Erasmus School of Economics

Bachelor's Thesis Business Analytics and Quantitative Marketing

Neighborhood-based Collaborative Filtering: Providing the best recommendations

Eveline Mathol
supervised by Gertjan van den Burg

Abstract

In order to improve customer experience companies can invest in recommendation systems. Recommendation systems attempt to profile user preferences and provide users with good personalized recommendations. We focus on the neighborhood-based methods for Collaborative Filtering. Collaborative Filtering only relies on past user behavior. Neighborhood-based methods select the most similar items or users (neighbors) and make predictions by determining interpolation weights for these neighbors. We implement the methods proposed by Bell and Koren in [3]. Bell and Koren proposed an improved neighborhood-based Collaborative Filtering method, which addresses some of the issues of previous neighborhood-based methods. This improved method consists of three main components, namely data normalization, the selection of neighbors and the determination of interpolation weights. This method was evaluated on the Netflix prize data [4]. The inclusion of global effects alone gives us a RMSE of 0.9658. If we perform no data normalization the root mean squared error (RMSE) for Bell and Koren's neighborhood interpolation method is worse, around 0.975. For Double Centering, the RMSE improves with approximately 0.05. We find the best RMSE by applying full data normalization, which resulted in a RMSE of 0.9194. We extend the methods by Bell and Koren by performing further data normalization. We remove the Genre effect and some temporal effects, but this does not improve the RMSE much. Only the Genre effect lowers the RMSE for the inclusion of global effects to 0.9655.

June 30, 2016

Contents

1 Introduction

The growing popularity of e-commerce has increased the interest in recommendation systems. The movement to e-commerce has provided customers with a greater variety of products. This information overload has made it more difficult for customers to select products. For companies, it is not easy to match customers to products. Recommendation systems help customers with this selection by providing personalized suggestions of products. Over the years, recommendation systems have changed the way customers behave. Examples of applications of recommendation systems include books, movies, restaurants, fashion, music and webpages.

Recommendation systems consist of programs and algorithms that analyse patterns of user behavior. These systems attempt to profile user preferences and model the user-item relationships. Good personalized recommendations can improve the overall customer experience and therefore lead to retaining customers and attracting new customers. This economic potential led some of the big e-commerce websites, like Netflix, Amazon, and Google, to invest in recommendation systems. In 2006, Netflix even launched a competition to enhance its recommendation system. The winner of the competition had to outperform Netflix′s recommendation system and received an award of $1,000,000 [4]. This competition has stimulated many researchers to improve and innovate algorithms for recommendation systems.

Typically, recommendation systems are based on two different strategies, Collaborative Filtering and Content-based Filtering. The difference between those two is that Content-based Filtering is based on the profiling of users or items through the use of descriptive labels. Collaborative Filtering (CF) only relies on past user behavior. The most successful approaches to CF are neighborhood-based methods and latent factor models. In this paper, we will focus on neighborhood-based CF. Since the dataset does not contain any descriptive labels needed for Content-based Filtering, we will be focusing on CF. The advantages of the neighborhood-based approach for CF are its intuitiveness, its easy implementation and its ability to easily explain the user the reasoning behind a recommendation [3].

The objective of this research is to predict user preferences for items by implementing and enhancing the neighborhood-based CF approach. The dataset contains ratings of the customers of Netflix, a global media provider. This is the same dataset used for the competition of Netflix. To examine the accuracy of the predictions of the ratings, we measure their RMSE. We compare the RMSEs of various stages of preprocessing against varying sizes of neighbors to evaluate the performance of the model. Additionally, we will extend the implemented neighborhood-based CF approach by Bell and Koren by performing further data normalization steps.

The research question that will guide this research reads as follows:

> How can we implement the neighborhood-based Collaborative Filtering approach by Bell and Koren [3] to predict user preferences for items and extend this approach by performing further data normalization?

In the next section of this paper we analyse the dataset and we focus on its important features. After that, we describe the methods for the three components of the neighborhood-based approach in detail. In section 4 we extend this approach by incorporating temporal effects and genre effects during data normalization. This is followed by an evaluation of the results of the implemented methods. We will conclude with a discussion of our results and several issues that arise from these results.

1.1 Literature Review

In this section we will give an overview of the existing literature on Collaborative Filtering methods. A wide variety of algorithms has been developed for the purpose of recommending items to users. We will focus on the differences between the algorithms, and discuss their advantages and disadvantages.

During the early 1990s, with the growing popularity of e-commerce, the first works in the field of recommendation systems were published. Since then, recommendation systems became an independent research area of machine learning. These recommendation systems enabled companies to give customers good personalized recommendations and improve customer experience, therefore many companies started to adapt recommendation systems. Recommendation systems are usually classified in the following three categories, based on their recommendation techniques [2]:

- Content-based
- Collaborative Filtering
- Hybrid, a combination of Content-based and Collaborative Filtering methods

Since our paper focuses on Collaborative Filtering we will only be discussing existing literature on this approach for recommendation systems.

The earliest CF system was the Tapestry system [10]. With this Tapestry system users could annotate electronic messages and other users could use these annotations to filter messages. The Tapestry system enabled users to collaborate to filter messages. GroupLens [13, 17], Ringo [21] and Bellcore's Video Recommender [12] were the first Automated Collaborative Filtering systems. Instead of the user needing to find like-wised users manually, the user was now automatically matched to users with similar opinions.

According to [5] CF algorithms can be divided into two main categories, memory-based and model-based algorithms. Memory-based algorithms use the user-item ratings to directly predict new ratings for items. This is done by calculating similarities. This approach can be user-oriented or item-oriented. For the user-oriented approach, unknown ratings of an item are predicted based on the weighted average of all the ratings of similar users on the item. For the item-oriented approach, unknown ratings of an item are predicted based on the weighted-average of all the ratings of the user on similar items. The similar users or items are called neighbors and they are selected by using the similarities. According to [20] the item-based algorithms provide better performance and more efficient computations. Most simply, the aggregation is a weighted average. However, this approach does not take into account that users use a different rating scale. In [20] the authors propose to deduct the mean rating of the user to overcome this problem. Other researchers proposed preference-based filtering, where similarities are calculated based on relative ratings. The most common choices for the similarity measure are the Pearson correlation coefficient [17] and the cosine similarity [5]. Others have also used the mean squared difference between users or items as similarity measure [21].

Most early CF system used memory-based algorithms. The advantages of the memory-based algorithm are its simplicity (intuitiveness), its justifiability (users can easily be explained why they received a certain recommendation), its efficiency (no costly training phases) and its stability [7]. However, the memory-based methods also has its shortcomings. When data is sparse, the similarities between users or items are based on a small amount of neighbors. This will result in skewed similarities. This can happen when there is a cold start problem, new items have not received many ratings yet and new users have not made many ratings yet. Also, the memory-based methods grow in size with the number of items or users, so there are scalability issues.

To overcome the shortcomings of memory-based algorithms, researchers started to investigate model-based algorithms. These machine learning or data mining algorithms use the user-item ratings to estimate or learn a model used to predict new ratings for items. The most popular

model-based CF models are Bayesian network models, clustering models and factorization based models. Matrix factorization techniques have been successful at addressing the sparsity and scalability issues of CF systems. These factorization based systems assume that data can be explained by latent features and leave out less significant features of the data. These systems have as advantage that dimensionality of the recommendation systems databases can be decreased. The most important matrix factorization techniques are the Singular Value Decomposition (SVD) [19], Principal Component Analysis [11] and Probabilistic matrix factorization [18].

Common clustering models cluster similar users or items into segments based on the rating data. Then they estimate the probability that a user or an item is in a specific segment and compute the conditional probability of a rating. In Bayesian network models, items are represented with nodes in a Bayesian network. In [5] the authors have proven that this model is small, fast and as accurate as neighborhood-methods. Model-based algorithms are better at dealing with sparsity than memory-based algorithms. Also, model-based algorithms can better address scalability issues. The issues with model-based algorithms are that they require expensive model building, that there is a trade-off between prediction performance and scalability and that with the reduction of the models useful information can be lost.

The Netflix prize competition launched in 2006 stimulated many researchers to improve and innovate algorithms for the recommendation systems. Especially since then, researchers started to combine memory-based and model-based algorithms. By combining the algorithms it is possible to overcome the shortcomings of the individual models. We will give an overview of the best and most influential methods used during the Netflix prize competition.

Bell and Koren found out that the removal of global effects can improve the estimation accuracy [3]. Most models work best when global effects have been accounted for. Bell and Koren also proposed the combination of matrix factorization and neighborhood-based methods in this paper. This combination leads to very good predictions. The top teams mostly formed variations on SVD models or blended models to gain higher estimation accuracy. Simon Funk was the first to reveal the use of SVD models using gradient descent [8]. Another important matrix factorization model was the SVD++ model. This asymmetric model incorporates both implicit and explicit feedback [14]. Also, Koren showed that in factorization models and in neighborhood models the inclusion of temporal dynamics is very useful in improving quality of predictions [16]. In the end the Netflix prize was won by the team 'BellKor's Prachmatic Chaos' by blending a number of SVD++ model with Restricted Boltzmann machines [15]. Combinations of different methods have proven to give the best results for Collaborative Filtering approaches.

2 Data

In this section, we describe the content of the dataset that served as the basis for this paper. We will analyse the data and evaluate important features graphically. Furthermore, we elaborate on some important notations used throughout the paper.

The dataset contains ratings of customers at the global media provider Netflix, offering a wide assortment of movies and series. The rating is the number of stars given to a movie by a customer, ranging from 1 (not interesting) to 5 (very interesting). There are over 100 million ratings in the dataset, collected between October 1998 and December 2005. The dataset reflects the distribution of all received ratings by Netflix during this period [4]. The dataset includes 480,189 anonymous customers, each identified with a unique customer ID. There are 17,770 movies in the dataset, each identified with a unique movie ID. Furthermore, there is also information on the title and release year of the movie and the date of the rating. The data was released in a training-test set format. Therefore, the data has been partitioned in two parts: a Training set and a Hold-out set. The Hold-out set was created by the nine most recent ratings of the customers (or less if the customer did not rate at least 18 movies over the examined period) and contains around 4.2 million ratings. The remaining data formed the Training set. The Hold-out set is randomly split into subsets called

Probe, Quiz and Test, with the same statistical properties. The goal of the Netflix prize contest was to predict ratings for the Quiz and Test set, also known as the Qualifying set. The ratings for the Hold-out set are harder to predict, they contain more ratings of users that do not rate much. The true ratings for the Probe set are given, for the Qualifying set the ratings are kept by Netflix to evaluate the performance for the contest. The Probe set is attached to the Training set and its main purpose is to evaluate the performance of algorithms by participants of the contest. Because we can only evaluate the performance of our algorithm for the Probe set, we will only be reporting results on this set.

The following three figures give more insights in the data of the Training set (including Probe). From Figure 1 and Figure 2, we can see that the distribution for the ratings per user and the distribution for the ratings per movie are both very skewed. The average number of ratings per movie is 5654.5, the average number of ratings per user is 209.252. Most movies were rated under 1000 times while Miss Congeniality, the most rated movie, was rated by almost half the users in the dataset. Likewise for the users, a few users rated over 10,000 movies, but a quarter of the users rated under 36 movies. This wide variation makes it more difficult to estimate movie and user parameters, but this is a typical characteristic for collaborative filtering data. From Figure 3 we can see an unexpected peak in the number of ratings on January 19 2005. Perhaps this peak has to do with the introduction of profiles for Netflix accounts on January 18. With the profiles family members have their own sign-in name, ratings and recommendations [1]. Further we would like to comment that the data is very sparse. About 99% of the possible ratings are unknown.

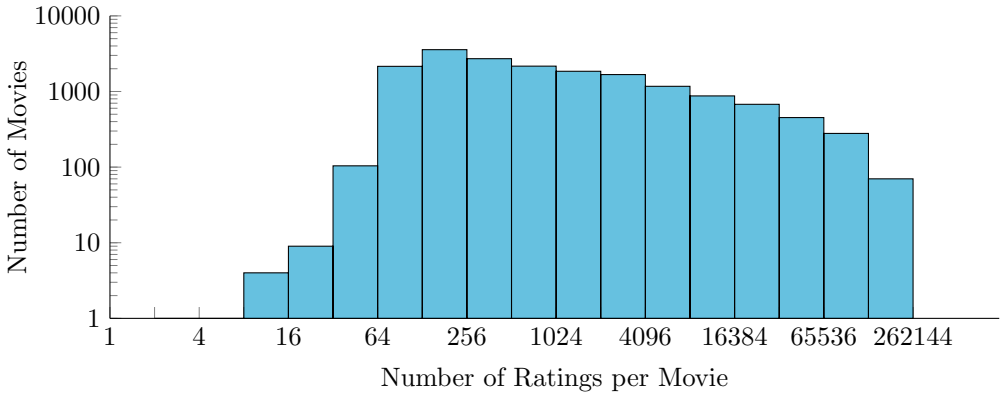

Figure 1: Distribution User Ratings Training set (including Probe), both axes in log scale

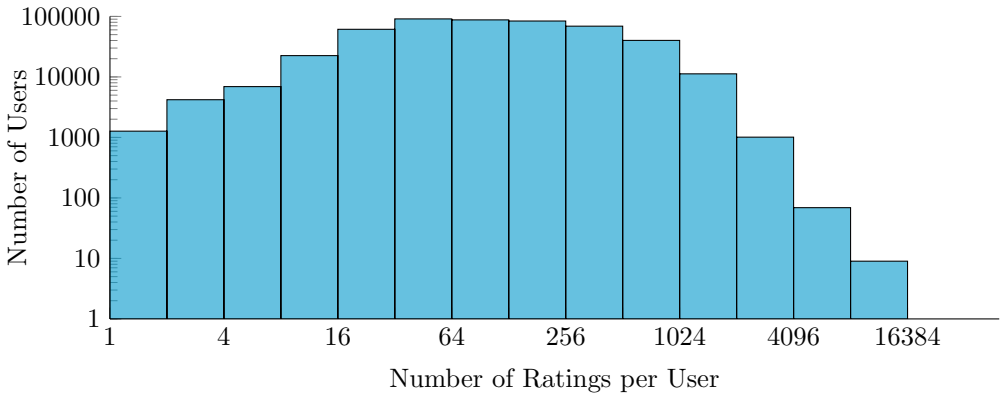

Figure 2: Distribution Movie Ratings Training set (including Probe), both axes in log scale

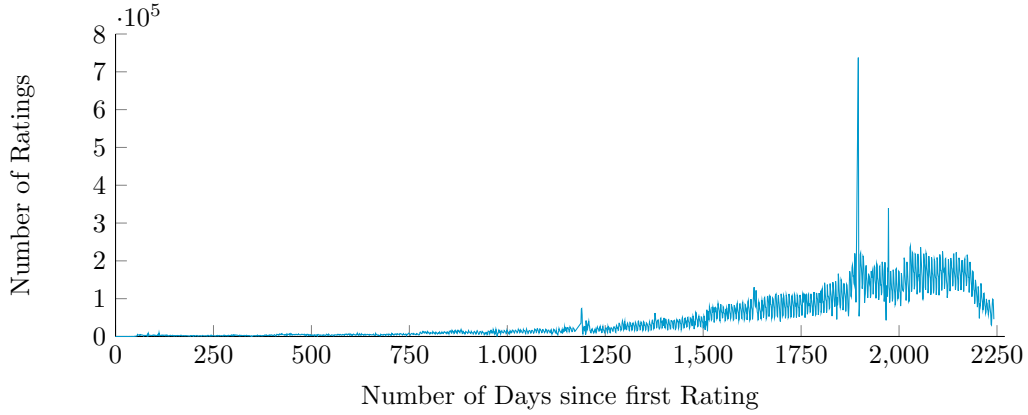

Figure 3: Number of Ratings in Time (including Probe)

2.1 Notation

We will use he following notation throughout the paper:

1. u,v for users, taking values between $1, 2, ...U$ with $U = 480,189$
2. i,j,k for movies, taking values between $1, 2, ...I$ with $I = 17,770$
3. r_{ui} the real rating of user u for movie i
4. $\hat{r}_{ui}$ the estimated rating of user u for movie i
5. $R(u)$ the set of all items rated by user u
6. $R(i)$ the set of all users that rated item i
7. $N(i;u)$ the set of neighbors of movie i that have been rated by user u

For some formulas we will need extra definitions.

3 Methodology

The following section describes the algorithm for the neighborhood-based Collaborative Filtering approach as proposed by Bell and Koren [3]. It covers the methods used for the three main components of the algorithm, as well as a detailed description of the methodology for our extension.

The prediction of user preferences is performed with the neighborhood-based approach (also known as k Nearest Neighbors or kNN). Usually the neighborhood-based approach computes the similarities between users and items and using these similarities to predict unknown ratings. We will follow the algorithm by Bell and Koren presented in [3]. Bell and Koren have extended the common kNN by adding a preprocessing step, namely data normalization by the removal of global effects. They have also proposed an alternative for the determination of interpolation weights. Therefore, this algorithm consists of three main components:

1. Data Normalization
2. Neighbor Selection
3. Determination of Interpolation Weights

The unknown rating by using an item-based kNN is predicted by:

$$\hat{r}_{ui} = \sum\nolimits_{j \in N(i;u)} w_{ij} r_{uj} \tag{1}$$

with w_{ij} the interpolation weight between item i and item j, $\hat{r}_{ui}$ the estimate for the unknown rating of user u of item i and $N(i;u)$ the set of neighbors of item i which have been rated by user u. We use the item-based kNN in this paper because according to [20] item-based algorithms provide dramatically better performance than user-based algorithms. The computations of user-based algorithms are more complex and inefficient compared to item-based algorithms.

3.1 Data Normalization

The first step is to normalize the rating data by eliminating global effects. In Table 1, we list all global effects that we are going to eliminate. Without normalization, these global effects could bias the predictions for the ratings. By data normalization the ratings become more comparable and this improves the estimation accuracy. This is because some users tend to give higher or lower ratings to items with respect to their average rating (User effect) and some items tend to receive higher or lower ratings with respect to other items (Item effect).

Besides user and item effects, there could also be other factors influencing the rating data. We will also take into account influences of the date of the rating. Time effects can explain additional variability in the ratings. For example, a movie could go in and out of popularity over time and similarly a user's taste could change over time. We will look at the following time effects: $\text{User} \times \text{Time(user)}^{1/2}$, $\text{User} \times \text{Time(movie)}^{1/2}$, $\text{Movie} \times \text{Time(movie)}^{1/2}$ and $\text{Movie} \times \text{Time(user)}^{1/2}$. The first effect allows the rating of the user to change linearly with the square root of the numbers of days elapsed for the rating since the first rating of the user. The second effect allows the rating of the user to change linearly with the square root of the number of days elapsed since the first rating of the movie by any user. $\text{Movie} \times \text{Time(movie)}^{1/2}$ allows the rating of the movie to change linearly with the square root of the number of days elapsed for the rating since the first rating of the movie. $\text{Movie} \times \text{Time(user)}^{1/2}$ allows the rating of the movie to change linearly with the square root of the number of days elapsed since the first rating of the user that rated the movie.

Furthermore, we will incorporate effects for users with movie characteristics (the average movie rating and the movie support). These effects tell us how a user is affected by the popularity and

the public opinion of a movie. Finally, we also incorporate effects for movies with the user characteristics average rating and support. We can easily derive these user and movie characteristics from the data.

We will sequentially estimate and incorporate one global effect at a time. By doing this, we can analyse the effect for incorporating the global effect one by one. We estimate the user and item specific parameters over the Training set excluding the Probe, to not overfit the data. We will describe the method for user specific parameters (θ_u), but the method for item specific parameters (θ_i) is exactly the same. Since we remove one global effect at a time, the dependent variable at each step is the residual from the previous step. So after the first step, r_{ui} refer to the residuals. The regression is given by:

$$r_{ui} = \theta_u x_{ui} + error \tag{2}$$

Hence, the residuals will be $r_{ui}(1) = r_{ui} - \text{effect}_1$ after the removal of the first effect, $r_{ui}(2) = r_{ui}(1) - \text{effect}_2$ for the second, until we have $r_{ui}(n) = r_{ui}(n-1) - \text{effect}_n$, with n the number of incorporated global effects. The effect in each step is equal to $\theta_u x_{ui}$, with x_{ui} the explanatory variable. For the main effects (Overall mean, Movie effect and User effect), the x_{ui}'s are identically 1, for the other effects we center x_{ui} by subtracting the mean of x_{ui} for that user. For example, for the User $\times$ Time(user)$^{1/2}$ effect, x_{ui} is equal to $\sqrt{days} - avg(\sqrt{days})$. For this effect, $days$ refers to the number of days elapsed since the first rating by user u till the time of rating r_{ui}. In Table 1 we list all sequentially incorporated global effects with their corresponding explanatory variable x_{ui}.

#	**Effect**	x_{ui}
1	Overall mean	1
2	Movie effect	1
3	User effect	1
4	User x Time(user)$^{0.5}$	$\sqrt{days} - avg(\sqrt{days})$
5	User x Time(movie)$^{0.5}$	$\sqrt{days} - avg(\sqrt{days})$
6	Movie x Time(movie)$^{0.5}$	$\sqrt{days} - avg(\sqrt{days})$
7	Movie x Time(user)$^{0.5}$	$\sqrt{days} - avg(\sqrt{days})$
8	User x Movie average	$avg(rating)_{movie} - avg(rating)$
9	User x Movie support	$support_{movie} - avg(support)$
10	Movie x User average	$avg(rating)_{user} - avg(rating)$
11	Movie x User support	$support_{user} - avg(support)$

Table 1: Global effects and their corresponding x_{ui}'s

An unbiased OLS coefficient estimator for θ_u is given by:

$$\hat{\theta}_u = \frac{\sum_{i \in R(u)} r_{ui} x_{ui}}{\sum_{i \in R(u)} x_{ui}^2} \tag{3}$$

where the sum is taken over all items rated by user θ_u. However, for sparse data this may give an unreliable estimate due to large variance. Following the approach by Bell and Koren in [3] we solve this sparsity problem by performing Bayesian shrinkage. The idea behind shrinkage is to give a penalty to the parameters with sparse data, so to the items with less ratings. With shrinkage we prevent overfitting on the dataset. This leads to the best estimator for θ_u being its posterior mean, $E(\theta_u|\hat{\theta}_u)$ [6]. We will use a simplified version of the estimator for θ_u by Bell and Koren as used in [3] given by:

$$\frac{n_u \hat{\theta}_u}{n_u + \alpha} \tag{4}$$

where n_u is the number of ratings by user u and α is a constant, determined by cross validation. After the removal of each effect, we will measure the accuracy of the predictions for the ratings

with their root mean squared error (RMSE). The RMSE for a specific dataset is given by:

$$RMSE = \sqrt{\frac{1}{N}\sum_{j=1}^{N}(\hat{r}_j - r_j)^2} \tag{5}$$

where $\hat{r}_j$ is the estimated rating for r_j and N is the number of ratings in the dataset. Note that j includes all combinations for u and i in the dataset. An estimated rating $\hat{r}_{ui}$ is equal to the inclusion of all removed global effects. For example, after the removal of the User effect, $\hat{r}_{ui}$ = Overall mean effect + $(\text{Movie effect})_i$ + $(\text{User effect})_u$. By calculating the RMSE's sequentially after the removal of a global effect, we are able to measure how the removal of a global effects improves the estimation accuracy.

3.2 Neighbor Selection

Now we have removed the global effects, our goal is to predict the unknown rating of user u of item i. If we refer to a rating, we now assume global effects have been removed. However, the methods that will follow could also be applied to ratings without removed global effects. For the prediction of an unknown rating, we use formula (1). For this formula we need to determine the neighbors for item i. In order to identify the closest neighbors of item i, similarities are required. In [3] Bell and Koren propose to use the following formula, based on the mean squared error between items:

$$s_{ij} = \frac{|U(i,j)|}{\sum_{u \in U(i,j)}(r_{ui} - r_{uj})^2 + \beta} \tag{6}$$

where $U(i,j)$ is the set of users who rated both items i and j and β a small constant, added to avoid division by zero. Previous approaches took s_{ij} as the Pearson correlation coefficient or the cosine similarity function, but the formula given in (6) works similarly well according to Bell and Koren [3]. The values of s_{ij} are precomputed once. Note that we only need to store the values of s_{ij} where $i \geq j$ because of symmetry. The total size of the similarities matrix is $I \times (I+1)/2$, so for the Netflix data with I equal to 17,770 this results in 157,895,335. Then, the top K items with the highest similarity are selected as the neighbors. We will consider different sizes of neighborhoods (K = 20, 35, 50) to compare their accuracy.

3.3 Interpolation Weights

To derive predictions for the ratings with formula (1), we need to determine the interpolation weights. The optimal interpolation weights can be found by solving the following least-squares problem:

$$\min_{w} \sum_{v \neq u} \left(r_{vi} - \sum_{j \in N(i;u)} w_{ij} r_{vj} \right)^2 \tag{7}$$

This leads to the solving a system of linear equations $Aw = b$, where A is a K×K matrix, such that $A_{jk} = \sum_{v \neq u} r_{vj} r_{vk}$, and b is a vector of size K, such that $b_j = \sum_{v \neq u} r_{vj} r_{vi}$. However, now we have assumed that all users but u rated both i and all its neighbors $N(i;u)$. This is often not the case, since we have a sparsity problem in our data. Even if there are users that have rated both i and all its neighbors, we would ignore lots of information on relationships between items by using the above formulas. To not ignore these relationships between items we estimate A and b by normalizing them by dividing by the number of users that rated both i and j. This leads to the following adjusted formulas:

$$\bar{A}_{jk} = \frac{\sum_{v \in U(j,k)} r_{vj} r_{vk}}{|U(j,k)|} \tag{8}$$

$$\bar{b}_j = \frac{\sum_{v \in U(i,j)} r_{vj} r_{vi}}{|U(i,j)|} \tag{9}$$

There is still a sparseness problem because there are many item combinations with small values for $|U(j,k)|$. To further solve the sparseness problem, Bell and Koren [3] use Bayesian shrinkage by giving penalties to values with small $|U(j,k)|$. We shrink the values towards a common mean, the baseline value. This value is calculated by precomputing the $I \times I$ matrix $\bar{A}$. We denote the baseline value by avg and this is the average of the entries of the matrix $\bar{A}$. We will use two different values for avg, one for the diagonal by averaging over the diagonal entries and one for the non-diagonal by averaging over the non-diagonal entries. This is because the diagonal entries are non-negative and therefore they are expected to be higher. Similarly as with the similarities, this matrix $\bar{A}$ is symmetric, so we only need to store the values for item $i \geq j$. After shrinkage, our estimates for A and b will be:

$$\hat{A}_{jk} = \frac{|U(j,k)| \cdot \bar{A}_{jk} + \gamma \cdot avg}{|U(j,k)| + \gamma} \tag{10}$$

$$\hat{b}_j = \frac{|U(i,j)| \cdot \bar{b}_j + \gamma \cdot avg}{|U(i,j)| + \gamma} \tag{11}$$

Now, we will derive the interpolation weights by solving the linear system:

$$\hat{A}w = \hat{b} \tag{12}$$

To solve this system the interpolation weights are constrained to be non-negative, following Bell and Koren, and we solve this linear system as a nonnegative least-squares constraint problem. Bell and Koren found out that restricting the interpolation weights to be non-negative results in better estimation accuracy.

With this three step algorithm, Bell and Koren are trying to overcome some issues that arise with the common form of kNN. These issues are:

1. Different CF algorithms use different similarity functions. Previous approaches derive the interpolation weights directly from the similarities, but this improved approach derives the interpolation weights directly from the ratings.
2. Common kNN ignores the similarities between neighbors. The improved interpolation weights take these interactions among neighbors into account (formula (10)).
3. The interpolation weights are not restricted to sum up to one. Items with uninformative neighbors may cause overfitting if the interpolation weights are restricted to sum up to one.
4. If there is lots of variation in the data among neighbors, kNN does not work well. This improved algorithm adjusts for these variations among neighbors.

4 Extension

This section describes our extension of the implemented neighorhood-based Collaborative Filtering approach by Bell and Koren. The extension consists of incorporating the Genre effect and incorporating temporal effects at the data normalization step.

We extend the neighborhood-based Collaborative Filtering approach by Bell and Koren by performing further data normalization. We begin with eliminating the effect for the genre of a movie, the Genre effect. In Table 2 there is an overview of the movie genres of the Netflix prize data [9]. These genres were obtained by analyzing the movie synopses on the Netflix website.

#	**Genre**
1	!! Uncensored
2	Action & Adventure
3	Anime & Animation
4	Children & Family
5	Classics
6	Comedy
7	Documentary
8	Drama
9	Faith & Spirituality
10	Foreign
11	Gay & Lesbian
12	Horror
13	Independent
14	Music & Musicals
15	NA
16	Romance
17	Sci-Fi & Fantasy
18	Special Interest
19	Sports & Fitness
20	Television
21	Thrillers

Table 2: Movie Genres

For each genre we will calculate a specific parameter using forumulas (1),(2) and (3). The x_{ui}'s for the Genre effect are identically 1. We also use the Training set excluding Probe to estimate the parameters. By incorporating the Genre effect we hope to eliminate the fact that some genres tend to receive higher or lower ratings with respect to other genres. For example, it could be possible that the genre Classics tends to receive higher ratings.

Furthermore, we will incorporate temporal effects. We will distinguish between ratings on a specific day of the week, ratings during weekdays and weekend, ratings during a specific season and ratings on holidays and non-holidays in the United States of America. The table below lists all the holiday days in the United States that we will take into account. These are a combination of holidays that most people celebrate and during which most people have paid time off.

#	**Holiday**
1	New Year's Eve & New Year's Day
2	Easter
3	Memorial Day
4	Independence Day
5	Labor Day
6	Halloween
7	Thanksgiving
8	Christmas

Table 3: Holidays in the United States of America

For these four temporal effects we will also use formulas (1), (2) and (3). By incorporating these temporal effects we want to eliminate the fact that ratings tend to be higher or lower during a specific time. For example, it could be possible that during holidays or during the weekend people tend to give higher ratings compared to non-holidays and weekdays respectively.

5 Results

In the following section we analyse the results of the implemented neighborhood-based Collaborative Filtering approach by Bell and Koren and our extension. First, we report the RMSEs on the Probe set with the inclusion of global effects. Then we report the RMSEs on the Probe set after varying stages of preprocessing the data and for varying neighbor sizes. After this we give the results for our extension.

5.1 Data Normalization

Table 5 lists all our incorporated global effects, the RMSE after their inclusion, the improvement of the RMSE after their inclusion and the shrinkage parameter used in formula (4). We use the shrinkage parameters used by Bell and Koren in [3]. Note that these RMSEs are calculated on the Probe set, with parameters trained on the Training set excluding Probe. The RMSEs for the inclusion of the global effects match with the results of Bell and Koren, until the effect Movie $\times$ User average. The RMSE for this effect is 0.9679, which deviates from the results of Bell and Koren of 0.9670. A possible explanation for this is that for the calculation of the Movie and User average we have taken the average over the raw ratings. Perhaps Bell and Koren have taken the average over the residuals for these effects. To get to the RMSE of 0.9657 by Bell and Koren after including all effects, we calculate the effect Movie $\times \sqrt{\text{User support}}$ instead of Movie $\times$ User support. After the inclusion of this effect, we lower the RMSE to 0.9658. This is a close match to the results of Bell and Koren. Therefore, in the further process we will continue with the Movie $\times$ $\sqrt{\text{User support}}$ effect if we remove all global effects.

Effect	**RMSE**	**Improvement**	**Shrinkage Parameter**
Overall mean	1.1296	-	-
Movie effect	1.0527	0.0769	25
User effect	0.9840	0.0687	7
User x Time(user)$^{0.5}$	0.9809	0.0031	550
User x Time(movie)$^{0.5}$	0.9785	0.0024	150
Movie x Time(movie)$^{0.5}$	0.9766	0.0019	4000
Movie x Time(user)$^{0.5}$	0.9758	0.0008	500
User x Movie average	0.9718	0.0040	90
User x Movie support	0.9689	0.0029	90
Movie x User average	0.9679	0.0010	50
Movie x User support	0.9667	0.0012	50

Table 4: RMSEs for the Probe set after the inclusion of global effects

From Table 5 we can see that the inclusion of the Movie effect and User effect give the largest improvement in RMSE. This is not a surprise, since these are the main global effects. This confirms that some users tend to give higher or lower ratings to movies compared to their average and that some movies tend to receive higher or lower ratings compared to other movies. The next two effects interact users with the time passed since the first rating of the user and the first rating of the movie. These effects improve the RMSE with 0.0031 and 0.0024 respectively. The effects for time for movies provide smaller improvements in RMSE compared to these effects for users, telling us that the ratings of the users change more over time compared to the ratings for the movies. The last four effects interact users and movies with their characteristics average and support. The RMSE after the inclusion of the interaction of the user with the movie average and movie support improves with 0.0040 and 0.0029 respectively. From these results we can conclude that users are definitely affected by the popularity of a movie. The last two effects provide a smaller decrease in RMSE. If we include the interaction of movies with the square root of user support, the RMSE improves with 0.0021 instead of 0.0012. So, performing data normalization without any interpola-

tion can deliver us a RMSE of 0.9658.

5.2 Neighbor Selection & Interpolation Weights

After the precomputing of all similarities s_{ij} and all A_{ij} values we are able to compute the matrix $\hat{A}$ and $\hat{b}$ by using formulas (10) and (11). By solving formula (12) we derive the interpolation weights. Table 6 gives the RMSEs of our implemented neighborhood-based method for different sizes of neighbors (K) and after varying stages of data normalization. These are the RMSE values on the Probe set. First we applied our methods on the raw ratings. For the raw ratings we have not performed any data normalization. For this stage we used shrinkage parameter γ equal to 50,000 in formula (10) and (11), determined by cross validation. Then we applied our methods on the ratings for which the Overall Mean, Movie and User effect have been removed. These effects are comparable to Double Centering. For the stage Global Effects we applied our methods on the ratings for which all global effects have been removed. For the last two stages of data normalization we used a shrinkage parameter γ equal to 500, the same value used by Bell and Koren as stated in [3].

Data Normalization	**K=20**	**K=35**	**K=50**
None (raw ratings)	0.9752	0.9746	0.9764
Double Centering	0.9253	0.9211	0.9221
Global Effects	0.9233	0.9194	0.9216

Table 5: RMSEs for varying stages of data normalization and varying sizes of neighbors (20,35,50)

The values for the RMSEs for the raw ratings are around 0.9750. These RMSEs are higher than the RMSE for the inclusion of all global effects of 0.9658, during which no neighborhood interpolation has been applied. The removal of the Overall Mean, Movie and User effect lower the RMSEs to 0.9253, 0.9211 and 0.9221. This improves the RMSEs with approximately 0.05. After applying the methods on the ratings for which all global effects have been removed, the RMSEs are equal to 0.9233, 0.9194 and 0.9216 for the different sizes of K. This stage improves the RMSEs with 0.0020, 0.0017 and 0.0005. These improvements are very small compared to the previous improvements. From this table we can conclude that Double Centering improves the predictions for ratings a lot compared to predictions for ratings without any data normalization. Removing more global effects, although providing a much better RMSE without neighborhood interpolation, does not improve the RMSE much compared to Double Centering. For the varying sizes of neighbors K we see that K = 35 provides the lowest RMSE.

Our RMSEs do not completely match with the RMSEs of Bell and Koren in [3]. An explanation for this could be that Bell and Koren allocated one byte for each individual value of the precomputed s_{ij} and A_{ij} values. We store these values as doubles (8 bytes), to not loose any precision. Therefore, our values for the similarities and A matrix could differ a lot. Another explanation could be that we use a different algorithm to solve the system of linear equations given in formula (12). Bell and Koren use the algorithm in Figure 1 in [3], based on the Gradient Projection method. We solve (12) as a nonnegative least-squares constraint problem with the MATLAB function lsqnonneg. Also, we probably use a different shrinkage parameter in formula (11) and (12) for the neighborhood interpolation on the raw ratings.

5.3 Extension

For our extension we perform further data normalization by removing the Genre effect and some temporal effects. We begin with eliminating the Genre effect. In Figure 8 the number of ratings per genre are visible. Genres 2, 6 and 8 have received many ratings compared to genres 3, 9, 11, 14, 15, 18 and 19. We estimate the Genre effect on the residuals after the removal of the Overall Mean, Movie and User effect. For this effect we have used a shrinkage parameter equal to 200,

determined by cross validation. This gives us a RMSE of 0.9834, and thus lowers the RMSE with 0.0006. This is a smaller improvement than we expected.

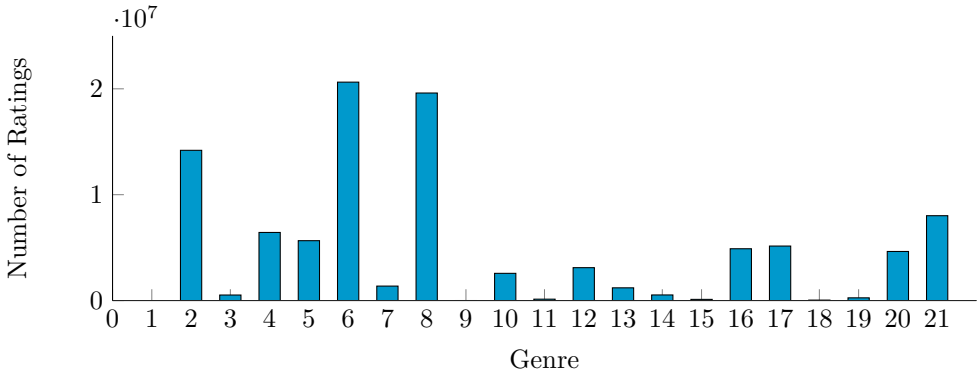

Figure 4: Number of Ratings per Genre

After the incorporating of the Genre effect we try the eliminate some temporal effects. More specifically, we try to eliminate the effect for Day of Week, Weekend, Season and Holiday. The below figures give information about the number of ratings for these specified times. All four temporal effects did not lower the RMSE of 0.9834 after the inclusion of the Genre effect. Therefore we conclude that ratings do not tend to be higher or lower during different days of the week, weekend and weekdays, seasons and holidays and non-holidays. Table 6 lists the months taken into account per season.

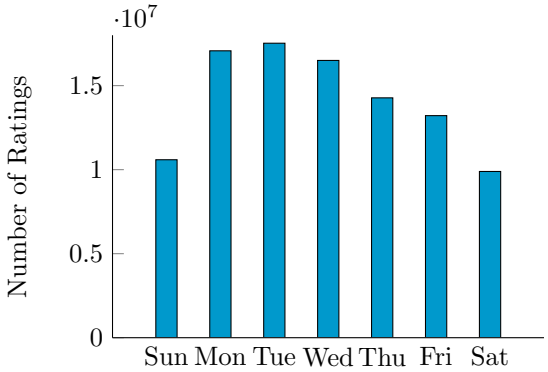

Figure 5: Number of Ratings per Day of Week

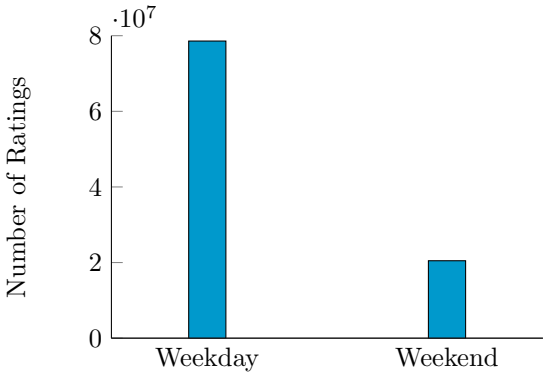

Figure 6: Number of Ratings Weekday vs. Weekend

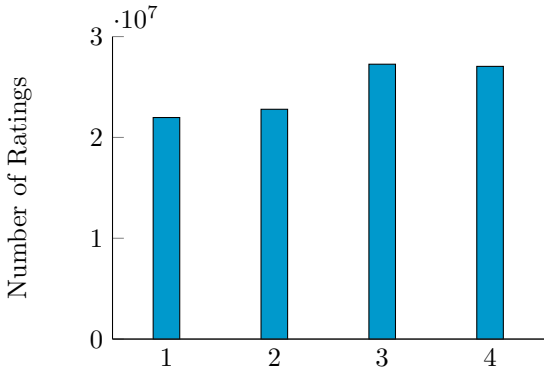

Figure 7: Number of Ratings per Season

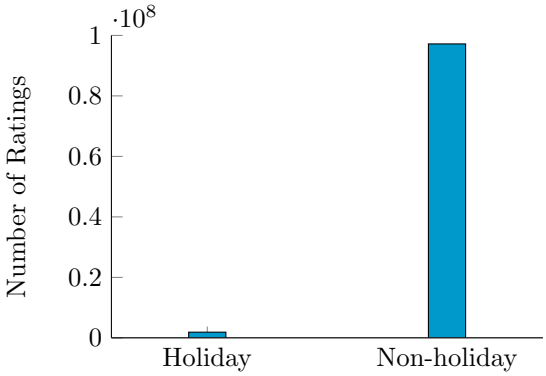

Figure 8: Number of Ratings Holiday vs. Non-holiday

# Season	Months
1	Dec, Jan, Feb
2	Mar, Apr, May
3	June, July, Aug
4	Sep, Oct, Nov

Table 6: Months per Season

Since the Genre effect is the only effect of our extension that lowers the RMSE after the removal of the overall Mean, Movie and User effect, we also remove the Genre effect after the removal of the 11 Global effects stated in Table 1. This gives us a RMSE of 0.9655. The improvement of the Genre effect is 0.0003. Since this improvement is very small, we do not expect that removing the Genre effect will further lower the RMSEs when applying the neighborhood interpolation methods. Therefore, we do not perform these methods for our extension.

6 Conclusion and Discussion

In order to make recommendations Bell and Koren proposed an improved neighborhood-based Collaborative Filtering method [3]. In this paper we have implemented these methods by Bell and Koren. Neighborhood-based methods make predictions by determining interpolation weights for similar items or users. Advantages of neighborhood-based methods are its easy implementation, its ability to easily explain the user the reasoning behind a recommendation and its intuitiveness. The improved methods by Bell and Koren overcome some of the issues of previous neighborhood-based methods. We used the item-oriented approach, since this approach provides better performance and more efficient computations [20]. We have evaluated the performance of the proposed methods on the Netflix prize dataset.

The approach by Bell and Koren consists of three main components. The first component is data normalization. We perform data normalization by sequentially eliminating global effects. By performing this normalization, we remove variability in the data and make the ratings more comparable. The data normalization step proofs to be very successful and an important component of neighborhood-based methods. The inclusion of global effects alone can lower the RMSE to 0.9658 (and 0.9655 with removal of the Genre effect). Data normalization definitely improves estimation accuracy.

The next two components are the selection of neighbors and determination of interpolation weights. For the selection of neighbors we have used a similarity function proposed by Bell and Koren, which is based on the mean squared error of items. This function seems to work good. By applying the neighborhood interpolation implemented by Bell and Koren, we can get to a RMSE of 0.9194. Our implementation of the methods do not seem to work very good on the raw ratings. These give higher RMSE values than the RMSE of the inclusion of all global effects. Adding data normalization improves the RMSE for the neighborhood interpolation a lot. However, the difference between the Double Centering and Global effects stage is very small. With the inclusion of full global effects, so without the neighborhood interpolation, the improvement was larger between these two stages. Therefore, we believe that performing further data normalization will not lower the RMSE much when applying the neighborhood interpolation methods by Bell and Koren.

For our extension we have performed further data normalization by incorporating a Genre effect and four temporal effects. However, only the Genre effect improved the RMSE a little bit. Therefore, ratings do not seem to be affected by these specified times (specific days of the week, weekend, seasons and holidays).

For future research we would propose performing further data normalization based on characteristics of the users. For example, it would be interesting to look at the difference in ratings by men and women, age classes and addresses or nationalities.

References

[1] `https://media.netflix.com/en/press-releases/netflix-unveils-profiles`.

[2] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[3] Robert M Bell and Yehuda Koren. Improved neighborhood-based collaborative filtering. In *KDD cup and workshop at the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 7–14. Citeseer, 2007.

[4] J Bennet and S Lanning. The netflix prize. kdd cup and workshop, 2007, 2007.

[5] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[6] Bradley Efron and Carl Morris. Data analysis using stein's estimator and its generalizations. *Journal of the American Statistical Association*, 70(350):311–319, 1975.

[7] Ricci Francesco, Rokach Lior, B Shapira, and PB Kantor. Recommender systems handbook, 2011.

[8] Simon Funk. Netflix update: Try this at home, 2006. *URL http://sifter. org/˜ simon/journal/20061211. html*, 2009.

[9] Sai Chaitanya Gaddam. Movie genre prediction dataset. *URL http://cns.bu.edu/ gsc/MovieGenre.html*, 2011.

[10] David Goldberg, David Nichols, Brian M Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[11] Ken Goldberg, Theresa Roeder, Dhruv Gupta, and Chris Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[12] Will Hill, Larry Stead, Mark Rosenstein, and George Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 194–201. ACM Press/Addison-Wesley Publishing Co., 1995.

[13] Joseph A Konstan, Bradley N Miller, David Maltz, Jonathan L Herlocker, Lee R Gordon, and John Riedl. Grouplens: applying collaborative filtering to usenet news. *Communications of the ACM*, 40(3):77–87, 1997.

[14] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434. ACM, 2008.

[15] Yehuda Koren. The bellkor solution to the netflix grand prize. *Netflix prize documentation*, 81, 2009.

[16] Yehuda Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 53(4):89–97, 2010.

[17] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.

[18] Royi Ronen, Noam Koenigstein, Elad Ziklik, Mikael Sitruk, Ronen Yaari, and Neta Haiby-Weiss. Sage: recommender engine as a cloud service. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 475–476. ACM, 2013.

[19] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Application of dimensionality reduction in recommender system-a case study. Technical report, DTIC Document, 2000.

[20] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[21] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating word of mouth. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 210–217. ACM Press/Addison-Wesley Publishing Co., 1995.