ERASMUS UNIVERSITY ROTTERDAM

Erasmus School of Economics

Master Thesis Data Science and Marketing Analytics

Data minimization for model-based recommender systems: a comparative analysis

Name student: Floor van Zon
Student ID number: 430757

Supervisor: A.C.D. Donkers
Second assessor: P.J.F. Groenen

Date final version: 07/11/2020

Table of Contents

Summary

Recommender systems are intelligent engines which can handle an overload of user information for decision-making purposes. They were originally developed to process massive amounts of user data. Nonetheless, in 2018, the European Union has introduced the General Data Protection Regulation (GDPR) which imposed restrictions on data storage and processing with the result that limited user-generated data will be available. This thesis investigates the impact of data minimization for model-based recommender systems. Data minimization does not only reduce storage, but also reduces privacy concerns and improves computation speed for recommendations. In this thesis, the performance of the Non-Negative Matrix Factorization (NMF) algorithm is evaluated for different data minimization techniques. This work proposes new techniques which incorporate contextual information into existing techniques. The first technique is based on a temporal window sliding backward in time and truncates data outside the selected time window. The second technique applies user profile truncation which truncates the oldest N ratings at the user level. The third technique creates new pseudo user profiles based on activity gaps and context. In order to measure the effectiveness of these techniques, a performance evaluation is executed on the MovieLens 10M dataset. This thesis demonstrates that the third data minimization technique, which is based on splitting users into smaller pseudo users, provides the most interesting results. It was demonstrated that a substantial amount of data can be truncated without deteriorating the performance of recommender systems. On top of that, it was proven that incorporating contextual information by creating segments does not further enhance the performance. Meanwhile, splitting users into smaller pseudo users based on the context of ratings was proven to be most accurate.

Keywords: Recommender System, Collaborative Filtering, Non-Negative Matrix Factorization, Data Minimization.

1. Introduction

1.1. Recommender systems

As of the start of this century, the amount of available user-generated data has grown unprecedently. Moreover, the introduction of big data processing techniques has strengthened the demand of businesses to store more and more user information that could potentially be used for future data processing . Many companies such as Amazon, Netflix and YouTube rely on user data as it is used as input for recommender systems. They benefit from new insights extracted from this big data in terms of increasing profits (Krishnaraj, 2019).

A recommender system can be defined as a decision-making strategy for users under complex information environments. They are intelligent engines that can collect information related to what a specific user has previously bought or liked, and they provide help by recommending items that customers might find interesting. These systems assist the social process of using recommendations of other users to make choices when there is insufficient personal knowledge of alternatives available (Resnick & Varian, 1997). Recommender systems can predict whether an item is preferred by a specific user based on his or her user profile. Also, they can handle an overload of information that users normally run into when they are provided with personalized item and service recommendations. Recommender systems are beneficial to both providers and users since they reduce transaction costs of finding the right item and improve the process of decision making accordingly (Pathak et al., 2010).

Several approaches have been developed for building recommender systems, of which content-based filtering, collaborative filtering and hybrid filtering are the three most commonly used approaches. A content-based filtering system matches content resources to user characteristics. This type of recommender system bases its predictions or recommendations only on user information, while ignoring contributions of other users. In contrast, collaborative filtering systems make predictions or recommendations by identifying users with similar preferences; they make use of their opinions to recommend items to the active user (Isinkaye et al., 2015). Hybrid filtering techniques usually combine multiple types of recommendation strategies in different ways to benefit from those complementary advantages. An example of hybrid filtering is merging content-based filtering with collaborative filtering techniques (Çano & Morisio, 2017). This thesis will solely focus on the collaborative filtering approach for building a recommender system.

1.2. Problem statement

In May 2018, the European Union has imposed rigorous restrictions on storage and processing of user data, referred to as the General Data Protection Regulation (GDPR). The main idea behind these restrictions is to minimize the collection of user data. By imposing the GDPR, users are enabled to remove their data from a business' data processing as stated by the 'right to be forgotten'. This right implies that users can choose to be forgotten such that data processing entities are obligated to delete a user's personal data. As a result, limited user data will be available (Goddard, 2017).

Data minimization is a privacy practice which limits the amount of data that can be collected from users. This principle states that collecting and processing data is allowed if and only if this is essential for reasons that were clearly stated in advance to support data privacy (Intersoft Consulting, 2018). By imposing these stringent requirements, data processing entities are obligated to process only 'adequate, relevant and limited' user data (GDPR, Article 5). Next to these restrictions, users have several rights in order to control their personal data. Users have the right to inspect their personal data an entity is processing (GDPR, Article 15). In addition, users have the right to be forgotten and therewith erasing their personal information from the data processing (GDPR, Article 17). Data minimization is therefore also referred to as the principle of reducing the amount of data used for processing. As stated by Hu et al. (2006), data reduction can be achieved, for example, by reducing the number of samples and thereby limiting the user data to an amount that is necessary to achieve certain goals.

Recommender systems were originally developed to utilize massive amounts of data. However, it is questionable whether after a certain amount of user data the performance of recommender systems continues to increase. If abundant user information does not significantly increase the performance of a recommender system after a certain point, computation time will increase unnecessarily (Larson et al., 2017). Whether or not using ample user-generated data for training a recommender model may have a meaningful effect on prediction accuracy is still questionable. This 'problem statement' of user data reduction has not already been studied well in literature and is often still disregarded in the development of current recommender systems.

This thesis sets out the investigation of data minimization for training a recommender system. The objective of this research is to analyze different techniques for minimizing data and the requirements of training data for recommender systems. We build further on research of Larson et al. (2017) and Krishnaraj (2019) who introduced different minimization techniques. Larson et al. (2017) proposed a technique based on a sliding time window in which data outside the considered time window will be discarded. Krishnaraj (2019) introduced a user profile truncation technique which first determines long users and then discards the oldest data for each long user while it keeps the most recent ratings given a long user.

Next to truncating data on the user level, this research performs a truncation at the item level in order to investigate the importance of item recency next to rating recency. Additionally, emphasize is placed on the context in which a rating was made. Since context plays a crucial role in determining user behavior providing additional information that can be exploited in building predictive models (Adomavicius & Tuzhilin, 2008), this research investigates the impact of context-aware data minimization techniques on the performance of recommender systems. The utility of a certain item to a user may depend significantly on contextual information such as day of the week (Bashiri, 2018). Also, rating data often comes with contextual information such as time and does not require explicit user input. This type of information might transfer additional explanatory value since users' short-term preferences rely on context-specific factors (Richthammer & Pernul, 2018). Incorporating this contextual information to recommender systems captures the impact of factors that affect users' short-term preferences. Results will prove whether user preferences differ across different points in time, e.g. weekend versus weekdays. If so, recommender systems can be enhanced in terms of providing more accurate recommendations based on contextual information.

1.3. Structure

The rest of this thesis is structured as follows. The next section provides an overview of work related to data minimization techniques for recommender systems. Section 3 examines the technique of collaborative filtering and (non-negative) matrix factorization for recommender systems. Also, the evaluation metrics will be addressed in this section. Section 4 describes and visualizes the MovieLens 10M dataset which is used in this research to experiment on. Next to the dataset, the recommender framework will be addressed in the fourth section. Sections 5 to 7 cover a detailed analysis for each of the three proposed frameworks for data minimization, including the obtained results. For more clarification, section 8 provides an overview of the techniques applied in this research. Finally, section 9 concludes this research and provides some recommendations for future work.

2. Related work

Academic research on recommender systems has faced an unprecedented large increase over the past years. As of the publication of the first research paper of Resnick & Varian (1997) on collaborative filtering technique, recommender systems have become a very popular research field. In October 2006, Netflix has released a large anonymous dataset containing 100 million movie ratings to challenge computer science and machine learning communities to develop a recommender system that outperforms its own recommender system Cinematch (Bennett & Lanning, 2007). Due to the Netflix Prize, many researches have evaluated the performance of recommender systems against the number of users in training set. Over the last years, there has been some research related to data minimization and time-based filtering techniques which will be reviewed in this section.

Gordea & Zanker (2007) aim to improve the quality of recommender systems by proposing time heuristics into the current recommendation process. They present two new approaches of collaborative filtering techniques for solving the problems of cold start and sparse data matrices. A time filtering technology was presented which was demonstrated to improve the quality of recommender systems. They conclude that both approaches, a time decay and a time window filter, are effective methods for recommenders when used in the right context.

Baltrunas & Amatriain (2009) propose a contextual pre-filtering technique based on implicit user data. They introduce micro-profiling for time-aware collaborative filtering, which is a new context-aware recommender approach. In this approach, the user profile is split into multiple sub-profiles, each representing a different context. The subsequent predictions are done using these micro-profiles rather than a traditional single user model. Different time splits have been evaluated and the results prove that using solely user micro-profile data can potentially increase prediction performance of the algorithm.

Campos et al. (2010) propose several strategies for taking into consideration the temporal context for movie recommendations. Their strategies are mainly based on variations of the KNN algorithm. From their results they conclude that considering information close to the recommendation date, i.e. most recent data, can help improving prediction performance of recommender systems. Additionally, recommender systems benefit from this method in terms of reduction of information overload.

Research of Shang et al. (2014) was mainly motivated by the development of information technology and the corresponding concern of risk in revealing private user data to providers. They presented a privacy preserving recommender framework based on different groups. These groups serve as a natural intermediary to protect users' privacy. A distributed preference exchange algorithm was proposed in order to preserve data anonymity, where the effective size of the anonymity set approaches the group size with time. Experiments show that the proposed method outperforms baseline methods despite absence of private user data.

Campos et al. (2014) build further on current time-aware recommender systems (TARS) and present a comprehensive survey and analysis. They identify several key conditions on offline evaluation of TARS and provide a classification of evaluation protocols for TARS correspondingly. Besides that, they present an empirical study on the impact of different evaluation protocols on measuring relative performances of three widely used TARS approaches. One approach discussed is time truncation, which drops ratings that are older than a specific threshold. From the analysis and experiments, Campos et al. (2014) conclude a set of guidelines, which recommend making a training-test split based on a time-dependent rating order over the complete set of ratings.

Larson et al. (2017) focus on the principle of minimal necessary data. They state that if two recommender algorithms are equally effective, then the algorithm requiring less user data is the better one. Also, in order to address the cold start problem, recommender systems should use minimal necessary data. They argue that using more data is unnecessary when improvements in prediction performance become negligible. They split the training set based on rating dates (i.e. the training set is temporally ordered and then split into segments) and construct temporal windows of unequal lengths to vary the size of the training sets. Based on their results, Larson et al. (2017) conclude that using minimal necessary user data represents a relatively small change in current practices.

Wen et al. (2018) aim to improve current recommender systems and emphasize the need for robustness against increasing privacy concerns and personal data protection. They investigate the effects of time-sensitive user data filtering on recommender performance. Based on their experiments, Wen et al. (2018) conclude that excluding historical user data and only considering recent data does not have a significant impact on the overall performance of recommender systems. However, they found that this effect may vary among users. Finally, they suggest a potential win-win solution for both users and providers since they demonstrate that more data is not always better.

Krishnaraj (2019) investigated the effects of data minimization principles advocated in GDPR on the performance of recommender systems. Three data minimization techniques were proposed and performed on the Biased Matrix Factorization algorithm. The experiments in this research solely use rating information for the rating prediction task. Results demonstrate that a substantial amount of the data can be truncated without having a large impact on the performance of the recommender model. However, they conclude that more research is needed to draw substantive conclusions, e.g. prediction accuracy could be enhanced by incorporating contextual or temporal information next to rating information.

Briefly, the research papers discussed in this literature review have addressed several time-aware filtering and data minimization techniques for recommender systems. The main findings include that filtering techniques that are time aware can actually be effective methods for recommender systems. Also, the technique of splitting users into multiple micro-profiles can increase prediction performance of the recommender algorithm. However, more research is needed as it can decrease performance for some contexts.

As the amount of research papers related to data minimization techniques is only limited and many adaptions of data minimization techniques can be created, more research is needed to draw substantive conclusions on the performance of recommender systems. The above mentioned research papers have indicated that 'the right conditions' are of great importance in order to make a technique effective. Furthermore, different users and contexts were proven to affect the prediction performance significantly and make it more difficult to draw substantive conclusions.

Both time-based filtering techniques and user splitting techniques seem to be able to provide interesting possibilities for further enhancing recommender algorithms. Next to that, incorporating contextual information by pre-filtering techniques has demonstrated to be effective in improving prediction accuracy for some contexts. As the combined impact of data minimization techniques and incorporating contextual information on the performance of recommender algorithms has not yet been studied well, this will also be examined in this research.

3. Methodology

This section addresses an extensive description of the methodology used in this research. The first subsection examines the technique of collaborative filtering for making recommendations. Subsection 3.2 deals with dimensionality reduction, while subsection 3.3 covers matrix factorization. Finally, the fourth subsection discusses the non-negative matrix factorization algorithm which is used for making recommendations in this research.

3.1. Collaborative Filtering

Collaborative filtering (CF) techniques make recommendations based on similar users' preferences. This technique relies on the assumption that similar users like similar items. Based on this assumption, an item is recommended to an active user that similar users have liked or bought in the past. The main goal of CF is to predict missing ratings within the user-item matrix, i.e. to predict the rating for a given user and item. Next to rating prediction, CF can be used to provide top-*N* list recommendations that predict the *N* most relevant items given a user.

CF algorithms can be distinguished into model-based and memory-based algorithms. This paper focuses solely on a model-based approach of collaborative filtering. Unlike memory-based, there is no need to use the complete dataset of ratings every time, which brings benefits of both scalability and computation speed. Also, it is proven that model-based algorithms can provide more accurate predictions with little available information over other recommender algorithms.

Model-based algorithms create a predictive model from the ratings in the training dataset which will be used to make predictions accordingly. This predictive model is created by learning from the training dataset (the user-item matrix) by finding hidden patterns and features. These hidden patterns and features are latent factors, which describe the characteristics of the users and items. Model-based recommenders find different features by only considering rating data rather than meta data extracted from items or users. User and item profiles are created in such a way that in a new latent factor space both entities can be compared.

3.2. Dimensionality Reduction

Recommender systems often suffer from the malediction of dimensionality since they have to deal with many different features. This means that for an increasing number of features, the prediction performance of the learning algorithms will decrease (Steeg, 2015). As an example, in case of high dimensionality, the similarity between two items becomes less meaningful and the computation time will increase with the number of features. On top of that, the user-item matrix becomes very sparse.

A solution to these problems is provided by introducing dimensionality reduction techniques such as Principal Component Analysis (PCA), Latent Dirichlet Analysis (LDA), Singular Value Decomposition (SVD) and Matrix Factorization (MF). These techniques transform the higher dimensional space into a low-dimensional space and thereby simplifying the data model. Here, only important data is kept, while irrelevant and noisy data is discarded. Next to their function as a preprocess step, dimensionality reduction techniques can also be used for predicting ratings directly (Amatriain et al., 2011). They have been proven to quickly generate recommendations with a high accuracy rate. While memory-based models are still a widely used technique for recommender systems since they are easy to implement, intuitively and easy explainable, matrix factorization (MF) models have increased in popularity over the last years. MF techniques have been widely used in research since they are a more scalable and accurate alternative.

In this research, MF will be used as it comes with several advantages. MF reduces computation time while training the algorithm, which is especially beneficial in highly dimensional datasets. This makes it easier and faster for the training algorithm to learn from compact information-dense features. In addition, MF techniques by itself can be used to build recommender systems.

3.3. Matrix Factorization

Matrix Factorization (MF) is a significant approach in many applications and pursues two goals. The first goal is dimensionality reduction of the rating matrix. Second, it aims at discovering potential features under rating matrix which serve as a purpose of recommendation. The main assumption of MF is that there exists a low dimensional latent space of features which can be used to represent both users and items. Users and items are represented in such a way that the interaction between a specific user and an item can be computed by taking the dot item of the corresponding dense vectors in that particular space. The algorithm works by decomposing the original matrix, which is often a huge and very sparse user-item interaction matrix, into an item of two smaller matrices.

The user-factor matrix which contains the user representations multiplies the factor-item matrix which contains the item representations, see Figure 3.1. Here, the user-factor matrix is represented by matrix W, while the factor-item matrix is represented by matrix H. The multiplication of matrices W and H gives the user-item matrix V.

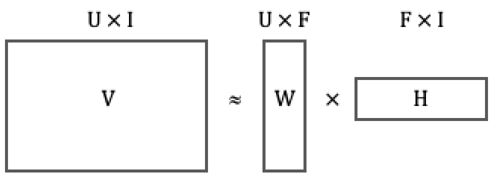


Figure 3.1: Matrix Factorization example where utility matrix V is decomposed into matrices W and H

Here it is assumed that there exist some features that describe items well and these features can be used to describe user preferences. Note that these features are not given explicitly but rather discovered automatically by the model, i.e. they are learned as a model. These extracted features do have a mathematical meaning but do not have an intuitive interpretation and are therefore difficult to understand. Though, a result of matrix factorization is that similar users in terms of item preferences and similar items in terms of characteristics end up having similar representations in the latent space.

Many applications of MF have been developed over the years, among which Singular Value Decomposition (SVD) is a well-known and commonly used application, introduced by Jolliffe (1986). Another application of MF is introduced by Lee and Seung (1999) and is called Non-negative Matrix Factorization (NMF) in which matrices V, W and H are nonnegative real-valued matrices. Non-negative Matrix Factorization (NMF) is especially useful when there are many attributes or when the attributes have weak predictability or are ambiguous. By combining different attributes, NMF can produce meaningful patterns.

3.4. Non-Negative Matrix Factorization

In machine learning it is often necessary to reduce the feature space of a dataset considering the ease and speed of computation. Like traditional MF, Non-Negative Matrix Factorization (NMF) is a method of reducing the dimensionality of a dataset into a linear combination of bases. NMF has a non-negative constraint, which implies that it can be used to represent data with non-negative features quite well. This method is similar to Principal Component Analysis (PCA) in such a way that it assigns weights to a set of bases. But, where PCA constrains the columns of matrix W to be orthonormal and the rows of matrix H to be orthogonal to each other, NMF does not allow for negative values in the matrix factors H and W. Imposing this non-negativity constraint brings two primary advantages: interpretability and storage. In many applications, the quantities involved simply cannot take a negative value. In such case, it might be difficult to interpret results of methods such as PCA, while for NMF these results can be interpreted immediately. Also, the NMF produces an ‘additive parts-based’ representation of the data. No subtractions can occur as parts are generally combined additively to form a whole because all elements in matrices W and H are positive (Lee & Seung, 1999). A consequence of this is that both W and H are generally naturally sparse which ensures that a large amount of storage space can be saved, compared to other low-rank algorithms such as SVD.

Consider a matrix V of dimension $(U \times I)$ where each element $v_{ij} \geq 0$. The NMF algorithm decomposes this matrix V into two matrices called W and H of dimensions $(U \times F)$ and $(F \times I)$ respectively, where each element $w_{ij} \geq 0$, $h_{ij} \geq 0$ and $F < \min(U, I)$ such that:

$$W \times H = V. \tag{1}$$

Here, matrix V is decomposed into a skinny, tall matrix W and a wide, short matrix H. The inner dimension of W and H, F, can be specified as long as $F < \min(U, I)$. Each column of V, v_i, can be calculated using the following formula:

$$v_i = W \times h_i, \tag{2}$$

where each column of V is equal to the sum of each column of W times its corresponding row in h_i.

The goal of NMF is to achieve a good approximation with:

$$K \ll \text{rank}(V). \tag{3}$$

Most of the information carried by matrix V is contained in matrices W and H if (1) is a good approximation. In case of recommendations, matrix V is very sparse, i.e. most of V is unknown. The goal is to fill in those entries by predicting unknown ratings.

In order to conduct non-negative matrix factorization, an objective function is needed which is a criterion for optimization. A commonly used measure of distance is the Frobenius norm, which is the Euclidian (L2) norm with the matrix treated as vector, or the sum of element-wise squared errors (Matloff, 2016).

Formalizing this, we obtain:

$$\min || V - \mathrm{WH} ||_F^2 \text{ w.r.t. W,H s.t. W,H } \geq 0. \tag{4}$$

There exist a couple of optimization methods that could be used, among which Multiplicative Update (MU) and Hierarchical Alternating Least Squared (HALS). MU is a commonly used method of optimization, due to Lee and Seung (1999), in which matrices H and W are updated iteratively according to the following formulas:

$$\mathrm{H} \leftarrow \mathrm{H} \odot \frac{\mathrm{W^T V}}{\mathrm{W^T W H}}. \tag{5}$$

$$\mathrm{W} \leftarrow \mathrm{W} \odot \frac{\mathrm{V H^T}}{\mathrm{W H H^T}}. \tag{6}$$

Here, the matrix that is not being updated is kept constant.

HALS is a less commonly used method of optimization, but is proven to be faster for computing NMF. This method updates one column of matrix W at a time, assuming that the columns of matrix W do not interact with each other. Thus, it computes the optimal solution for each column, while keeping the other columns constant:

$$\mathrm{W}[:, \mathrm{i}] \leftarrow \max\left(0, \frac{\mathrm{V\, h_i^T} - \sum_{\mathrm{k \neq i}} \mathrm{W}[:, \mathrm{i}]\mathrm{h_k h_i^T}}{||\mathrm{h_i}||^2}\right), \tag{7}$$

with $\mathrm{W}[:, \mathrm{i}]$ being the ith column of matrix W and $\mathrm{h_i}$ being the ith row of column H.

A major benefit of MF is its flexibility in dealing with various data aspects and application-specific requirements. Matrix factorization models map both items and users to a joint latent factor space of dimensionality F (see Figure 3.1), as a means to model the user-item interactions as inner items in that space. The resulting dot item captures the interaction between users and items, which denotes the overall interest in the item's characteristics (Koren et al., 2009). In spite of that, much of the variation in ratings among different users is caused by effects associated with either users or items. These effects are also called biases. For example, some users have tendencies to rate items higher than others. Or, some items are likely to receive higher ratings than others. As a result of these biases, it would be misguided to explain rating values solely by the user-item interactions.

Alternately, it is common to include these user and item-specific biases in order to explain the full rating value. The bias involve in rating values can be expressed as:

$$b_{ui} = \mu + b_i + b_u. \qquad (8)$$

Here, b_{ui} denotes the biases involved in the rating given by user u for item i. μ denotes the average rating overall and parameters b_i and b_u respectively indicate the observed variations of user u and item i.

3.5. Evaluation metrics

The quality of recommender systems can be evaluated using different types of metrics. The type of metric depends heavily on the type of filtering technique that has been used. There exist two types of measurements, which are accuracy and coverage. Accuracy can be defined as the proportion of correct recommendations out of the total possible recommendations. Coverage measures the proportion of objects in the search space the recommender system can provide recommendations for. Both types of measurements can be divided into statistical and decision support accuracy metrics. In this research, focus lies on the prediction accuracy of recommender systems. While two predictions tasks can be distinguished: rating prediction and top N-list recommendations, this thesis solely focuses on rating predictions. Therefore, only statistical accuracy metrics will be used for evaluation. A statistical accuracy metric evaluates the accuracy of a filtering technique by comparing the predicted user ratings for an item directly with the actual ratings.

3.5.1. Evaluation of rating predictions

Rating prediction is the task to predict the rating $\hat{r}_{ui}$ for items that have not yet been rated by users. The recommender system predicts for the test set Te for which the actual ratings r_{ui} are unknown. The error of rating prediction is determined by the difference between the predicted rating $\hat{r}_{ui}$ and the actual rating r_{ui}. Root Mean Square Error (RMSE) is a metric that is often used for rating prediction and measures the deviation of recommendations. RMSE is the square root of the mean of all squared errors in the test set Te:

$$\text{RMSE} = \sqrt{\frac{1}{|\text{Te}|} \sum_{(u,i) \in \text{Te}} (r_{ui} - \hat{r}_{ui})^2}, \qquad (9)$$

where Te is the total of user-item pairs (u, i) in the test set, r_{ui} is the actual rating for user u on item i and $\hat{r}_{ui}$ is the predicted rating on active item i from active user u. The lower the RMSE, the more accurate the predicted ratings are. In this thesis, RMSE will be used as primary metric for performance evaluation.

4. Experimental Design

In this section, the experimental design will be discussed. First, a detailed description and analysis of the MovieLens 10M dataset will be provided. This will be followed by discussing the evaluation metrics and tuning the hyperparameters in order to maximize the performance of the recommender system.

4.1. MovieLens 10M

The dataset used to experiment on is the MovieLens 10M dataset. This dataset was collected by GroupLens Research Project at the University of Minnesota and has a long temporal duration. Data was collected during the period from January 1995 to January 2009. The MovieLens 10M dataset captures explicit user preferences of movie ratings on a scale from 0.5 to 5. The complete dataset contains 10,000,054 ratings from 69,878 users on 10,681 movies. The timestamp denotes the time when the user has rated the movie (Unix epoch time measured in seconds from 1 January 1970). The user-item matrix shows that a user has rated 1.34% of all movies on average. A snapshot of the dataset can be found in Table 4.1.

Table 4.1: MovieLens 10M dataset snapshot

User ID	Movie ID	Rating	Timestamp
31796	595	3.0	841082997
23907	4993	4.5	1118203347
52212	1035	2.0	842500471
59710	3034	5.0	997794426
56463	2273	3.0	1181506749
11338	5219	2.0	1036089367

The values in the first column ‘User ID’ represent the users that have rated items, where each unique user has been assigned a unique user id value. In the same way, unique values are assigned to unique movies, which are represented by the second column ‘Movie ID’. The third column ‘Rating’ shows the corresponding ratings of a specific user given a movie. The fourth column ‘Timestamp’ denotes the corresponding amount of time that has passed since 1 January 1970, measured in seconds. Timestamp can be used to define the date on which a movie has been rated by a specific user. An example of interpretation is as follows: the first row of Table 4.1 shows that user 31796 has rated movie 595 with three out of five stars on Monday, 26 August 1996. In the rest of this thesis focus is on user ratings and the corresponding task of predicting unknown ratings.

4.2. Analysis

Several studies have shown that characteristics of recommender systems change over time as a result of changes in users and items (Bashiri, 2018). In this section, focus lies on those changes of users and items over time. First, the growth of ratings, users and items over time will be analyzed.

Figures 4.1 and 4.2 visualize the cumulative growth over time for users and items, respectively. From these plots can be observed that the number of both users and items increases as we move forward in time. This is caused by the fact that new items and users are added on a weekly basis. Also, from the cumulative growth of ratings in Figure 4.3, one can see that the number of ratings increases with items and users.

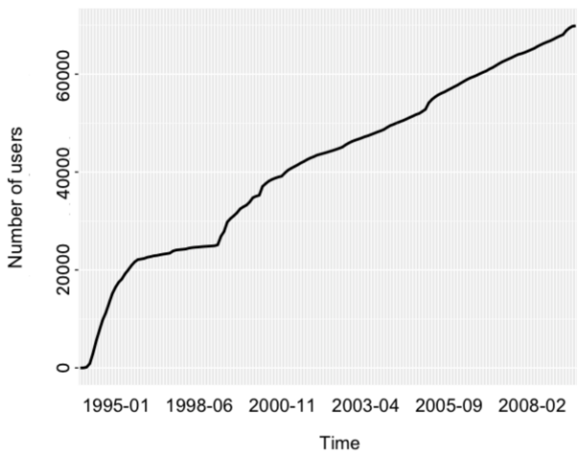


Figure 4.1: Number of users over time

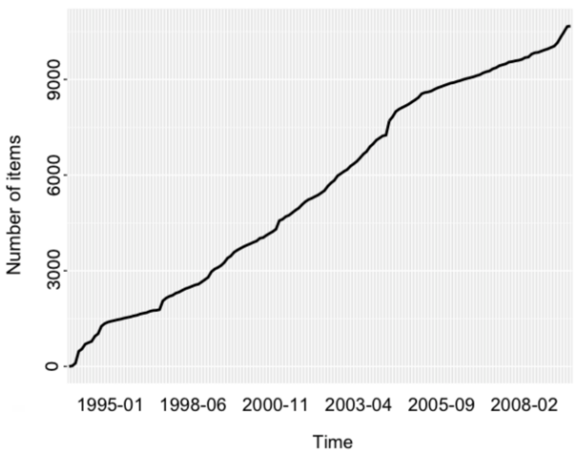


Figure 4.2: Number of items over time

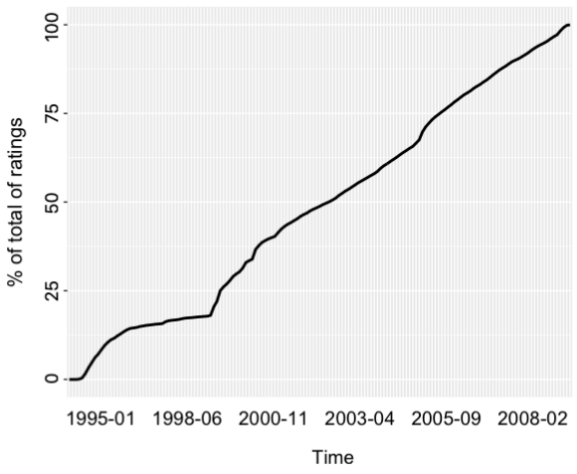
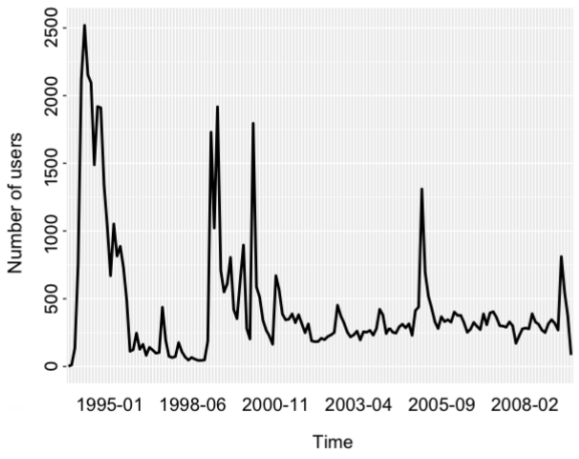
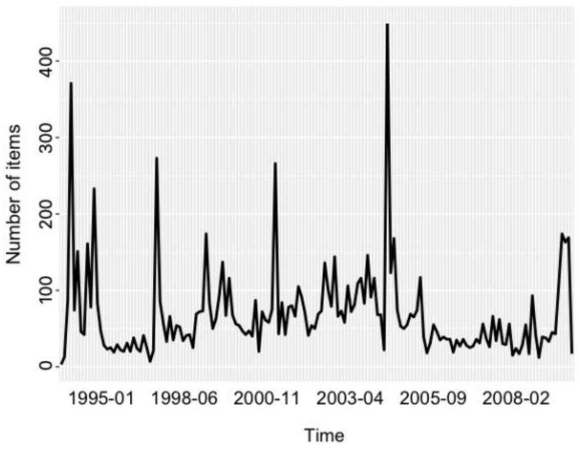


Figure 4.3: Number of ratings over time

In Figure 4.3, the y-axis denotes the percentage of the total of ratings. For example, around the year 2002 a level of 50% is reached. This implies that around 2002, half of all ratings were made. Logically, a level of 100% is reached in 2009, implying that at the end of the period concerned, all ratings have been made.

Next, the patterns of new users and items over time will be analyzed, which are visualized in Figures 4.4 and 4.5. Concerning the number of new users over time (Figure 4.4), one clear peak around 1995 can be observed and two slightly smaller peaks around 1999-2000. This implies that around these two periods, many new users have started rating their first item. In contrast, the number of new items over time shows a different pattern over time (Figure 4.5). Over the whole time span, there are several peaks visible, which refers to the fact that new items have been added multiple times per year. However, within each year there is one large peak visible. The highest number of new items added are in 1995 and 2004, where the largest peaks are shown.





` Figure 4.4: Number of new users over time

Figure 4.5: Number of new items over time

Regarding the ratings over time as shown in Figure 4.6, a pattern similar to that of new users over time (Figure 4.4) can be observed. This similarity is most likely caused by the fact that when there is a peak in the number of new users, there will correspondingly be a peak in the number of ratings as these new users have started rating items. Logically, the number of ratings is higher than the number of new items, which indicates that each user has rated at least multiple items.

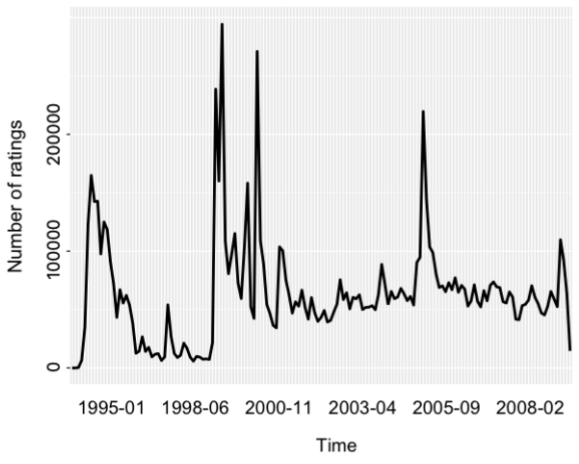


Figure 4.6: Number of ratings over time

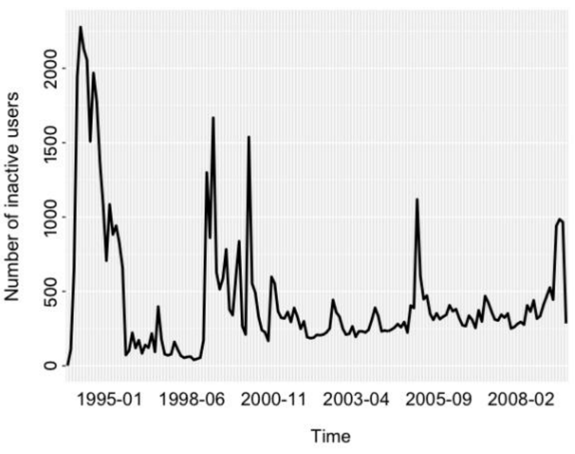


Figure 4.7: Number of inactive users over time

Now that the number of new users has been analyzed, it might also be interesting to examine the pattern of users who have become inactive, i.e. users who have stopped rating new items at a certain time. From the pattern of inactive users as shown in Figure 4.7, one can observe a high peak around 1995 and another smaller one around 1999-2000. These peaks correspond to the peaks of new users over time. This similar pattern indicates that there are users who started rating items (i.e. new users) but became inactive shortly after. Note that this does not necessarily imply that users have rated only one item as it is possible to rate multiple items in one day. Also, the number of inactive users is lower than the number of new users, which indicates that not all users have stopped rating items.

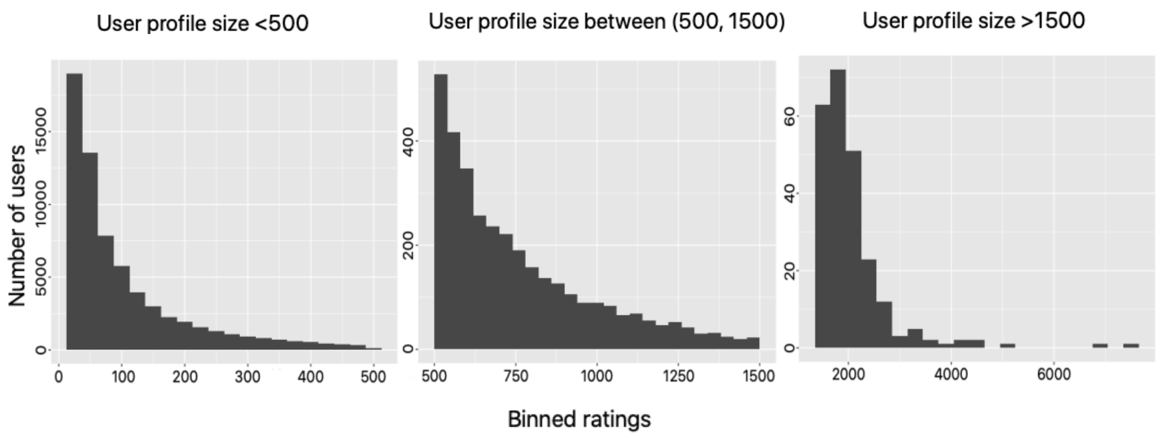


Figure 4.8: Number of users grouped by profile size

Finally, we will dive into users and their number of ratings. Figure 4.8 shows the distributions of the number of ratings for different user profile sizes. The left bar chart demonstrates the distribution of users with a profile size smaller than 500, i.e. users who have rated between 1 and 499 items in total.

There are more users who have rated a small number of items (<100) than a large number of items (>300). The middle bar chart focuses on users with a profile size between 500 and 1500. Again, it can be seen that there are more users who have rated a smaller number of items (<750) than a large number of items (>1250). The right bar chart shows the distribution of users with a profile size that exceed 1500 ratings. Note that the y-axis values differ substantially between the different user profile sizes.

In total, there are 66,195 users with less than 500 ratings. 3,444 users have a profile size between 500 and 1500 and there are only 239 users with a profile size larger than 1500. There are 97 profiles with more than 2000 ratings and only 7 profiles with more than 4000 rated items. As these 'extreme' numbers of items rated per user (e.g. >4000) are rare, not all values of binned ratings are represented in the dataset. These are indicated by the empty bars as shown in the right bar chart.

4.3. Recommender framework

As discussed in section 3.4, the Non-Negative Matrix Factorization algorithm will be used for rating prediction. The next subsection will provide a detailed overview of the algorithms' hyperparameter tuning in order to produce a fair comparison and reproducible results.

4.3.1. Hyperparameter optimization

Hyperparameter optimization is the task of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is referred to as a parameter whose value must be set before the training process starts; it is a parameter from a prior distribution. In contrast to a hyperparameter, a model parameter can be seen as a property of the training data and will learn on its own during the training process. Tuning hyperparameters is a crucial step in the machine learning process to obtain good results as they have a significant impact on the performance of recommender models. Hyperparameters can be used in order to maximize a recommender model's performance.

The training set is used to tune the hyperparameters. Later in this research, the optimized hyperparameters are evaluated on the test set to measure the performance. It is important to make sure that the training set and test set do not overlap in data. In other words, only data from the training set should be used to tune hyperparameters, while data from the test set is used for model evaluation only. In this research, we tune the following hyperparameters: the learning rate *(lrate)* and number of latent factors *(nlf)*. The hyperparameters are tuned for 100 iterations and the following values:

- Learning rate *(lrate)*: [0.005, 0.0075, 0.01, 0.025, 0.05, 0.075, 0,1, 0.15, 0.2];
- Number of latent factors *(nlf)*: [5, 10, 20, 25, 30, 50].

The results of hyperparameter tuning heavily depend on the dataset used for tuning. Since this thesis focuses on the impact of data minimization on the performance of recommender models, it was decided to use training sets of different lengths for hyperparameter tuning. The creation of training sets of different lengths will also be discussed later on (Section 5), but an overview is already provided in Figure 4.9.

This figure shows the 3-fold cross validation that is used. Within each fold, different window lengths are considered in order to create unique training sets. The window lengths represent how many segments are considered for the training set, based on recency of rating data. For example, a window length of 2 indicates that the two most recent segments are considered in the training set. A segment is created simply by dividing the temporally ordered dataset into different parts. The window length is represented by the light grey cubes in Figure 4.9 and can take a value ranging from 1 to 7. The dark grey cube represents the test set and has a fixed window length of 1.

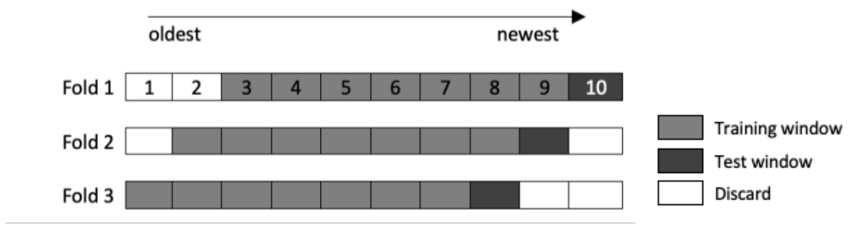


Figure 4.9: Overview of the training sets created based on a 3-fold cross validation

Results of hyperparameter tuning are represented in Figures 4.10 and 4.11 for *lrate* and *nlf*, respectively. Figure 4.10 shows that the RMSE decreases as the learning rate *(lrate)* increases. However, it can be observed that from *lrate* = 0.05 the decrease in RMSE is not significant anymore. Therefore, we set the learning rate fixed at *lrate* = 0.05. The value set for the learning rate does not heavily impact the performance of the recommender systems across the three folds. Based on the RMSE scores in Figure 4.11, one can see that, again, the number of latent factors (*nlf*) has only little impact on the performance of the recommender system across the three folds. However, since the computation time for training our model increases with the number of factors, we set the number of latent factors fixed at 20. From *nlf* = 20, the decrease in RMSE is negligible.

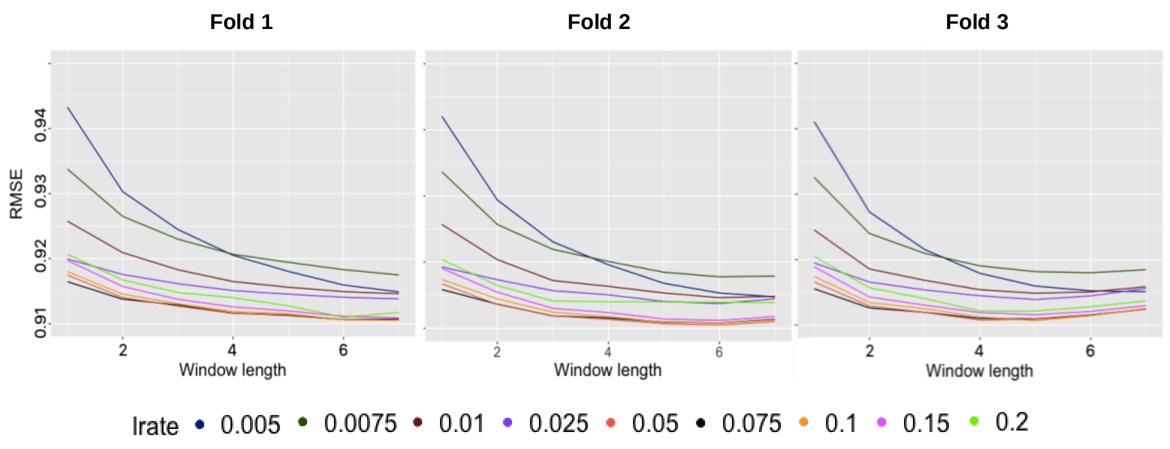


Figure 4.10: Hyperparameter optimization on the learning rate for *nlf*=20

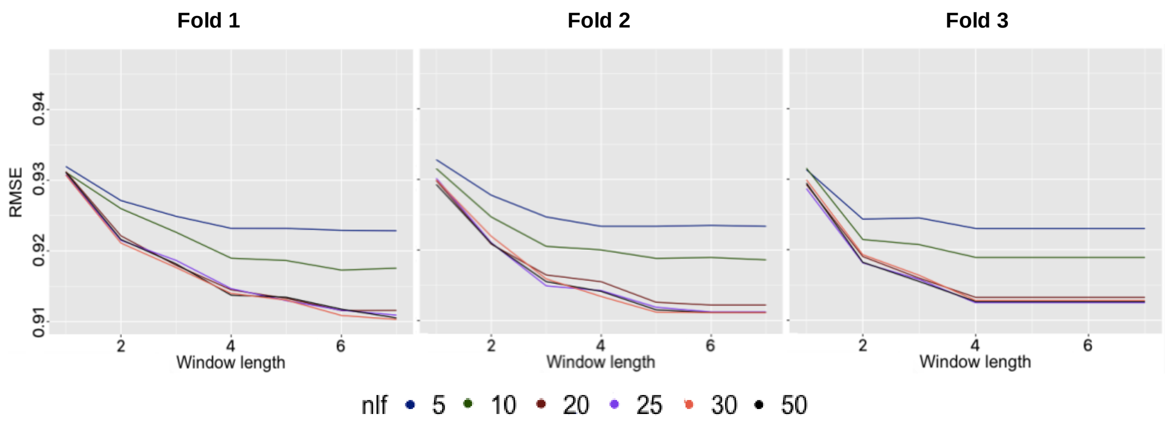


Figure 4.11: Hyperparameter optimization on the number of latent factors for *lrate*=0.05

5. Sliding window technique

This section addresses the first data minimization technique applied in this research; the sliding window technique. This technique distinguishes three different strategies: (1) the temporal sliding window strategy, (2) the random sliding window strategy, and (3) the contextual segments sliding window strategy. For each of the strategies, a detailed description will be provided along with the corresponding summary statistics of the training sets in the following subsections. Finally, the corresponding results will be provided and discussed.

5.1. Temporal sliding window

The first minimization technique involves data minimization based on a sliding time window. This technique is based on a technique introduced by Larson et al. (2017), which maintains the temporal order of the ratings. This technique works as follows: the user-item matrix is first ordered temporally such that a set of ratings between two points in time forms a temporal window. Then the ordered matrix is split into *N* temporal segments of equal size. The oldest ratings belong to the first windows while last window contains the most recent ratings. In order to mimic reality, the last window will be used as test set and the remaining windows will logically be used as training set.

Before creating the training and test sets, there is one important thing to consider first: the number of users who have become inactive at a given date, i.e. the date on which a user has rated his/her last item. The number of inactive users is reasonably large before the year 2002, which results in the fact that many users who became inactive before 2002 will not be included in the test set. In order to reduce this number of users not being evaluated, all ratings before 2002 will be discarded, see Figure 5.1. After the time span has been narrowed to the period 2002-2009, a final dataset containing 5,470,837 ratings is obtained.

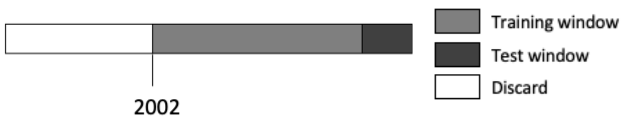


Figure 5.1: Selection of training and test data with ratings before 2002 discarded

This sliding window technique makes use of a three-fold cross validation (CV). Each fold of the CV consists of a different window split into a training and test set, see Figure 5.2 for clarification. From this figure can be seen that for each fold the window slides backward towards older rating data. Consequently, fold 1 contains more recent rating data compared to folds 2 and 3. Within each fold, the most recent ratings will be assigned to the test set in order to mimic reality.

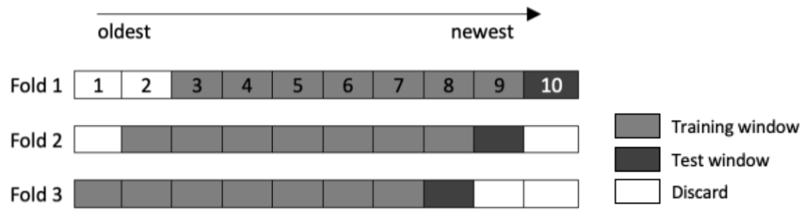


Figure 5.2: Overview of the 3-fold sliding time window

Next to a three-fold cross validation, this techniques makes use of varying history lengths (i.e. the number of segments considered) to create training sets of different sizes. In order to determine these different window lengths for the training set, the data will first be split into N segments. In this research, the number of splits will be set at N = 10. A representation of the segments can be found in Figure 5.2, in which the numbers 1 to 10 represent the segments.

Figure 5.3 shows the representation of the sliding window with different history lengths based on the ten segments created. Within each of the three folds, seven training sets of different history lengths have been created. The smallest training sets have a history length of one, while the largest training sets have a history length of seven. In this figure, the first number between brackets indicates the fold, whereas the second number indicates the history length. For instance, (3,7) refers to fold number 3 and indicates that seven segments are considered for the training data. By limiting the history length to seven, we follow research of Larson et al. (2017). In the end, 21 training sets and 3 test sets are created. The corresponding summary statistics of the 21 training sets can be found in Table 5.1.

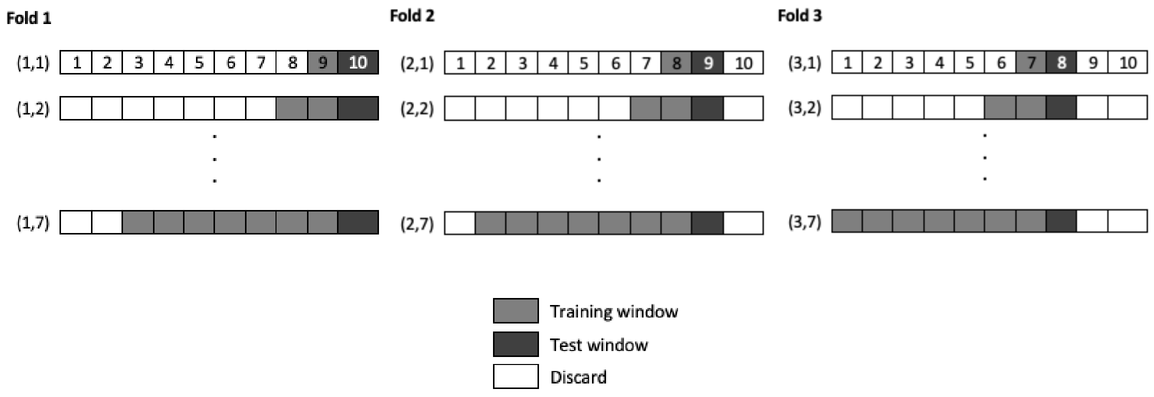


Figure 5.3: Representation of sliding window based on three folds

Table 5.1: Statistics of the training sets based on temporal truncation

History Length	Fold	Users ∈ Tr	Items ∈ Tr	% Rated
1	1	5,414	8,957	1.13
	2	5,355	8,929	1.15
	3	5,368	8,187	1.24
2	1	8,571	9,591	1.33
	2	8,512	9,216	1.39
	3	8,299	8,629	1.53
3	1	11,482	9,704	1.47
	2	11,191	9,306	1.58
	3	10,895	8,810	1.71
4	1	14,066	9,743	1.60
	2	13,721	9,358	1.70
	3	13,262	8,944	1.84
5	1	16,567	9,767	1.69
	2	16,043	9,413	1.81
	3	15.720	9,067	1.92
6	1	18,856	9,797	1.78
	2	18,472	9,469	1.88
	3	18,341	9,121	1.96
7	1	21,261	9,834	1.83
	2	21,077	9,498	1.91
	3	21,208	9,162	1.97

The first column in Table 5.1 ‘History Length’ denotes the number of segments considered for the training set, which are temporally ordered. For example, a history length of 2 implies that the two segments containing the most recent ratings are considered for the training set. The second column ‘Fold’ denotes which fold of the 3-fold CV is applied. Figure 5.3 depicts the overview of the corresponding training and test windows belonging to the three folds. The third and fourth columns in Table 5.1 show the number of users and items that belong to the specific training set, respectively. The fifth column ‘% Rated’ refers to the percentage of movies in the training set that has been rated by users in the training set. A score of 1.13% rated means that on average, users have rated 1.13% of all items.

5.2. Random sliding window

As discussed earlier in this section, data considered for the training sets is minimized based on recency of ratings (i.e. the oldest ratings will be removed first from the training set). In order to further investigate the impact of this sliding-window technique introduced in subsection 5.1, a comparison will be made between recency versus quantity of data in the training set. The results obtained by the temporal sliding window technique will be compared to the results obtained when data is discarded randomly. Doing this, we investigate whether recent data is more relevant compared to ‘random’ data and whether deletion of the oldest data indeed impacts the model performance in a positive way. The results will prove whether or not the performance of a recommender model could be increased simply by removing random data rather than specifically the oldest data.

The corresponding statistics of the training sets based on random truncation of data can be found in Table 5.2. From these statistics can be seen that the number of unique users in the training sets has increased. This might be caused by the fact that in the former technique, there are many ‘old’ users who have not rated any items in recent years. By truncating data based on recency, those old users will be discarded. In contrast, by truncating data randomly, it is likely that at least some ratings of these users will be kept in the training set and therefore increases the number of users. The number of items in the training set is somewhat larger, while the percentage of movies that has been rated by a user on average based on random truncation is somewhat lower compared to truncation based on oldest data. This can be largely explained by the number of users which is substantially larger.

Table 5.2: Statistics of the training sets based on random truncation

History Length	Fold	Users ∈ Tr	Items ∈ Tr	% Rated
1	1	20,991	9,092	0.29
	2	20,780	8,788	0.30
	3	20,877	8,451	0.31
2	1	21,156	9,505	0.54
	2	20,961	9,211	0.57
	3	21,095	8,833	0.59
3	1	21,207	9,641	0.80
	2	21,010	9,321	0.84
	3	21,145	8,990	0.86
4	1	21,223	9,737	1.06
	2	21,041	9,405	1.11
	3	21,163	9,073	1.14
5	1	21,243	9,781	1.32
	2	21,055	9,464	1.37
	3	21,192	9,116	1.42
6	1	21,256	9,811	1.57
	2	21,067	9,473	1.64
	3	21,200	9,146	1.69
7	1	21,261	9,834	1.83
	2	21,077	9,498	1.91
	3	21,208	9,162	1.97

5.3. Contextual segments sliding window

In addition to the temporal and random sliding window strategies, a contextual aspect will be added to the algorithm. The context in which a rating was made will be emphasized in this strategy since context plays a crucial role in determining user behavior. Research of Bashiri (2018) stated that the utility of a certain item to a user may depend significantly on contextual information, in particular on the day of the week. Therefore, a distinction is made between weekdays and weekends.

This technique is an adaption of the former sliding window technique and is based on partitioning windows into multiple segments that best represent a specific context of time. Like the traditional technique, the full dataset is first temporally ordered. A 3-fold CV is again performed based on seven different history lengths, which results in 21 unique training sets and 3 unique test sets. Then the dataset is subdivided into contextual segments retrieved from the implicit rating data. Finally, for each of the segments, a $user \times item$ explicit rating matrix is transformed. In order to rationally evaluate the performance of the different contextual segments, the same test set is used for evaluation.

In this research we investigate context $\subseteq$ weekday/weekend. See Figure 5.4 for the distribution of ratings per day of the week and Table 5.3 for the corresponding statistics of the training sets. Logically, the weekend training sets contain less ratings compared to the weekday training sets as there are more weekdays than weekend days. The goal of this experiment is to investigate the impact of considering only relevant segments (or: the context in which a specific rating was made) on the performance of recommender systems. In order to make reliable comparisons between the different techniques, the number of factors is again set as 20 and the learning rate is set as 0.05 as discussed in section 4.3.1.

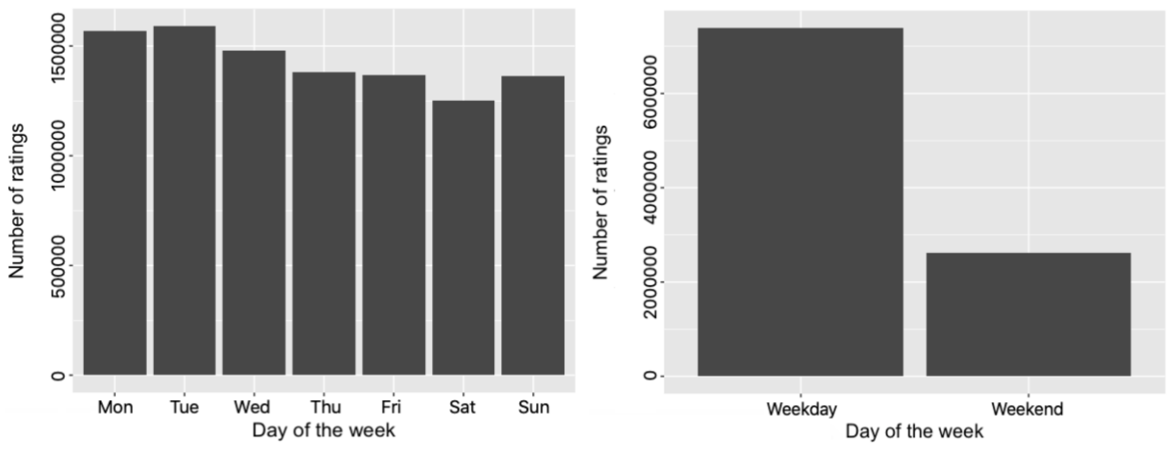


Figure 5.4: Distribution of ratings per day of the week

Table 5.3: Statistics of training sets based on contextual segments

History Length	Fold	Users ∈ $Tr_{Weekday}$	Users ∈ $Tr_{Weekend}$	Items ∈ $Tr_{Weekday}$	Items ∈ $Tr_{Weekend}$
1	1	4,683	3,017	8,608	7,259
	2	4,648	2,864	8,478	7,710
	3	4,635	2,803	7,905	6,654
2	1	7,421	4,591	9,348	8,694
	2	7,386	4,420	8,976	8,337
	3	7,291	4,193	8,445	7,578
3	1	9,955	5,978	9,544	9,008
	2	9,784	5,664	9,141	8,639
	3	9,586	5,096	8,663	7,945
4	1	12,257	7,153	9,632	9,188
	2	12,020	6,525	9,231	8,787
	3	11,646	6,153	8,830	8,273
5	1	14,464	7,991	9,677	9,289
	2	14,033	7,545	9,320	8,940
	3	13,792	7,243	8,992	8,546
6	1	16,446	8,992	9,728	9,395
	2	16,154	8,613	9,411	9,094
	3	16,023	8,420	9,067	8,703
7	1	18,543	10,050	9,787	9,508
	2	18,364	9,775	9,457	9,181
	3	18,480	9,658	9,120	8,782

5.4. Results

This subsection will visualize and describe the performance of the first technique, the sliding window technique, and its three different strategies: (1) temporal sliding window, (2) random sliding window and (3) contextual segments sliding window. This technique is performed in order to investigate the impact of both rating recency and considering different window lengths for the training sets on the performance of recommender models. The first technique truncated data based on the temporal order of ratings, i.e. the temporally oldest ratings of the dataset have been removed from the analysis. A 3-fold cross validation was performed which slides backward in time. Results of this temporal sliding window model are shown in Figure 5.5.

From this figure can be observed that the first fold of the NMF model ('Fold 1') performs best as it has the lowest RMSE scores, followed by fold 2. Fold 3 shows subsequently the worst performance. Fold 1 contains the absolute most recent data of all folds (as shown in Figure 5.2), followed by fold

2, while fold 3 contains the 'oldest' data. These results imply that considering more recent data for the training set leads to a better performance and thus making more accurate predictions.

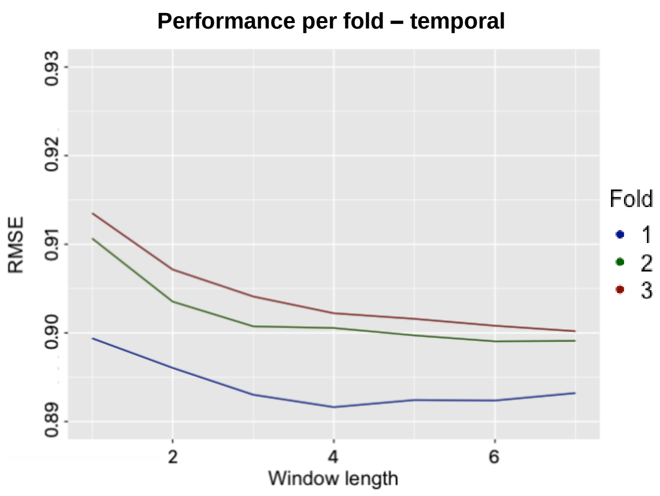


Figure 5.5: Performance of the temporal sliding window model for the three folds

However, an important step when using K-fold cross validation is to check the variances among the ratings across the three folds. By doing this, one can see whether ratings are more similar for some folds. If ratings have become more similar for a specific fold, the RMSE will be lower as a result of this higher similarity and not necessarily because the model outperforms another. Table 5.4 shows the corresponding variances for the ratings. One can clearly see that fold 1 comes with the lowest variances. These lower variances imply that ratings are more similar within this fold, compared to folds 2 and 3. This fact might have caused the RMSE scores for fold 1 to be lower than for folds 2 and 3.

Table 5.4: Variances of the ratings across the three folds

Window length	**Fold 1**	**Fold 2**	**Fold 3**
1	1.044	1.063	1.047
2	1.054	1.055	1.041
3	1.052	1.049	1.057
4	1.048	1.059	1.059
5	1.056	1.060	1.063
6	1.058	1.063	1.079
7	1.061	1.077	1.091

When looking at the pattern of each fold (Figure 5.5), it can be seen that the decrease in RMSE is flattening. From a window length of 4, the decrease in error is minimal. For the first fold there is even a small increase after the fourth fold. This implies that considering a window length of 4 (i.e. considering older data) or higher does not lead to better recommendations. One can conclude based on these results, that recent data is more important for making accurate recommendations compared to less recent data and that more data is not necessarily better. However, in addition, one must keep in mind that differences in variances across the folds might have affected the RMSE scores.

In order to verify the results obtained by the temporal model, a random truncation technique is applied to the dataset. This technique works the same way, but truncates data based on a random basis rather than the temporal recency of ratings. By truncating random data and comparing these results to the previous results, one can verify whether truncating the oldest data improves the performance of the model rather than truncating simply random data from the model. Results of this random sliding window model are shown in Figure 5.6.

The results obtained by random data truncation show a reasonably similar pattern of RMSE scores over the three folds. However, one can see that the RMSE scores are slightly higher compared to the temporal model. This indicates that truncating just random data rather than temporally old data does not increase the performance of the recommender model. One can conclude that the recency of ratings is indeed of importance for the performance of a model when truncating data.

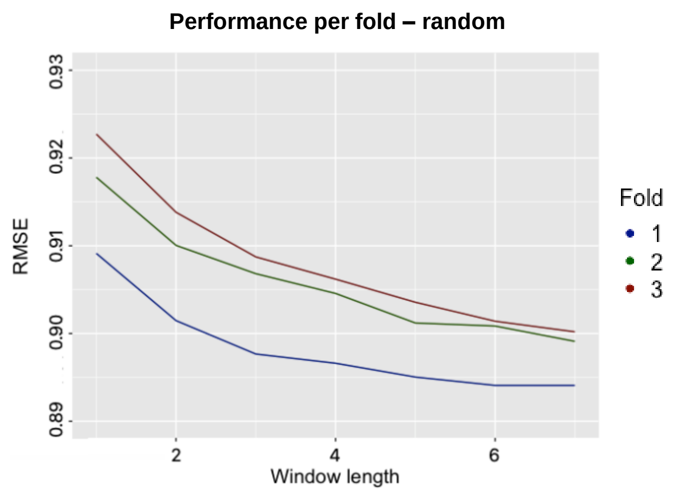


Figure 5.6: Performance of the random sliding window model for the three folds

In order to investigate whether context plays an important role in movie rating predictions, a contextual model was applied which distinguishes ratings made on weekdays and ratings made on weekends by creating contextual segments. Results are shown in Figure 5.7.

Looking at the patterns of the errors over the three folds, one cannot observe clear differences compared to the previous two models. Again, we can see that the performance of the model decreases as the fold number increases. This implies that fold 1 leads to a better performance compared to folds 2 and 3. Looking at the absolute values of the errors, one can see that the contextual model outperforms the model based on random truncation. Though, compared to the temporal model, the RMSE scores of the contextual model are higher which indicate a worse performance.

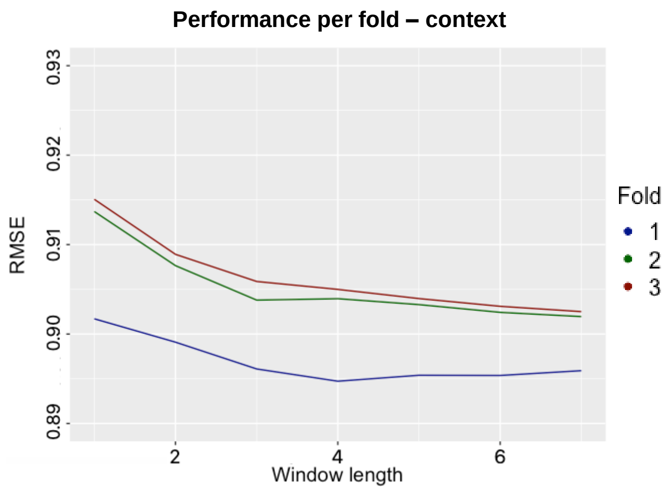


Figure 5.7: Performance of the contextual sliding window model for the three folds

Since a K-fold cross validation is performed, the sum of the errors across the folds will also be visualized to provide more insight into model performance differences. Figure 5.8 clearly shows that, based on the sum of the errors across the three folds, the temporal model has the lowest RMSE score and thus outperforms the other models. When considering context, the model improves compared to random truncation from a window length of 4, but does not outperform the temporal model in which context is not taken into consideration. This effect is possibly caused by the fact that when creating contextual segments, the number of ratings within the training sets decreases drastically. This reduction in size might impact the performance of the recommender model in a negative way, as less information can be used to making rating predictions, compared to the models in which no segments were created.

The contextual model shows that from a window length of 4, the RMSE scores fall below the RMSE scores of the random model. Also, as an increasing decline in RMSE can be observed, it might be possible for the contextual model to outperform the temporal model if a window length >7 would have been used. It is suspected that from this window length, ‘sufficient’ data is considered in the training sets, with the result that the contextual model outperforms the other.

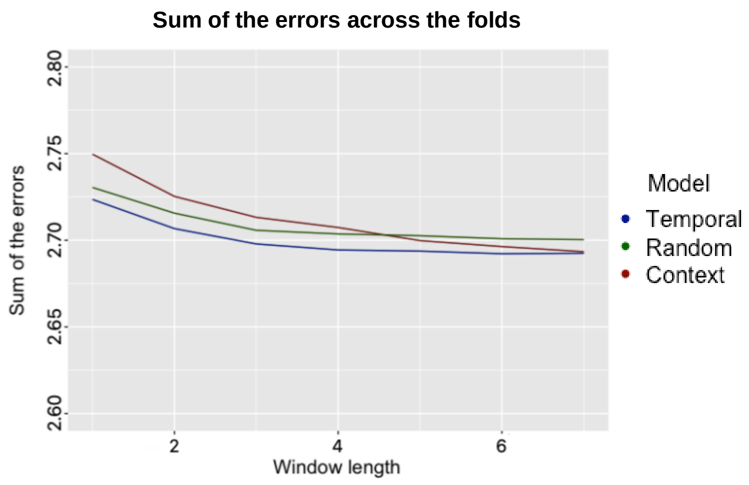


Figure 5.8: Sum of the errors across the folds

If more segments had been considered in the training sets, the RMSE of the contextual model could have fallen below the temporal model, indicating that incorporating contextual model indeed impacts the performance positively. Though, this suspicion will be further investigated in the next section.

When looking at the different window lengths considered for the training sets, it can be seen that there is a small decrease in RMSE as the window length increases. However, the temporal and random model show that from a window length of 4, the decrease is negligible. This implies that considering a training set with a window length of 4 or higher, the performance will not increase significantly while it increases the computation time. The obtained results suspect that considering more and older data is not necessary for making more accurate rating predictions. The contextual model, however, shows a continued decrease in RMSE scores.

6. User profile truncation

This section addresses the second data minimization technique applied in this research; the user profile truncation technique. This technique distinguishes four different strategies: (1) the temporal profile truncation strategy, (2) the random profile truncation strategy, (3) the contextual segments profile truncation strategy, and (4) the item truncation strategy. For each of the strategies, a detailed description will be provided along with the corresponding summary statistics of the training sets in the following subsections. Finally, the corresponding results will be provided and discussed.

6.1. Temporal profile truncation

This temporal profile truncation technique is based on research of Krishnaraj (2019). In this research, a minimization technique was presented which truncates the training set based on a certain predefined user profile length. Similar to the first technique, this technique will also be applied to the temporally ordered rating matrix. A major difference is that the complete rating matrix will now be split only once into a training and test set of 90% and 10%, respectively. Again, the most recent ratings belong to the test set and the remaining ratings belong to the training set. In this technique, no sliding window will be used, which is contrast to the former technique. Focus lies on the length of user profiles rather than the total length of the data. The users that exceed a certain number of items rated are called 'long users'. For these long users, the most recent N ratings will be retained while older ratings will be discarded from the training set.

Like the sliding-window technique (section 5), all data before 2002 will be discarded. After removing these ratings, the dataset contains ratings from 29,800 users for 10,666 items. A minimum number of ratings per user of 1 is obtained, with a maximum of 7047 ratings per user and a mean of 184. An item is rated at least one time with a mean of 513 and a maximum of 15,724 times.

The final training set which consists of 90% of the temporally first ratings has a size of 4,923,753 ratings, while the test set contains the 547,084 most recent ratings. The training set contains ratings from 26,708 users while the test set contains ratings from users 5,533 users. The number of ratings per user (= profile length) will determined by the truncation length. Short users' profile lengths will not be affected in this technique, while long user profiles will be truncated based on different truncation lengths. In this research, the following thresholds for profile length will be investigated: 10, 20, 50, 100, 200, 500, 750, 1000 and the full training set. The corresponding statistics can be found in Table 6.1.

The first column 'Truncation Length' denotes the profile length a long user will be limited to. For instance, a truncation length of 20 implies that for all long users, only the 20 most recent ratings will be retained, while all other ratings will be discarded. For those users who do not exceed a certain threshold for truncation (also called 'short' users), all ratings will be considered. 'Full' truncation length in Table 6.1 implies that the full training set will be used and that no ratings will be discarded. The second and third columns denote the number of users and items in the training sets, respectively. The fourth column denotes the total number of ratings in the training sets and the final column '% Rated' shows the percentage of items that has been rated by a user on average.

Table 6.1: Summary statistics of user profile training sets based on temporal truncation

Truncation Length	**Users ∈ Tr**	**Items ∈ Tr**	**Ratings ∈ Tr**	**% Rated**
10	26,708	8,213	265,250	0.12
20	26,708	8,878	527,648	0.22
50	26,708	9,408	1,168,112	0.46
100	26,708	9,644	1,937,645	0.75
200	26,708	9,758	2,920,582	1.12
500	26,708	9,841	4,187,386	1.59
750	26,708	9,857	4,560,764	1.73
1000	26,708	9,868	4,731,060	1.80
Full	26,708	9,882	4,923,753	1.87

Note that because of rating truncation, the percentage of movies that have been rated by users ('% Rated') of the training sets will decrease sharply for small truncation lengths as more data is being truncated while the number of users is unchanged.

6.2. Random profile truncation

Similar to the first 'sliding window' technique as discussed in section 5, training data in this 'profile truncation' technique is minimized based on recency of ratings. However, where the first technique focused on the overall recency of the ratings, the 'profile truncation' technique focuses on user profiles. In particular: focus is especially on the number of ratings and recency at the user level. This means that for each user, the number of ratings and the corresponding recency will be considered. For example, as discussed in section 6.1, the user profile truncation technique truncated data based on a predefined length and recency of each single user's ratings.

In order to further investigate the impact of this user profile truncation technique, a comparison will be made between recency versus quantity of data in the training set. This random profile truncation technique makes use of similar truncation lengths (which limit the number of ratings considered for long users). In contrast, data is discarded randomly. For example, take a truncation length of 20. For the temporal technique, this meant that only the 20 most recent ratings would be retained for each long user, while all other ratings will be discarded. For the random technique, this means that 20 random ratings will be retained for each long user, while all other ratings will be discarded. Thus, data will be discarded at the user level, but recency is not taken into account as ratings will be discarded randomly.

Subsequently, the results obtained by this random technique will be compared to the results obtained when data is discarded based on recency of ratings. The corresponding statistics based on random deletion of data can be found in Table 6.2. Note that the number of ratings in the training sets remain unchanged compared to the temporal truncation technique, while the number of items has decreased slightly. The levels of '% Rated' of the training sets based on random truncation are approximately equal to the training sets based on temporal truncation. In order to make reliable comparisons between the different techniques, the number of factors is set as 20 and the learning rate is set as 0.05, as was discussed in section 4.3.1.

Table 6.2: Summary statistics of user profile training sets based on random truncation

Truncation Length	Users ∈ Tr	Items ∈ Tr	Ratings ∈ Tr	% Rated
10	26,708	7,591	265,250	0.13
20	26,708	8,481	527,648	0.23
50	26,708	9,222	1,168,112	0.47
100	26,708	9,525	1,937,645	0.76
200	26,708	9,671	2,920,582	1.13
500	26,708	9,804	4,187,386	1.60
750	26,708	9,840	4,560,764	1.74
1000	26,708	9,844	4,731,060	1.80
Full	26,708	9,882	4,923,753	1.87

6.3. Contextual segments profile truncation

In addition to the temporal and random user profile truncation technique, a contextual aspect will be added to the algorithm. Similar to section 5.3 which covers the contextual sliding window technique, this technique is also based on partitioning windows into multiple segments that best represent a specific context of time.

In this context-aware technique, the dataset is first temporally ordered, and then all long user profiles will be truncated based on a predefined truncation length. Truncation of long users is based on recency of ratings. This implies that for a truncation length of 100, for all long users only the 100 most recent ratings will be retained in the training set. For those users who do not exceed a certain threshold for truncation, all ratings will be considered. After long users have been truncated, each set of ratings in the training sets will be partitioned into different segments based on the context in which ratings were made. Again, we investigate context $\subseteq$ weekday/weekend. For each of the nine training sets and the test set, two segments will be created to partition the ratings that were made in a different context. See Table 6.3 for the corresponding statistics.

Table 6.3: Summary statistics of user profile truncated training sets based on contextual segments

Truncation Length	**Users $\in$ Tr$_{\text{Weekday}}$**	**Users $\in$ Tr$_{\text{Weekend}}$**	**Items $\in$ Tr$_{\text{Weekday}}$**	**Items $\in$ Tr$_{\text{Weekend}}$**
10	21,665	9,224	7,812	6,244
20	22,131	10,051	8,557	7,238
50	22,679	11,153	9,198	8,270
100	22,968	11,793	9,505	8,838
200	23,127	12,184	9,654	9,206
500	23,175	12,386	9,788	9,463
750	23,182	12,413	9,812	9,522
1000	23,185	12,424	9,831	9,547
Full	23,185	12,428	9,851	9,626

These statistics show that the number of items in the weekday training sets are slightly higher than the weekend training sets, but the difference becomes smaller as the truncation length increases. Regarding the number of users, the differences are getting bigger, which indicates that there are more unique users who watch movies on weekdays compared to weekends.

6.4. Item truncation

The fourth truncation technique performed in this section is based on items, rather than user profiles. In this technique, the same predefined truncation lengths will be tested, but data truncation is based on the recency of items. Recency of items implies that for each profile length, we truncate the oldest items from the training set and retain the N most recent ones. For instance, a truncation length of 10 implies that for all long users, only the 10 most recent items are retained in the training sets. Table 5.7 shows the corresponding summary statistics. Note that, compared to the statistics of the temporal truncation model, the number of items in the trainings sets has decreased reasonably.

This can be explained simply by the fact that truncation based on items has been performed, which decreases the number of items. As more items have been truncated for smaller truncation lengths, the difference in the number of items becomes larger for smaller truncation lengths.

Table 6.4: Summary statistics of user profile training sets based on item truncation

Truncation Length	Users ∈ Tr	Items ∈ Tr	Ratings ∈ Tr	% Rated
10	26,708	7,014	265,250	0.14
20	26,708	8,378	527,648	0.24
50	26,708	9,288	1,168,112	0.47
100	26,708	9,570	1,937,645	0.76
200	26,708	9,735	2,920,582	1.12
500	26,708	9,824	4,187,386	1.60
750	26,708	9,843	4,560,764	1.73
1000	26,708	9,856	4,731,060	1.80
Full	26,708	9,882	4,923,753	1.87

6.5. Results

This section visualizes and describes the performance of the second technique, the user profile truncation technique, and its four different strategies: (1) temporal profile truncation, (2) random profile truncation, (3) contextual segments profile truncation and (4) item truncation. This technique truncates user data based on a predefined user profile length. Compared to the first technique which truncated data based on the overall recency of ratings, data is now truncated at the user level. Results of these four strategies are visualized in Figure 6.1.

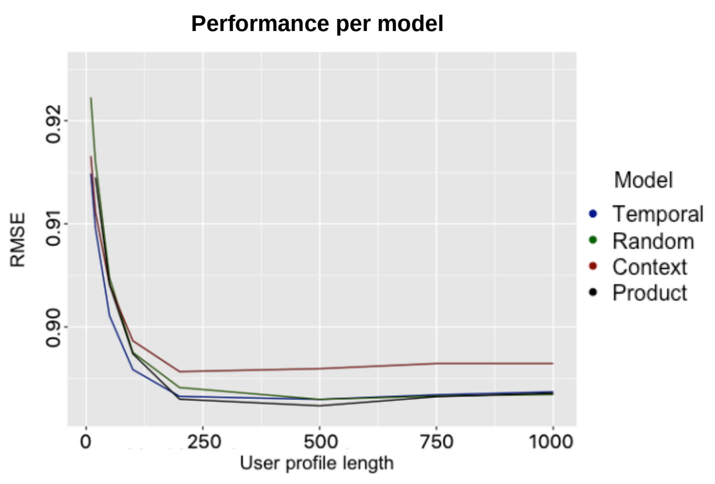


Figure 6.1: Performance of the user profile truncation strategy for the three models

The results demonstrate that the temporal model which truncates the temporally oldest user profile data, has slightly lower RMSE scores compared to the other models. However, from a user profile length of 200, the decrease in RMSE is negligible. This indicates that considering more than 200 ratings for each user does not increase the model performance for rating prediction. The model which performed truncation on a random basis has higher RMSE scores for small user profile lengths compared to the temporal truncation model. For larger user profile lengths, the errors are almost equal to the temporal model.

The model which incorporates contextual information by creating different segments underperforms the two other models as it shows significantly higher RMSE scores from a user profile length of 100. Again, this underperformance might be caused by the fact that the contextual model divided the original training sets into smaller segments. As the size of the training sets decreases, less information is provided, which may have negatively impacted the performance. This suspicion can be verified by looking at the prediction performance for the weekday and weekend segments separately. Doing this, we can indeed observe somewhat lower RMSE scores for the weekday segment compared to the weekend segment. Thus, for this context can be concluded that segments of larger size (= weekday) outperform smaller segments (= weekend) as more accurate information is taken into account for rating prediction. See Figure 6.2 for the corresponding RMSE scores.

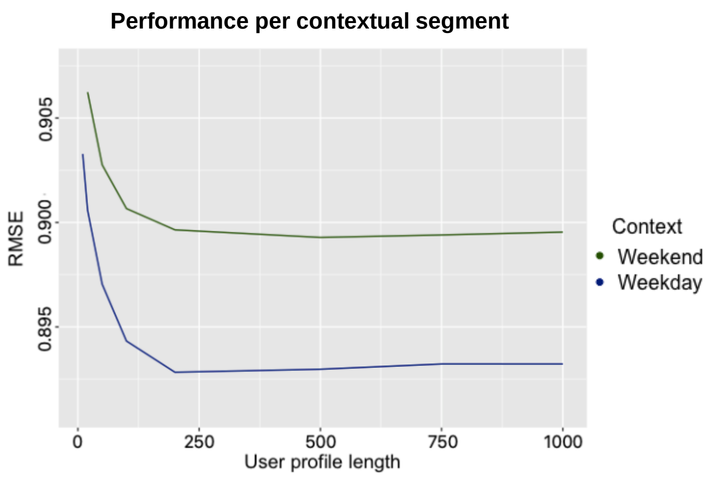


Figure 6.2: Performance of weekday and weekend segments separately

The fourth strategy that belongs to this technique truncated data based on the recency of items, i.e. truncated the oldest items from the training set. For small user profile lengths (0-200 ratings), the performance of the item truncation model is quite similar to the temporal and contextual models as the RMSE scores are approximately equal (see Figure 6.1). For larger user profile lengths (i.e. >200 ratings per user), one can observe that the model based on item truncation slightly outperforms the other models. This implies that truncating old items from the training data leads to more accurate rating predictions. One can deduce from these results that it might be better not to recommend old items to users. Also, it is better to truncate old items from the training set, rather than old user ratings.

7. User profile split

This section addresses the third data minimization technique applied in this research; the user profile split technique. This technique distinguishes three different strategies: (1) the activity-based profile split strategy (2) the contextual segments profile split strategy, and (3) the context-based profile split strategy. For each of the strategies, a detailed description will be provided along with the corresponding summary statistics of the training sets in the following subsections. Finally, the corresponding results will be provided and discussed.

7.1. Activity-based profile split

The third data minimization technique aims at minimizing user profile lengths by splitting them rather than dropping user data. A result of this technique is an increase in the number of users. Compared to the sliding window (section 5) and the user profile truncation (section 6) techniques, this third technique discerns itself since it does not minimize or discard the amount of data used for the training set. But, at the user level, data will be minimized since new users will be created to which a part of the rating data will be assigned. These 'additional' users created by segmenting are called pseudo-users and do not overlap in data. Several strategies can be used in order to split long user profiles meaningfully. Baltrunas & Amatriain (2009) proposed a time-aware split strategy in which user profiles are split based on the context in which a rating was made. Their idea is based on partitioning user profiles into multiple profiles that best represent the user in a particular context of time, such as morning, weekend and summer.

In this activity-based profile split strategy, the impact of time duration between ratings will be investigated, which is based on research of Krishnaraj (2019). Time duration between any two ratings is referred to as the 'activity gap' and is used as measure to create the pseudo user profiles. These activity gaps are used in order to anticipate changes in user preferences over time. In this technique, new pseudo users are created based on time duration between ratings since user preferences could have changed after a long period of inactivity. By applying this strategy, changes in user preferences will be accommodated into pseudo users by reassigning rating data. This strategy is applied by using a minimum profile length together with an activity gap of at least 8 weeks to determine the split point. Setting a minimum length for pseudo-user profiles precludes long profiles being split into (too) many small pseudo profiles. This strategy results in pseudo-user profiles of different sizes. A threshold of 8 weeks to determine a split point implies that a user has to be at least 8 weeks inactive, i.e. not rating any movie, in order to be split into a pseudo user.

This technique works as follows: first, short and long users will be determined. A user is considered 'long' if its number of ratings exceeds a certain threshold. Logically, the remaining users are considered 'short'. Short users will be appended to the training set directly. For long users, the activity gaps are considered for splitting the profile. If there is an activity gap of >8 weeks within a long user, and the minimum profile length can be maintained after the split, then the long user profile will be split and new pseudo-users are created consequently. If the minimum profile length cannot be maintained by splitting a long user, then the user will not be split at that point. Note that a long user can exhibit several activity gaps, which implies that a user can be split more than once (provided that the minimum length will be maintained).

In the rest of this experiment, we set the threshold for long users fixed at 1000. This threshold indicates the minimum number of ratings before a user can be considered 'long'. A threshold of 1000 ratings per user results in a total of 536 long users. We use the following values of minimum length after splitting for the new user profiles: 100, 200, 300, 400 and 500. An overview of the pseudo-users created based on the context-aware profile splitting technique can be found in Table 7.1.

Table 7.1: Overview of pseudo-users created from 536 long users based on the activity-based profile split

Min. Profile Length	**Pseudo-users**
100	952
200	746
300	658
400	603
500	565

Similar to the previous experiments, data before the year 2002 is excluded from this experiment since the number of inactive users is very high in this period. Data of the first 7 years (1995-2001) therefore increases the number of users and items that are no part of the test set.

The division of the data into a training and test set follows a procedure that has already been used in this research in previous sections. Based on the temporally ordered matrix, the first 90% will be used for the training set, while the 10% most recent ratings will be used for the test set. An overview of the summary statistics belonging to this splitting strategy can be found in Table 7.2. These statistics show that for the 'pseudo' model, the number of users has decreased drastically. This is a logical consequence of the fact that in the 'pseudo' model, only pseudo users are taken into account, while the short users (which are the majority) are disregarded. Subsequently, the 'full' model includes both pseudo and short users when evaluating the performance of the recommender system.

Table 7.2: Summary statistics of the activity-based splitting strategy

Model	Min. length	Ratings ∈ Te	Items ∈ Te	Users ∈ Te	Users ∈ Tr
Pseudo	100	72,515	8,924	209	880
Pseudo	200	70,035	8,957	162	700
Pseudo	300	68,427	8,973	145	618
Pseudo	400	67,108	8,957	136	567
Pseudo	500	66,664	6,938	131	535
Full	100	547,084	9,355	5,533	27,088
Full	200	547,084	9,355	5,533	26,896
Full	300	547,084	9,355	5,533	26,817
Full	400	547,084	9,355	5,533	26,765
Full	500	547,084	9,355	5,533	26,732

7.1.1. Results

In order to analyze the ‘traditional’ splitting strategy and be able to further improve the performance, the results obtained by this strategy will be discussed first. The corresponding results are shown in Figure 7.1.

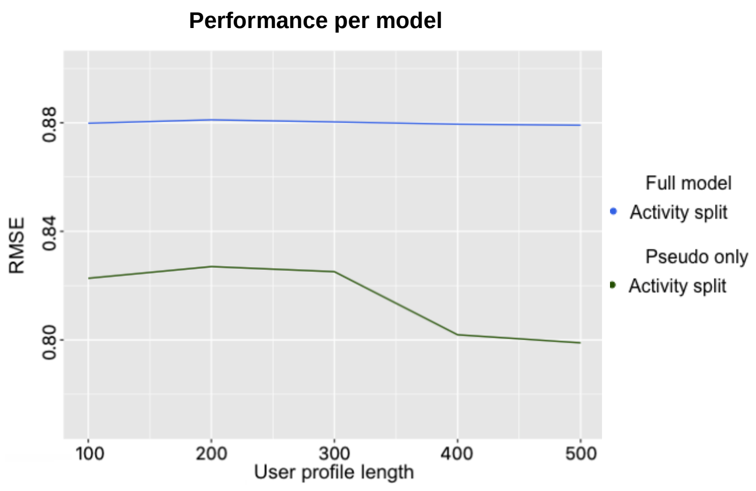


Figure 7.1: Performance of the pseudo and full model for different user profile lengths

Results demonstrate that the effect of splitting user profiles based on activity gaps has very little impact on the performance when tested against the full model. Results of the full model show an approximately flat line, indicating that the performance is not affected over the different user profile lengths. Considering a profile length of 100 versus a profile length of 500 does not have an impact on the prediction performance when tested against both short and pseudo users (full model). However, the overall model performance has improved compared to previous techniques, represented by decreased RMSE scores. When only pseudo users were evaluated, more interesting results can be observed. Tested against pseudo users only shows significantly lower RMSE scores compared to when tested against the full model. Also, an increase in performance can be observed as the minimum length increases, which is represented by a decrease in RMSE. These results show that splitting can improve the effect of recommender systems, but when considering the full test, the impact of splitting profiles is negligible. Also, it was demonstrated that pseudo users with a minimum profile length of 500 perform significantly better compared to a minimum length of 100. Based on this, one can conclude that it is best not to split long profiles into too many pseudo users.

7.2. Contextual segments profile split

As results of the activity-based splitting strategy (section 7.1) have proven; the proposed technique of splitting user profiles solely based on activity gaps does not significantly improve the performance of recommender models. To follow up on this strategy, we further investigate the impact of splitting strategies on recommender systems. This is done by creating different segments based on the context in which ratings were made. In order to investigate whether this contextual strategy improves the performance of recommender models, a contextual technique is applied in the same way as presented by the activity gap splitting strategy. Similar to the strategies proposed in sections 5.3 and 6.3, this technique is based on partitioning windows into multiple segments that best represent a specific context of time.

This contextual segments profile split strategy builds further on the activity-based strategy which investigated the impact of time duration between ratings. That strategy was applied by using a minimum profile length together with an activity gap of at least 8 weeks to determine the split point. By splitting long users based on activity gaps, new pseudo users have been created. The same procedure will be executed for the contextual segments profile split strategy. After this is performed, all users in the training set will be subdivided into contextual segments retrieved from the implicit rating data. Again, we investigate context $\subseteq$ weekday/weekend. For each of the segments, a $user \times item$ explicit rating matrix is transformed. The goal of this experiment is to investigate the impact of considering only relevant contextual segments on the performance of recommender systems, in addition to the activity-based splits. Table 7.3 shows the corresponding summary statistics.

Table 7.3: Summary statistics of the contextual segments splitting strategy

Strategy	Min. length	Items ∈ Te	Items ∈ Te	Users ∈ Te	Users ∈ Te
		Weekday	Weekend	Weekday	Weekend
Pseudo	100	8174	6576	203	172
Pseudo	200	8230	6563	159	137
Pseudo	300	8193	6647	143	126
Pseudo	400	8128	6693	134	120
Pseudo	500	8117	6663	129	114
Full	100	8807	7618	4707	2871
Full	200	8794	7588	4680	2849
Full	300	8763	7580	4661	2840
Full	400	9741	7577	4652	2834
Full	500	9747	7569	4646	2828

7.3. Context-based profile split

To follow up on both the activity-based splitting strategy and contextual segments profile splitting strategy, we further investigate the impact of splitting strategies on the performance of recommender systems. In this subsection, a context-based profile splitting strategy will be applied. As its name already suggests, this technique also incorporates contextual information. However, in contrast to the context-aware profile split technique which partitioned data based on the context of ratings (section 7.2), this technique is not based on creating contextual segments. This technique does not create smaller training and test sets, but rather uses the full dataset when training and evaluating the model. Here, a strategy similar to the activity-based splitting strategy is applied. Where the activity-based splitting strategy (section 7.1) splits long users based on temporal gaps between ratings, this strategy focuses on the context.

This technique works as follows. First, for each long user a distinction will be made between ratings made on weekdays versus ratings made on weekends. Second, ratings will be assigned to new pseudo users, which are created based on this distinction. This means that ratings made on weekdays will still belong to the original users, while ratings made on weekends will be assigned to 'pseudo' users. This results in a doubling of the number of long users, since for each unique long user, a pseudo user will be created (provided that each long user has rated items on both weekends and weekdays). Short users will be appended to the training set directly.

The main difference between this technique and the aforementioned technique which created contextual segments, is that in this technique all user data is taken into consideration. When creating contextual segments, the complete training set will be divided into smaller training sets that only contain ratings made in a particular context. Now, the complete training set is used for training the model, but data is reallocated to pseudo users.

Similar to the activity-based splitting strategy, for each long user (a user with >500 ratings) a pseudo user is created based on the context in which a rating was made. For this strategy, it was decided to use a value of 500 ratings instead of 1000 for considering long users as with many datapoints, estimating context-specific preferences might be feasible with much less data. Again, we investigate context $\subseteq$ weekday/weekend. This implies that each long user is split into a weekend and weekday pseudo user. Like the previous techniques, the following values of minimum length after splitting are tested: 100, 200, 300, 400 and 500.

The number of new pseudo-users created is an exact doubling of the number of long users (2474 long users) for each minimum profile length, since for each unique long user one new pseudo-user has been created. This implies that both conditions are met; each long user has rated items on both weekends and weekdays, and that the minimum profile length has been maintained. Note that for the context-based splitting strategy, long users will only be split once (weekday versus weekend), while for the activity-based splitting strategy, it is possible for long users to be split into multiple pseudo users.

7.4. Results

This section visualizes and describes the performance of the third technique. The third technique splits long users into smaller pseudo users based on activity gaps or context. For the activity-based splitting strategy, a long user is split into a pseudo user if there is a gap of at least 8 weeks between two ratings. This technique accounts for the size of the newly created users by applying a minimum for the user profile length. In contrast to the aforementioned techniques in sections 5 and 6, there is no data truncation in this technique. Rather, data is assigned to new pseudo users if, within user data, gaps between ratings exceed 8 weeks. The first splitting technique was solely based on activity gaps, while the second technique incorporates the context in which a rating was made by creating segments within the training data. Third, a splitting strategy is applied which creates pseudo users based on the context in which a rating was made. An important difference with creating segments is that here, all data is taken into account while training the model; the training data is not segmented.

Figure 7.2 visualizes the results obtained. The ‘Full model’ denotes the performance of the recommender model when the full data (short users + pseudo users) is considered. ‘Pseudo only’ indicates the performance when only pseudo users are considered. Results show that when the full model is used for evaluation, the effect of splitting users has only little impact on the performance of the recommender model. There is no significant difference in accuracy for the different profile sizes.

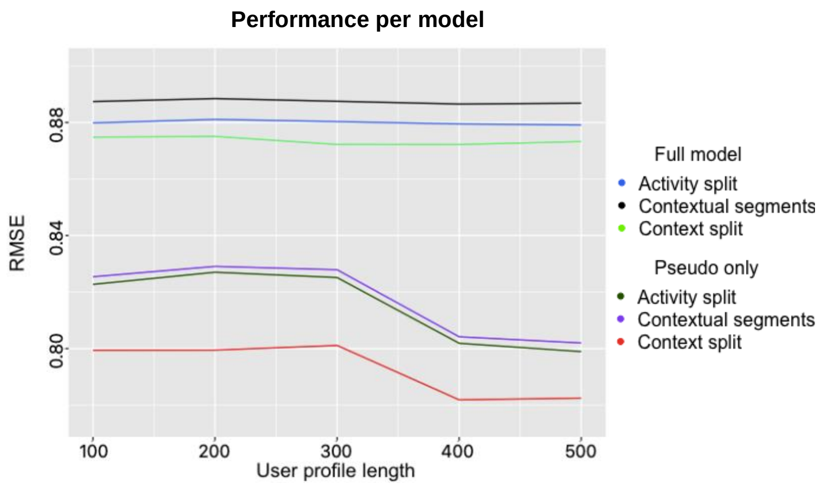


Figure 7.2: Performance of the user profile split strategy for the six models

The activity-based splitting strategy which created contextual segments underperforms all other models as it shows the highest RMSE scores. The ‘original’ activity-based splitting stategy which did not incorporate contextual information performs slightly better. However, the splitting strategy based on context performs best.

When these three techniques are tested against the test set with pseudo users only, an increase in performance can be observed. The RMSE scores are significantly lower compared to the full model results. Also, it can be observed that for all three strategies tested against the pseudo users, the RMSE scores decrease for a minimum profile length of 400. From a profile length of 500, the RMSE seems to be approximately constant. When comparing the results of the three different strategies, an approximately similar pattern of RMSE scores can be observed. Again, the activity-based splitting strategy which created contextual segments underperforms the other two models. Here also applies that the best performing strategy is the third one; the context-based splitting strategy. It is striking that the difference in RMSE is substantially bigger for the pseudo model than the difference in performance for the full model, indicating that the context-based splitting strategy has more impact on the performance when tested against the ‘pseudo model’.

Shortly, the results show that creating segments does not enhance the performance of the recommender models. Instead, splitting based on context seems to be an interesting strategy as it reduces the RMSE scores significantly. Also, results have shown that after a user profile length of 400, the decrease in RMSE is negligible.

8. Conclusion and future work

This thesis examined the impact of different data minimization techniques on the performance of a model-based recommender system. The Non-negative Matrix Factorization (NMF) algorithm was used to train the recommender model and MovieLens 10M was used as dataset to experiment on. This thesis started by giving an overview of related literature about data minimization techniques. The methodology section described how Collaborative Filtering (CF) techniques work and especially the NMF algorithm. A detailed description and analysis of the dataset was provided, along with the evaluation metrics and tuned hyperparameters in order to maximize the performance of the recommender system. Subsequently, three different frameworks were described and analyzed in detail. Where the previous sections provided an extensive analysis and description of the results, this section will be used to give an overview and draw conclusions from the findings for each of the techniques performed.

8.1. Overview of techniques

This subsection provides an overview of the three different techniques applied in this research. Each of the techniques can be subdivided into different strategies, see Table 8.1.

Table 8.1: Overview of the frameworks applied and the corresponding strategy

Technique	**Strategy**	**Description**
Sliding window	1 Temporal 2 Random 3 Contextual segments	Training sets are created by applying a 3-fold cross validation of varying history lengths based on recency of ratings (or randomly). Test set is fixed and contains 10% most recent ratings.
User profile truncation	1 Temporal 2 Random 3 Contextual segments 4 Item	Long user profiles are truncated and only the most *N* recent ratings (or items in strategy 4) are retained. Test set is fixed and contains 10% most recent ratings.
User profile split	1 Activity gap 2 Contextual segments 3 Context	Long user profiles are split into new pseudo-users based on gaps between ratings (or context). Test set is fixed and contains 10% most recent ratings.

8.2. Conclusion

Technique 1: Sliding window

This technique constructed temporal windows of unequal lengths to vary the size of the training sets. The evaluation of the model performance is based on a three-fold cross validation method. The findings demonstrated that the NMF algorithm performs best when the most recent ratings are considered, while the performance worsens as more older data is considered in the training sets. It was also proven that the accuracy of the recommender system does not continue to increase as more data is considered. On top of that, it can be concluded that incorporating contextual information in this case does not lead to a higher accuracy. However, differences in variances across the folds might have affected differences in prediction performance. More research is needed in order to draw substantive conclusions.

Technique 2: User profile truncation

This technique truncated training data at the user level. For all users with a profile size larger than the threshold, the oldest data will be truncated. Based on the findings, it can be concluded that considering more than the 200 most recent ratings per user does not significantly increase the accuracy of rating predictions. Also, it is better to truncate data temporally than randomly, which confirms that recency of ratings is indeed of importance. Incorporating contextual information by creating segments does only improve accuracy of the recommender system from a user profile size of 500 or higher. Finally, data was truncated based on recency of items. From these results can be concluded that it is better to truncate old movies rather than old ratings.

Technique 3: User profile split

This technique split long users into smaller pseudo users. The first splitting strategy is based on temporal gaps between any two ratings and splits users if there are activity gaps of at least 8 weeks. The second technique created contextual segments based on implicit rating data, while the third technique incorporated contextual information by splitting long users based on the context of their ratings. Results show that if the full model is considered for evaluation, there is no significant difference in accuracy for the three techniques. Also, it was proven that creating contextual segments leads to a deterioration in the performance of the recommender system and underperformed all other techniques. When the NMF algorithm is evaluated against pseudo users only, the splitting strategy based on the context of ratings shows the most interesting results.

The goal of this thesis was to analyze different techniques for minimizing data and the requirements of training data for recommender systems. In conclusion, this study has shown that minimizing training data does not deteriorate performance of recommender systems and that incorporating more data at the user-level does not necessarily improve the performance. For all techniques, it was proven that creating (weekend/weekday) segments in order to incorporate contextual information was the least effective technique. Meanwhile, the most interesting results were provided by the user profile splitting strategy, in which new users were created based on the context of their ratings. In this strategy, all data is kept, but at the user level data is minimized as ratings have been reassigned to new pseudo users. The results have shown that at the user level, using less data does not deteriorate prediction performance.

8.3. Future work

This thesis has demonstrated that data minimization does not necessarily have a negative impact on the performance of recommender systems. Since this topic has not been studied widely in literature, more research needs to be done. This section provides some recommendations for future work in data minimization for recommender systems:

- This thesis experiments on a sample of the MovieLens 10M dataset which contains explicit user input. A recommendation for future work is to experiment on other datasets that contain implicit data such as purchase data;
- In view of running time, it was decided to use a subsample of the dataset containing ratings from the last 8 years. By doing this, data is already minimized, which might have affected the performance of the frameworks. A possibility for future work is to experiment on the full dataset or even larger MovieLens datasets (MovieLens 20M or MovieLens 25M);
- This thesis focuses on incorporating contextual information to data minimization techniques. However, only a distinction was made between ratings made on weekends and weekdays. It might be possible that users have watched a movie and rated that movie not until a few days later. A possibility for future work could be to focus on another definition of context, such as seasons, hour of the day, etc. and investigate its impact on the performance of the NMF algorithm;

- Multiple studies in literature have focused on the NMF algorithm and presented several adaptions of this algorithm. A direction for future work might be to dive into these adaptions of NMF in order to investigate whether this could enhance recommender systems;
- This thesis is limited to rating prediction only. Possibilities for future work lie in examining the task of top-N recommendations;
- The experiments in this thesis are solely evaluated based on statistical accuracy metrics. A possibility for future work could be to focus on other metrics and investigate the impact of data minimization techniques on the performance.

References

Amatriain, X., Jaimes, A., Oliver, N., & Pujol, J.M. (2011). Data mining methods for recommender systems. Recommender systems handbook, 39-71.

Baltrunas, L., & Amatriain, X. (2009). Towards Time-Dependent Recommendation based on Implicit Feedback. In Workshop on Context-aware Recommender Systems in conjunction with the 3rd ACM Conference on Recommender Systems.

Bennett, J., & Lanning, S. (2007). The Netflix Prize. In Proceedings of KDD Cup and Workshop.

Bashiri, P. (2018). Recommender systems: Survey and Possible Extensions.

Campos, P., Bellogin, A., Diez, F., & Chavarriaga, J. (2010). Simple time-biased KNN-based recommendations. In Proceedings of the Workshop on Context-Aware Movie Recommendation.

Campos, P., Diez, F., & Cantador, I. (2014). Time-aware recommender systems: A comprehensive survey and analysis of existing evaluation protocols. User Modeling and User-Adapted Interaction, 24, 67-119.

Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. Intelligent Data Analysis, 21(6), 1487-1524.

Goddard, M. (2017). The EU General Data Protection Regulation (GDPR): European regulation that has a global impact. International Journal of Market Research, 59(6), 703-705.

Gordea, S., & Zanker, M. (2007). Time Filtering for Better Recommendations with Small and Sparse Rating Matrices. In Web Information Systems Engineering, Springer Berlin Heidelberg, 273-284.

Hu, Q., Yu, D., & Xie, Z. (2006). Information-preserving hybrid data reduction based on fuzzy-rough techniques. Pattern Recognition Letters.

Intersoft Consulting. (2018). General Data Protection Regulation (GDPR), Articles 5, 15 and 17. Retrieved from: https://gdpr-info.eu/.

Isinkaye, F.O., Folajimi, Y.O., & Ojokoh, B.A. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16, 261-273.

Jolliffe, I. T. (1986). Principal components in regression analysis. Principal component analysis, 129-155.

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques For Recommender Systems.

Krishnaraj, M. (2019). Comparative Analysis of Techniques for Data Minimization for Recommender System Algorithms.

Larson, M., Zito, A., Loni, B., & Cremonesi, P. (2017). Towards Minimal Necessary Data: The Case for Analyzing Training Data Requirements of Recommender Algorithms.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755), 788-791.

Matloff, N. (2016). Quick Introduction to Nonnegative Matrix Factorization.

Pathak, B., Garfinkel, R. Gopal, R., Venkatesan, R., & Yin, F. (2010). Empirical analysis of the impact of recommender systems on sales. J Manage Inform Syst, 27(2), 159-88.

Resnick, R., & Varian, H.R. (1997). Recommender systems. Commun. ACM, 40(3), 56-8.

Richthammer, C., & Pernul, G. (2018). Situation awareness for recommender systems. Electronic Commerce Research, 1-24.

Steeg, F. V. (2015). Context-aware recommender systems. Master's thesis.

Wen, H., Yang, L., Sobolev, M., & Estrin, D. (2018). Exploring recommendations under user-controlled data filtering. Proceedings of the 12th ACM Conference on Recommender Systems, 72-76.