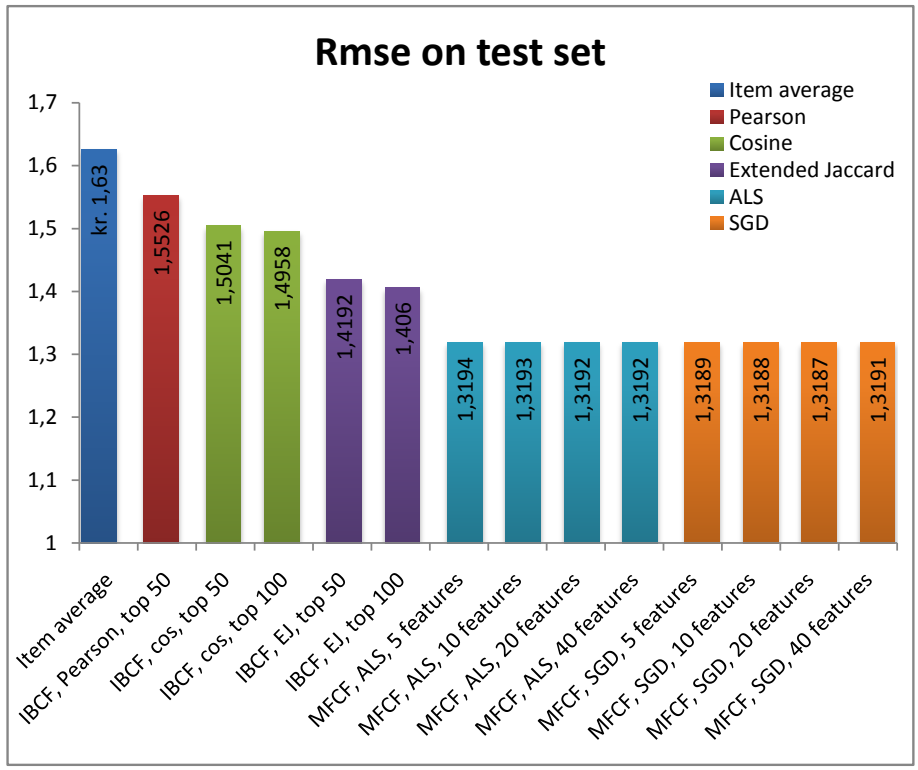

Figure 6.3: RMSE test scores of each of the implemented models with varying parameters

As the figure shows, the naive predictor, that predicts the average of an item's ratings every time, results in the highest error. After that comes the IBCF models, where the extended Jaccard coefficient performs best. Furthermore the accuracy increases a little but when increasing the amount of similar items in the similarity matrix from 50 to 100. The best results come from the MFCF models, where SGD is slightly more accurate than ALS. Generally the accuracy increases a little bit with the amount of features. The best result came from

running SGD with 20 features, while the regularisation value may not be optimal for 40 features, since it scores a slightly higher error.

6.3 Conclusions

In conclusion, the results from the Netflix Prize are confirmed in the way that MFCF results in the most accurate predictions beating the best IBCF algorithm with about 6%. Both models beat the naive predictor.

Unfortunately the accuracy of the MFCF predictors does not improve much with the amount of features, or the amount of iterations (figure 6.1), which suggests that the data set does not contain a lot of user-item characteristics. Most of the variation comes from either user or item biases. Some users just generally read for a longer period of time, and some items are generally read for a longer period of time, while the patterns of some users preferring some kinds of items, and other users preferring other kinds, are small. This is unfortunate since those patterns are exactly the ones that collaborate filtering should use to make quality recommendations.

It is not possible to find patterns that are not there, so if the characteristics are not evident in the data set, they will not be evident in the models. This does not mean that the models do not work, they show good properties like not overfitting the data, and actually finding some degree of user-item patterns.

The data set from Issuu has a density of about 0.008% compared to about 1% for the Netflix data set. When taking that into account as well as the fact that the ratings are based on inhomogeneous data from implicit feedback, compared to Netflix' well defined 1-5 explicit ratings, it is not so surprising that the models struggle to find patterns. There are however several things Issuu could do to improve the results:

- Registering read time from anonymous users would increase the density of the data set tremendously, since anonymous users account for most of the traffic on Issuu's website.
- Fix the issue of read time counting even if the user has the publication open in the background.
- When calculating the read time per page, only pages that have actually been read should be taken into account.

When considering the fact that the rating values are mostly explained by user and item biases, it can be questioned whether the RMSE score can in fact be used as a good way to measure the accuracy of the models' abilities to detect user preferences. That the reason MFCF scores a lower RMSE is simply because it is better at modelling biases. The relevancy of the recommendations made by IBCF might still be better because they are made from items that are directly linked to items read by the user, while MFCF might have a tendency to recommend the items that have the highest item bias to all users. Nonetheless the RMSE score has been shown to be a good reflection of user preference on the denser Netflix data set, even though other measures have also been suggested [Kor10].

Chapter 7

Making Recommendations

So far it has been described how each of the implemented models (IBCF and MFCF) are capable of predicting ratings. As described earlier it is assumed that users will prefer items with the highest predicted ratings, so by looking at the predicted ratings of the models it is possible to make recommendations. The way recommendations are given, and how they can be interpreted is however a bit different for the two models.

7.1 IBCF

The model can recommend items based on a number of seed items. The seed items are the items that that will be used to find related items. Each seed item is looked up in the similarity matrix to find the top similar items and calculate their predicted rating.

7.1.1 Profile based recommendations

One way of giving recommendations is based on the user's whole read history. These profile recommendations are often used on the front page of a website

where the user has still not selected anything and it gives a broad range of recommendations, using all items previously read by the user as a seed set.

As the user begins to read items in a user session, recommendations can be narrowed down a little bit by only considering items read in the current session as the seed set. This way recommendations can be given that suits the current interests of the user.

The recommended set is found by looking at the similar items of each of the items in the seed set. The total set of similar items are the possible candidates for recommendation. For each similar item the predicted rating is calculated and the similar items with the highest predicted ratings are given as recommendations.

7.1.2 Item based recommendations

A different use case is recommending items based on a single item. This item recommendation gives a more narrow set of recommendations and is suitable in cases where the user is reading a specific item and a recommendation based solely on that item is requested, as an alternative to the list of recommendations specific for that user. This makes it possible to give recommendations based on the current context of the user. In this use case only the single item is used as a seed item.

Finding the list of recommendations for a given item is simply a lookup in the item-item similarity matrix and returning the top most similar items for the chosen item. Since the seed set only consists of one item it is not necessary to calculate the predicted ratings. This kind of recommendation does not take the user's general preferences into account, other than his interest in the given item.

7.2 MFCF

The MFCF model makes recommendations by looking at all items not already read by the user. This means that for each user, the predicted rating of all items should be calculated and sorted before returning the recommended set of items with the highest predicted rating. Calculating the predicted rating is however relatively fast. It is not necessary to calculate the exact predicted rating, since only the order is relevant for making a recommended set. That

means the calculations simplify to:

$$rating = b_i + \mathbf{q_i} \cdot \mathbf{p_u} \tag{7.1}$$

The rating is calculated from the item and user feature vectors and the item bias. Since the global average and the user bias is identical for all items for a given user, they do not have to be included in the rating. Furthermore these lists can be computed in advance and cached, so that the calculations do not have to be performed while the user is on the website.

The recommended set will always be a recommendation based on all the user's ratings, that were used to train the model, since all the user's ratings are incorporated in the factor matrices. That means that it is not possible to give recommendations based on a current user session or a single item.

It also means that items read after MFCF has been trained are not taken into account when giving recommendations, so the recommendations will not change while the user is reading publications online. For the recommendations to change during a user session the factor matrices would have to be updated while the user reads publications.

Since MFCF can not be expected to be updated while a user is browsing the website, the recommendations given would work best as an initial presentation to give the user when he enters the website.

7.3 Comparison

IBCF has the advantage of being more versatile than MFCF, in the way that the items used to make recommendations can be changed independently of the model, which makes it possible to change the recommendations as the user browses the website. This also means that it can quickly begin making recommendations for new users as soon as they have read a couple of items.

While MFCF is not as adapt as IBCF, it does have the advantage over IBCF that it can give recommendations that are not necessarily directly linked with the user's rated items, but recommended based on latent factors. MFCF has also proved more accurate in guessing predicted ratings than IBCF, so the relevancy of the recommendations is assumed to be higher than those of IBCF.

In terms of explaining the recommendations, IBCF is simple in the way that it is

always possible to detect which items were the reason for the recommendations. Each recommended item is directly linked to one or more of the user's existing ratings, so it is possible to tell the user what items were the reason for the recommendation of an item. For MFCF it is a bit more complicated, because you would have to look at the feature vectors and try to explain the characteristics they cover, and then explain which rated items show the same characteristics as the recommended items.

With regards to computational run time, IBCF has the advantage that top lists of most similar items are computed while training the model, so not all items have to be considered for recommendation. Furthermore IBCF is a simpler algorithm which is generally faster to train than MFCF.

Chapter 8

Improving Recommendation Quality

The models described in this report are designed to be able to predict ratings so that they can be used for benchmarking. That means that the recommendations they give are based solely on user ratings. The insight into other commercial recommender systems do however suggest that it is preferable to take other parameters into account before passing the recommendations to the users.

8.1 Adapting to a dynamic data set

The implemented models are suitable for testing, but they are only designed to handle a static data set. In order for the model to be used on a live website it must be able to constantly update the recommendations as new publications are uploaded and read, and it must diminish the effects of old ratings to keep recommendations up to date.

The models implemented treats the whole data set equally. This means that old entries are given the same weight as new entries. This makes sense for testing since the test set contains both old and new entries which all are supposed to simulate new unknown entries. In practice the popularity and relevance of a

publication will decrease over time and a user's interests might also shift. New publications will often be new versions of the same magazine and often users will be interested in the latest issues of these.

To account for this, the impact of user ratings should decrease over time so that newer entries will have more influence over recommendations than older entries. This will mean that recommendations are mostly based on a user's recent ratings so that users will not keep getting recommendations based on publications read a long time ago, and recommendations are going to consist of newer publications if the general tendency of the user is to favour newer publications.

8.1.1 IBCF

There a different ways to account for the dynamics in the data set. YouTube [DV10] have taken the approach of only looking at ratings from the last 24 hours when calculating the similarity matrix. Since they have a huge amount of user traffic, it is adequate for them to disregard anything older than that.

An alternative approach could be to calculate the similarity matrix for the past 24 hours and then define the new similarity matrix as a weighted average of the existing similarity matrix and the one for the past 24 hours. The weight ratio between the existing and the 24 hour similarity matrix as well as the interval between updating the similarity matrix would then control how fast old ratings diminish on account of new ones. A clear advantage of this approach is that only ratings from the past 24 hours are needed when calculating the costly similarity matrix, while the existing ratings are still taken into account in the new similarity matrix.

8.1.2 MFCB

Each time new ratings are added to the data set, the factor matrices will become outdated so in order to keep recommendations relevant the matrix factorisation algorithm should be run often. When updating the model, the feature matrices will be initiated at their old values, so the algorithm should be relatively fast at descending on new values.

Handling the temporal effects on recommendations is not easily done inside the model. It would be possible to use the same approach as Youtube and calculate the model based only on the past 24 or so hours, but it could be the case the data from that time period is insufficient to create a good matrix factorisation

model. Alternatively the output from MFCF can be treated as a gross list, which is then post processed and weighted based on the amount of recent ratings.

8.2 The cold start problem

Collaborate filtering suffers from what is known as the cold start problem. When an item is first added to the system it is not going to have any ratings, and therefore it is not possible to deduct any characteristics or related items, which means that it will not be included in any recommendations. The same problem occurs for a user that has not previously read any publications. The system will not know the user's preferences and can not make any intelligent recommendations. These issues are why collaborate filtering should be accompanied by other means of making recommendations. Alternatives for new users could be to recommend a list of the overall most popular publications or to have staff put together lists of recommended publications. For new items the solution could be to include them in random recommendations, to see how users behave towards them, until the items have gained some ratings.

8.3 Post-processing

By treating the output of the collaborate filtering models as a list of candidates that recommendations can be chosen from it is possible to take more features into account before making the final set of recommendations. The predicted rating is one of the features, but several others are also relevant. Language of the recommended publications is a feature that should be taken into account since users should not be recommended publications in languages that they do not understand. Diversity is another aspect that should be considered. To account for this the collaborate filtering model could be combined with the topic model already implemented by Issuu to ensure diversity in topics throughout the recommendations, by picking the publications that result in the greatest overall distance in the topic space.

Chapter 9

Conclusion

By looking at how collaborate filtering has been used and implemented in companies like Amazon, YouTube and Netflix it was established that the two of the most successful collaborate filtering models are currently item-to-item collaborate filtering models (IBCF) and models based on matrix factorisation (MFCF). These two models were also applicable to the data set provided by Issuu, so they were chosen for implementation and testing.

The accuracy in predicting unknown ratings, was tested for both of the models and it was discovered that MFCF gave the most accurate predictions with a lead of about 6 % in RMSE compared to IBCF. From the assumption that the RMSE error is a measure of how good a model is in guessing user preferences MFCF should be capable of giving recommendations that best suit the users' preferences. It was however shown that the RMSE score has some weaknesses and it could be beneficial to complement it by other means of measuring the models' abilities to detect user preferences.

Both models are capable of being implemented in a live environment and can be designed to be able to absorb new ratings and account for temporal dynamics in the data set. Furthermore it has been shown how each of the models can make recommendations based on the predicted ratings.

The two models have some differences in the way recommendations are given

and how they can be interpreted, and in that regard IBCF was shown to have a number of advantages over MFCF. The recommendations made by IBCF can be easily interpreted, since they can always be tracked back to one or more items that the user has read, while MFCF recommends items based on latent factors that can be hard to explain. The ability to explain recommendations has proved to be an important factor in the recommendation systems implemented by other companies. IBCF is capable of making recommendations based on an arbitrary set of items without changing the model, which means that recommendations can change based on what the user is currently reading. This is a clear advantage over MFCF which always gives recommendations based on all the ratings used to train the model.

Since it has not been possible to definitively prove that MFCF gives the best recommendations, there are a lot of parameters that point at IBCF as the most suitable model for Issuu's data set. It would however be beneficial to be able to test the models on a denser data set and experiment with other ways of measuring the accuracy of the models.

Bibliography

[DV10] Nandy Davidson, Liebald and Van Vleet. The youtube video recommendation system. In *RecSys '10 Proceedings*, 2010.

[Fun06] Simon Funk. Netflix update: Try this at home. `http://sifter.org/~simon/journal/20061211.html`, dec 2006.

[HKBR99] Jonathan L. Herlocker, Joseph A. Konstan, Al Borchers, and John Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[HKV08] Yihan Hu, Yehuda Koren, and Chris Violinsky. Collaborative filtering for implicit feedback datasets. In *ICDM '08 Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008.

[KBV07] Yehuda Koren, Robert Bell, and Chris Volinsky. The bellkor solution to the netflix prize. Technical report, AT&T Labs, 2007.

[KBV08] Yehuda Koren, Robert Bell, and Chris Volinsky. The bellkor 2008 solution to the netflix prize. Technical report, AT&T Labs, 2008.

[KBV09] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, August 2009.

[Kor09] Yehuda Koren. The bellkor solution to the netflix grand prize. Technical report, AT&T Labs, 2009.

[Kor10] Yehuda Koren. Factor in the neighbors: Scalable and accurate collaborative filtering. *ACM Trans. Knowl. Discov. Data*, 4(1):1:1–1:24, January 2010.

[LSY03] Greg Linden, Brent Smith, and Jeremy York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7(1):76–80, January 2003.

[Net12] Netflix. Netflix recommendations: Beyond the 5 stars (part 1). `http://techblog.netflix.com/2012/04/netflix-recommendations-beyond-5-stars.html`, apr 2012.

[Net13] Netflix. Let "max" be your netflix guide on ps3. `http://blog.netflix.com/2013/06/let-max-be-your-netflix-guide-on-ps3.html`, jun 2013.

[Seg07] Toby Segaran. *Programming collective intelligence*. O'Reilly, first edition, 2007.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW '01 Proceedings of the 10th international conference on World Wide Web*, 2001.

[TJ09] Andreas Töscher and Michael Jahrer. The bigchaos solution to the netix grand prize. Technical report, commendo research & consulting, 2009.

[TSK05] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[Wik13a] Wikipedia. Linear least squares (mathematics). `http://en.wikipedia.org/wiki/Linear_least_squares_(mathematics)#Derivation_of_the_normal_equations`, jun 2013.

[Wik13b] Wikipedia. Log-normal distribution. `http://en.wikipedia.org/wiki/Log-normal_distribution`, jun 2013.

[ZWSP08] Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, and Rong Pan. Large-scale parallel collaborative filtering for the netflix prize. In *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, AAIM '08, pages 337–348, Berlin, Heidelberg, 2008. Springer-Verlag.