Åbo Akademi University

A Study on Recommendation System with Library Data

Master's Thesis in Computer Science

Umesh Raj Satyal
Supervisor: Dr. Mats Neovius
2-6-2019

Abstract

The recommendation is a difficult task in the lack or absence of domain knowledge such as facts and information of the item which is the subject of the recommendation. This study presents and compares algorithms used in building a recommendation system that uses only the information from the artifact loan log of a library. The library has different materials available to lend, such as books, CDs, DVDs, which as a whole will be represented as artifacts in this thesis. This system is not limited to artifacts from the library but could have a vast range of usability for example in supermarkets.

The thesis introduces the reader to the recommendation system followed by an overview of recommendation problem formulation, interpretation of recommendation problem as data mining problem, different techniques and algorithms used in recommendation system and validation techniques for those algorithms. A case study is performed by using the library dataset obtained from Vantaa City Library. The objective of the case study is to test different recommendation algorithms with the dataset. The algorithms used are based on Collaborative Filtering. The models are evaluated using 5-fold cross-validation and the evaluation metrics used are Root Mean Square Error(RMSE) and Mean Square Error(MSE).

Keywords: Recommendation system, Collaborative Filtering, Matrix Factorization

Contents

1 Introduction

1.1 Background and motivation

The recommendation is, in simple terms, the event of suggesting a subject. The subject being recommended varies with its kind. The sources of recommendation can sometimes be reviews from other users, letters and, at other times just a verbal recommendation from the friends. A recommendation system, similar to the one described in this study, automates the process of recommending. It takes the observations as input and helps the user to make a decision by making a model from those observations. The model builds on observations of the user's past behavior, the nature of the item recommended, the user's demographic status and the contextual aspect to predict items that the user might possibly be looking for. For new users, recommendation on those mostly depends upon their similarity with the existing users.

The study of the recommendation system can be understood in a similar way as any decision-making process so as to make a recommendation. The present world demands that the recommendation is automated. Different machine-learning algorithms and statistical tools are used to form a recommendation model. It can also be studied as a general data mining process.

At the time of writing this thesis, recommendation systems are frequently used to present items to users in the case of digital businesses such as *Amazon*[1], *eBay*[2], *Netflix*[3] , and *Youtube*[4]. Amatriain (2014) has shown that two-thirds of the movies watched on Netflix are recommended by the Netflix recommendation engine. Of all the google news read, around thirty-eight percent is recommended through the recommender system. Thirty-five percent of sales in Amazon is through recommendation [1].

[1] https://www.amazon.com/
[2] https://www.ebay.com/
[3] https://www.netflix.com/
[4] https://www.youtube.com/

1.2 Scope and objective

The main objective of this thesis is to study, analyze and compare different algorithms used in the recommendation system for artifacts of Vantaa City Library, a public library in the city of Vantaa. This thesis deals with offline data from the library. The recommendation system in general aims to help the users of the library to find interesting artifacts to borrow by providing a recommendation of possibly interesting artifacts to those users. There are a few sub-objectives, such as dealing with the new users, analyzing the effect of the contextual factor to the recommendation system and finding the subset of users in an unbiased and representative way.

The approach used in a recommendation system varies for existing and new users who may join the system later. For the existing users, their past behavior is observed and passed as input to different models and evaluated. The information provided is utilized in the optimal possible way with available computational capacity.

The case study is performed in python. The *Jupyter Notebook*[5] (formerly known as IPython notebook) is used as an interactive computation environment which enables combining code execution, rich text, mathematics, plots, and rich media together in the same notebook. The data used in the case study is obtained from Vantaa City Library situated in the city of Vantaa, Finland.

1.3 Structure of the thesis

The thesis is divided into five chapters. Chapter 1 is an overview of the study. Chapter 2 introduces the theory behind the recommendation system. Chapter 3 contains the case study in which a recommendation system is built using a library dataset. Chapter 4 presents the result of the case study. Chapter 5 discusses the result and finally concludes the thesis.

[5] https://jupyter.org/

2. Recommendation system

2.1 Recommendation problem

A recommendation problem is to estimate a utility function whose output is a prediction that predicts the level to which a user will like an item based on his past behavior, relation to other users, item similarity, context.

Let *C* be the set of all users, *S* the set of all possible recommendable items, *u* be the utility function measuring the usefulness of item s to user c, i.e. u: C $\times$ S $\rightarrow$ R, where R is the totally ordered set. For each user c belongs to C, choose items s $\in$ S that maximizes the utility function *u* is to be chosen to recommend [2].

The formal definition of the utility function described above is presented as:

$$\forall\, c \in C, s'c = argmax(u(c,s))s \in s.$$

2.2 Recommendation as a data mining process

A recommendation problem can be studied as a data mining process. A typical data mining process contains three steps: data preprocessing, data analysis and result interpretation as shown in figure 1. A recommendation system deals with real-life data which are generally inconsistent and unmanaged. Therefore, the data needs some preprocessing, such as data cleansing and filtering the unwanted inputs from the data. For example, every customer with *null* as their address is assigned a *NaN* record for disregarding this later.

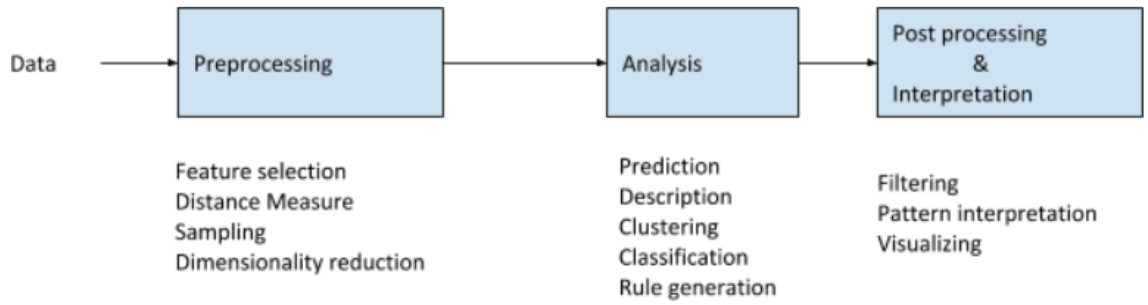

Figure 1: Data mining in a recommendation system

The main theme of the traditional recommendation system approaches is recommending the similar item that the active user has previously observed and the items observed by other users who are similar to the active user. For this purpose, the similarity between users (and items) needs to be calculated. One type of measure is the distance between them. Some of the distance measurement metrics include Euclidean distance, Minkowski distance, Mahalanobis distance, Cosine distance, Jaccard distance [3]. These distance metrics used in the recommendation system are briefly described below.

2.2.1 Euclidean distance**:**

This approach is widely used to calculate distance. Euclidean distance is the root of the summed square of the difference between coordinates.

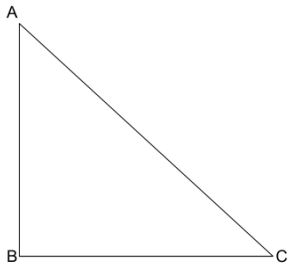

Figure 2: Visualizing Euclidean distance in two dimension

Figure 2 shows a right-angled triangle, where the length of AB is the difference between the y coordinates of points A and B. The length of BC is the difference between the x coordinates of points B and C. The length of AC is the root of summed square of AB and BC.

Euclidean distance is defined as:

$$\boldsymbol{d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}}$$

Here, n is the number of dimensions (features), x_k the k^{th} attributes of the data objects x and y_k the k^{th} attributes of data objects y.

Table 1: An example of user-item matrix

Item/ User	i_1	i_2	i_3	i_4
u_1	4	NA	2	2
u_2	2	5	2	2
u_3	1	NA	NA	3
u_4	5	3	3	4

Suppose there are four users and four items. Each of the users has rated some of the items. A pivot table is created by filling the cells of the table with the value $R_{ij.}$ R_{ij} refers to the rating given by the user i (u_i) to the item j (I_j) as shown in table 1. The cells with missing values are represented by NA. The Euclidean distance between the items i_1 and i_4 is calculated as:

$= \sqrt{(4-2)^2 + (2-2)^2 + (1-3)^2 + (5-4)^2}$

$= \sqrt{4+0+4+1}$

$= \sqrt{9}$

= 3 units

2.2.2 Minkowski distance

It is a generalized metric distance defined as:

$$d(x,y) = \left(\sum_{k=1}^{n} |x_k - y_k|^r\right)^{1/r}$$

Here, r is the degree of distance. When the value of r is 1, Minkowski distance is known as city block distance. The city block distance is also known as Manhattan distance, taxicab distance or L1 norm distance. When the value of r is 2, the Minkowski distance is known as the Euclidean distance. Likewise, when the value of r tends to infinity the Minkowski distance is known as the supremum (Lmax norm or L∞ norm) distance.

When the value of r is 3, the Minkowski distance between i_1 and i_4 from table 1 can be calculated as shown below:

$= (|4-2|^3 + |2-2|^3 + |1-3|^3 + |5-4|^3)^{1/3}$

$= (8 + 0 + 8 + 1)^{1/3}$

$= (17)^{1/3}$

$= \sqrt[3]{17}$

= 2.57 units

2.2.3 Cosine distance

A cosine distance is a measure of orientation between two vectors. While calculating cosine distance, items are considered as vectors of an n-dimensional space and the distance is computed from the angle between them.

$$cos\,(x,y) = \frac{(x.y)}{||x||\,||y||}$$

x.y = the dot product of vectors x and y

||x|| = the norm of vector x

$||y||$ = the norm of vector y

The cosine distance between two the items i_1 and i_4 from table 1 can be calculated as shown below:

$i_1.i_4 = 4 \times 2 + 2 \times 2 + 1 \times 3 + 5 \times 4 = 8 + 4 + 3 + 20 = 35$

$||i_1|| = \sqrt{4^2 + 2^2 + 1^2 + 5^2} = \sqrt{16 + 4 + 1 + 25} = \sqrt{46} = 6.78$

$||j_1|| = \sqrt{2^2 + 2^2 + 3^2 + 4^2} = \sqrt{4 + 4 + 9 + 16} = \sqrt{33} = 5.74$

$\cos(i1, i4) = \frac{35}{6.78 \times 5.74} = 0.9$

Cosine distance between $i_1.i_4$ is 0.9.

Since the cosine distance is the measure of cosine of the angle between two non-zero vectors, it measures the orientation between those two vectors, but does not measure the magnitude unlike the Euclidean distance. The dataset used for the recommendation system can be highly sparse. In this case, measuring the orientation or the pattern of the data points might produce better results than measuring the magnitude between them. Using a metric that measures the magnitude, such as the Euclidean distance, can bias the data points that are far from each other.

2.2.4 Jaccard distance

The Jaccard distance is based on the common preferences between users. For instance, the items that the users have borrowed could serve as such preference. Suppose there are two sets X and Y. The elements of the sets X and Y are the books borrowed by two users. To calculate the Jaccard distance, the ratio between the intersection and the union of those two sets is multiplied by 100. The Jaccard distance is commonly known as the Jaccard index.

$$Jaccard\ (X, Y) = \left(\frac{|X \cap Y|}{|X \cup Y|}\right)$$

$X \cap Y$ = The common items between the sets X and Y.

X∪Y = All the items between the sets X and Y, without repetition.

The Jaccard distance between the two users u_1 and u_4 from table 1 can be calculated as shown below:

X∩Y = 3

X∪Y = 4

Jaccard(u_1 ,u_4) = 3/4 = 0.75

The choices of the distance measures depend upon the nature of the dataset and the problem. A study, that used a large scale of data from a social networking site, has shown that, the choice of the distance measure does not affect the prediction accuracy of the recommendation system in a general scenario [4].

2.2.5 Treating noisy and incomplete data

A Noise is an unwanted data point that can potentially affect the result. A Noise is also an important factor that needs to be handled in the preprocessing step. For example, a course book may give an impression of being more popular compared to other books. However, this can be a misleading information for the recommendation system as the compulsory courses in school may influence the borrowing of that particular book. A noise can be natural or malicious. While the natural noise comes from the user involuntarily, the malicious noise is deliberately imposed in the system to achieve biased result. A malicious noise undeniably affects the system, which emphasizes the need of measures to stop the noise. A natural noise affects the system less effectively compared to the malicious noise. Requesting some of the users to re-rate the item could benefit the reduction of these noises [6].

Using a computational model with a large amount of data is not only time consuming, but also computationally greatly expensive. This problem could be addressed by selecting a representative subset from the data, which is called sampling in data mining. The Sampling process in data mining is used in both the preprocessing phase and interpretation phase. The data is split into a training

set and a testing set. The training set is used to apply the model and the result is validated in the testing set. The main challenge in sampling is to find the subset which is representative of the whole dataset [5].

A sparsity is when data occurs at widely sparse intervals. In the recommendation system, it might occur that some artifacts are borrowed only once or twice and some are borrowed hundreds of times. The dataset used for the recommendation system can have many features, define a multi-dimensional space, and ultimate result difficult for a human to understand. This is known as the *curse of dimensionality*. The sparsity and the curse of dimensionality are among the problems that a recommendation system often faces. These issues can be solved by using dimensionality reduction techniques. The Principal Component Analysis (PCA), the Singular Value Decomposition(SVD) and the Matrix Factorization(MF) are to name some of the commonly used dimensionality reduction approaches in the recommendation. It is not unusual that the resulting user-item matrix can be very sparse with a lot of zeros even in a simple recommendation setting. For this reason, dimensionality reduction approaches are not only used in preprocessing of the data but also in the main process of building a recommendation system [5].

The Classification is the process of mapping the data between the feature space and the label space. The feature space contains the characteristic of the element and the label space contains the classes. A book can be classified into a bestseller or not a best seller by using the different available features of the books. These features could be the number of copies sold, the number of stars that the book has got in review and so on. There are different types of classifiers and they can be categorized from different perspectives. Supervised and unsupervised classifiers are among them.

When the label space and the feature space are known, the classification is known as supervised. Suppose, we have an input variable A and an output variable B. If we use an algorithm to find a mapping function that maps between the variables A and B, i.e. B = f(A), the algorithm used in this scenario comes under supervised learning. When there is only input data which does not

have any corresponding output, then this is known as unsupervised learning. One of the uses of unsupervised learning is to find the underlying structure of data. There are different algorithms for supervised learning. Some of those algorithms are Nearest neighbors, Decision trees, Rule-based classifiers, Bayesian classifier and Support vector machine. Clustering comes under unsupervised learning. Some of the evaluation metrics for classifiers in recommendation systems are a mean average error (MAE) and root mean squared error(RMSE) [5].

Association rule mining is another data mining technique used in the recommendation system. It finds a rule of occurrence of an item based on the occurrence of other items in the list [5].

2.3 Different categories of recommendation system

This chapter includes a broad categorization of recommendation systems by different techniques. The collaborative filtering and the matrix factorization use the numerical value such as the ratings provided by or inferred from the user. The content-based approach uses the descriptive attributes of the item such as metadata and the hybrid approach uses two or more than two approaches. A hybrid approach can also use demographic information of the user such as age, sex, and locality.

2.3.1 Collaborative filtering

The Collaborative filtering (CF) is one of the widely used approaches for building a recommendation system. It uses past observations of the active user and other users similar to the active user to recommend an item to the active user. The developer of the first recommendation system of the modern era, Tapestry, coined the term collaborative filtering in 1992 [7]. Previous recommendation systems used either *rule-based recommendation system* or *user customization* in their system. The fundamental assumption of the collaborative filtering is that if two users: X and Y rate n number of items similarly or they have similar observation behavior with each other, then they will also rate other items similarly [8].

In order to provide a scale with semantics for observation, a rating is needed. Rating in the recommendation system is the evaluation or assessment of the user's behavior towards an item. The rating can be explicit and implicit. Explicit ratings are obtained when a user rates an item, for example, with thumbs up and thumbs down in *YouTube* or star rating in *Amazon*. In case of implicit ratings, the users are not asked to rate an item. It is rather calculated using the user's observation behavior towards the item observed by the users or the items that are similar. In case of implicit rating, a user profile can be created observing the user's click on websites or the books that the user borrows from a library.

A dataset is generated in collaborating filtering which contains a list of the *m* number of users $\{u_1, u_2, ..., u_m\}$ and a list of n number of items $\{i_1, i_2, ..., i_n\}$ where each user u_i has a list of items Iu_1 which is the list of items that the user has rated or derived from the user observation or preference [9].

Table 1 from chapter 2.2 is a user-item matrix of rating generated randomly that contains four users and four items. The cell M_{ij} of the matrix is filled with the rating provided by user i to item j. Collaborative filtering fills the empty cells of the matrix observing the similarity between two users (or items). User u_1 and u_2 have rated items i_3 and i_4 equally, their observation behavior appears similar to each other although they have rated item i_1 differently. User u_2 has rated 5 to item i_2 but the user u_1 has not rated this item yet. Since they have a similar user profile, it can be predicted that the user u_1 might also rate 5 to the item i_2. Assuming the rating scale is from 1 to 5, 5 being the highest rating, user u_1 might be interested in the item i_2. The item i_2 is recommended to user u_1 in a similar manner. All the empty cells of the matrix are filled and the user will be recommended the item that the user might give better rating value.

There are two types of collaborative filtering named with the techniques they use: memory-based and model-based. The memory-based model uses the user rating data to calculate the similarity between the users and items and that similarity is used in recommendation and prediction. Many commercial web shops including *Amazon* used this technique in the beginning as it is highly

effective and easy to implement [10]. However, the implementation of memory-based collaborative filtering can be highly expensive when the data is very sparse and when the similarity between the items is very difficult to achieve.

The model-based collaborative filtering method was introduced to solve the problems of the above-mentioned kind [11]. The model used in this method can be a data mining or a machine-learning method depending on the nature of the data. Some of the models used are Bayesian belief nets, Clustering models, Latent semantic models, and Markov decision process.

Matrix factorization (MF) is another method used in collaborative filtering which converts a matrix with a higher dimension into two or more matrices of lower dimensions. The method is useful to discover the latent factor underlying the user-item interaction which can be useful to predict the rating in collaborative filtering. This is just the inverse of matrix multiplication. Matrix factorization became popular in the field after Simon Funk's blog post in the Netflix challenge [21]. This technique helps to convert the user-item matrix used in building a recommendation system to lower dimension latent space. An illustration of matrix factorization is shown in the figure below.

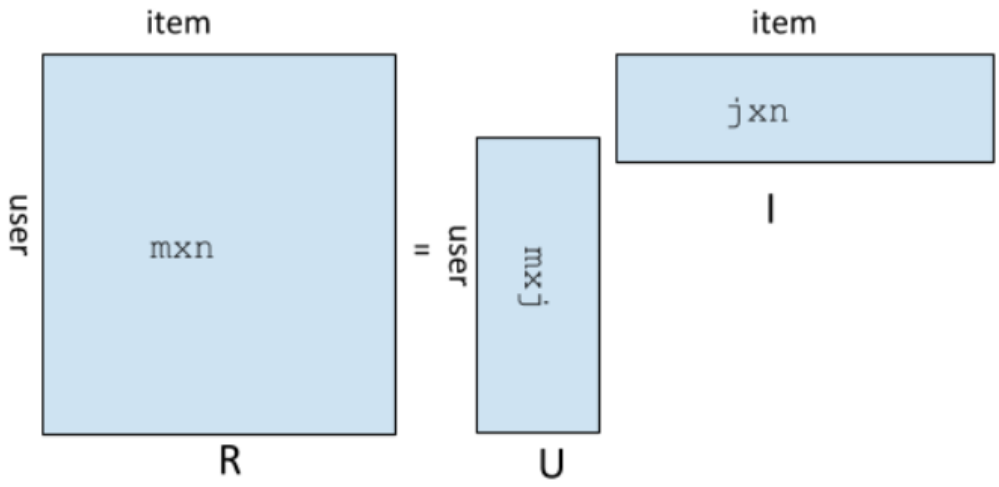

Figure 3: Illustration of matrix factorization

Suppose there is *m* number of users and *n* number of items and *R* is the user-item matrix of order *m×n* filled with the ratings that the user has given to the corresponding item. When the number of users and items are very large, the size of matrix *R* becomes very large. When there are 10 thousand users and 10 thousand items, the number of entries in the matrix *R* becomes 100 million which takes a lot of space and this becomes computationally expensive. Matrix *U* is a matrix of the order *m×j* and matrix *I* is of the order *j×n*. Matrix U and Matrix I are smaller in size than matrix R because these are obtained from matrix *R* by factorizing. If the order of matrix *U* is 10 000 by 5 and the order of matrix *I* is 5 by 10 000, then there will be only 100 thousand entries altogether. This is very small compared to the original matrix *R*. The matrices *U* and *I* contain the latent factors of users and items. Latent factors can be known or unknown but they represent the respective user or item. Machine-learning techniques such as Stochastic gradient descent [25] help to minimize the error while factorizing the matrix for latent features. After factorizing, the predicted rating for user *i* to item *j* is obtained by obtaining the dot product between an i^{th} row of user matrix to a j^{th} column of item matrix.

There are various models for matrix factorization. Some of them are Singular Value Decomposition (SVD), Principal Component Analysis (PCA) and Probabilistic Matrix Factorization (PMF). The matrix factorization method is also known as the latent factor model.

Neighborhood-based is another approach of recommendation using collaborative filtering. It is subdivided into two categories as user-based and item-based. User-based recommendation relies on the opinion of like-minded users. In user-based recommendation the rating r_{ui} for a user u to an item, *i* is calculated using the ratings given to i by users who are most similar to u. These users are called nearest-neighbors of user u. Item-based recommendation depends upon the ratings given to similar items on the list. The process of neighborhood-based recommendation consists of rating normalization, similarity weight computation and neighborhood selection.

The collaborative filtering model is evaluated using metrics that vary with the types of the model used. An example of an evaluation metric for prediction is mean absolute error. For decision accuracy, the ROC curve is also used. This will be discussed later in chapter 2.4 of this thesis.

2.3.1.1 Challenges in collaborative filtering

The main challenges of collaborative filtering are data sparsity, scalability, synonymy, gray sheep, shilling attacks, privacy protection, and serendipity. They are discussed briefly in the following paragraphs.

Data sparsity

A public library dataset can be generally assumed to be a very large dataset. The user-item matrix used for collaborative filtering can be very sparse as the users may or may not want to rate the item explicitly. This makes the prediction for recommendations difficult to achieve.

The sparsity of the data affects the collaborative filtering model at many levels. Many approaches which contain the dimensionality reduction techniques have been proposed to solve the data sparsity problem. Singular Value Decomposition(SVD) is one, which removes the insignificant and unrepresentative users and items, directly from the list. This facilitates the ease to map similarity between the users and item since space is reduced and data is less sparse. Another method Principal Component Analysis (PCA), is an eigentest that reduces the dimension of the dataset preserving the important components. This may, however, remove very important users or items containing very important information [12].

Cold start

When there are new users or items arriving in the system, there is no information available about them. In the absence of rating by new users and purchase or loan history of the user, no item can be recommended to them. Thus, there is a need to wait until the user rates or purchases any item. This problem is called a cold start or a new user/item problem [12].

Scalability

With the increasing number of users and items, the collaborative filtering algorithm needs to scale and that may be time-consuming. Regardless of the situation, the algorithm should have high availability and meet the application's temporal requirements. If the increase in the number of users and items is rapid and very large, heavy computational complexity might make computation impractical [12].

Synonymy

A number of same or similar items might have entered the dataset in different names. This is called synonymy. Most of the recommendation systems treat them as different items and this may decrease in the performance of the system. There are many attempts to solve this problem. One of them includes automatic term expansion or the construction of a thesaurus. It is fully automatic and therefore, not able to deal with the problem properly since there can be the different meanings of the word from the intended one. Some SVD techniques such as Latent Semantic Indexing(LSI) deal with the synonymy problem more effectively but their performance is high only at higher recall levels. When it comes to the lower level recall, the performance is still poor [12].

Gray Sheep

There are some users whose agreement and disagreement related to the items are inconsistent. This makes it difficult to recommend any item to those users and they might not be benefited by the collaborative filtering. A hybrid approach that combines the CF and content-based and other approaches such as demographics might solve this problem to some extent [12].

Shilling Attacks

In an open system that anyone can visit and provide with ratings and reviews, there are possibilities that people cheat by providing many positive ratings to their items and providing negative reviews and low ratings to the items that are similar to their items. This is called Shilling

Attacks. Due to the nature of CF, this will affect the recommendation system. The system based on the CF algorithm should be aware of this sort of attack and have some precaution [12].

Privacy protection

Many users do not want to provide public reviews as they do not want to make their habits and views known publicly. Collaborative filtering utilized system should be fully aware of this and provide the protection of a user's privacy [12].

Serendipity

Serendipity is the occurrence of surprise in a beneficial way. There is a chance that users might also like new items that they do not expect to be in their list but somehow occur there. If a user reads a book that he or she assumed that he/she would never prefer but finds interesting later after reading, this is called as serendipity in the recommendation system. Since collaborative filtering depends upon the user's past behavior, serendipity is a challenge to it [12].

2.3.3 Content-based recommendation system

In a content-based recommendation system, the descriptive attributes or metadata of the item are utilized to form a model of recommendation. Suppose a user A reads a book B which was related to international politics and the user did not rate the book. The rating information related to this, thus, goes missing and the collaborative filtering method cannot be used to recommend another book to the user. However, if books C and D are also related to politics, this information is relevant. This means that A might be interested to read the books C and D and those books can be recommended to A.

In a content-based recommendation, each user is assumed to be unique and assumed that they operate independently and each item are represented by the features of the items. For a book, the features can be writer, description, language and for a movie, the features can be actors, a plot of the movie, director, language of the movie and this data is used as training data to form a

model. The machine-learning methods such as classification or regression is used to train the data to form a model which predicts the item and it will be user-specific. The general architecture of a content-based filtering model is pictorially shown in fig. 3.

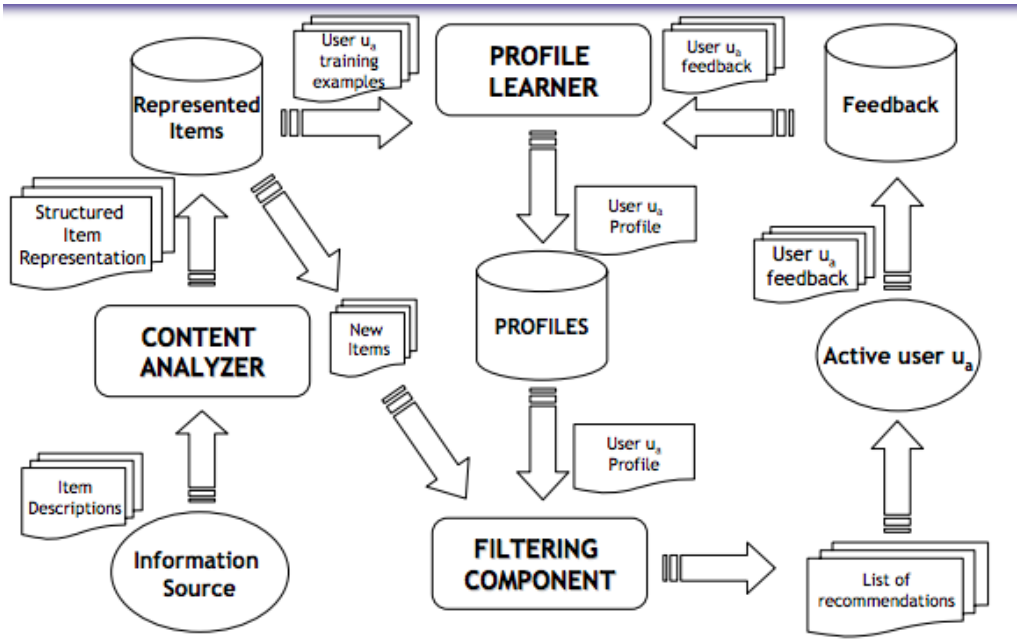

Figure 4: General architecture of content-based model [13]

Content analyzer: Content analyzer helps to extract relevant information from the data source when they are not in any structure. The content analyzer preprocesses the data and makes it ready for another level of operation [13]. Suppose there is a list of books to be recommended and some of the relevant information from the books is the name of the author, category of the book, keywords used to describe the book. This information can be used for the representation of the books and these information are utilized in the following steps such as profile learner and filtering components.

Profile learner: The data coming from the content analyzer is collected and generalized using different machine-learning techniques by the profile learner and it will match them with the user preference and build user profile [13]. For an example, if a content analyzer from books resulted

from a list of authors, a profile learner for a user can be built by looking at whether the user has read the books from the author in the past or not.

Filtering component: This module utilizes the user profile built by profile learner from the previous step to construct a list of recommendation to the user [13].

Unlike collaborative filtering, the content-based approach is user-independent. It uses the history and rating of the active user and utilizes it to build his/her profile. Collaborative filtering uses ratings from the other users to form the nearest neighbors of the active user. Another advantage of the content-based approach over collaborative filtering is that the explicit list of contents that are used to build the model makes the model more transparent and not just a black box. It is easier to find the trustworthiness of the model by comparing the list of items recommended with the user profile. It is also able to recommend new and unpopular items [14].

Domain knowledge is required in content-based filtering, but not necessarily required in collaborative filtering. The content is needed from which the meaningful features can be extracted and encoded and this is not always easy and applicable. The model also faces the problem of serendipity. The model suggests the items that are matched against the profile of the users. As a result, similar items are recommended to the user each time. Enough rating suffices the making of an effective recommendation to the new user. In other words, an adequate amount of rating should be collected for the system to understand user preferences. The model could overfit easily as the model will be able to recommend items also for the new users, though the recommendation may not necessarily in a reliable way in that case. In fact, a few ratings could be more valuable than metadata [14].

2.3.4 Others

Besides the approaches of recommendation system listed in section 2.3.3, there are other approaches such as personalized learning to rank, the demographic approach of recommendation, recommendations by social connectivity and hybrid recommendation system.

In the personalized learning to rank approach, the recommendation problem is studied as a ranking problem using the popularity of the item as the baseline. There are three approaches for learning to rank, namely pointwise, pairwise and listwise. In a pointwise approach, ranking function minimizes the loss function defined on individual relevance judgment. The ranking score is based on classification and regression. In pairwise learning to rank approach, the loss function is defined on pairwise preferences with the goal of minimizing the number of inversions in ranking then the ranking problem is transformed into the binary classification problem. The loss function in listwise approaches is the similarity between the ranking list and ground truth of the item [15].

In the demographic approach, the demographic status of the user is taken into account while building the recommendation system. This approach tries to find the relation between different demographic attributes such as age, sex, address, language, education, and occupation of the users while recommending items.

A recommendation system which consists of two or more approaches is called hybrid recommendation approach.

2.4 Evaluating recommendation systems

After building a recommendation system using one or more methods, the system could be evaluated. There are many approaches for the evaluation of the recommendation system. With a number of choices, the selection of the evaluation method and metric is delicate. Not only the accuracy but robustness and scalability are also the key players for selecting the right algorithm. Also, users might want to try a new taste, that is different from his consumption history considered while modeling. Different properties need to be changed during the process to get a better performance result. There should be a tradeoff between the different properties used to avoid any chances that change in one property might affect the other.

There are two metrics in recommendation algorithms evaluation: evaluation metric and benchmarking. While testing the algorithms with evaluation metric, some items are recommended to the users using the algorithm and checked whether they are really viewed or consumed by the user or not. Recommendation system benchmarking is done by comparing the algorithms with datasets. Results obtained are usually compared with the results obtained while implementing the algorithm in a similar dataset. In the time when the recommendation system was introduced, the prediction accuracy was the only key evaluation of recommendation system. Although this is crucial, this is not sufficient in the time of writing this thesis.

2.4.1 Experimental setting

Before setting the experiment, an experimental hypothesis should be formed. For instance, “algorithm A predicts better than B” could be a hypothesis. There should also be some controlling variables for instance “algorithms A implemented with dataset B and algorithm C implemented with dataset D cannot be compared because there is no clarity whether the algorithm is better or it is just performing better because of the quality of the dataset used”. Thinking beyond only one dataset that was used to build the model and testing it with different dataset will improve the generalization power of the algorithm.

Shani et al (2011) divide experimental settings into user studies, offline experiments and online experiments. In user studies, the behavior and influence of the users towards the recommendation can be studied. User studies are usually conducted as a survey within a group of people asking them a list of questions related to the built recommendation system. User studies need a suitable group formation, so that the user's behavior towards the recommendation system could be read. A-B testing is done with real users and real tasks. Many real-world applications have an online testing environment where they direct some of their data from the system. A-B testing sometimes sends irreverent recommendations to the users. Another approach is an offline experiment. An offline experiment is conducted in pre-collected data. Although the information that data gives will be lesser than the real users in online testing and

user studies, different algorithms can be tested with this setting at a relatively lower cost [16] [17].

2.4.2 Recommendation System evaluation metrics

There are different metrics that help to evaluate recommendation systems. Some metrics that Shani et al (2011) presented in their article are described here briefly.

Prediction accuracy: This is the most discussed property in recommendation system literature. The basic assumption behind it is that if the item predicted is more accurate, the user will prefer that item. This does not care about the user interface and this can be calculated offline. One of the metrics for measuring the prediction accuracy is the root mean squared error(RMSE). This measures the distance of the predicted item from the actual item and shows how close the predicted item is with the real item.

$$RMSE = \sqrt{\frac{1}{|\tau|}\sum_{\langle u,i\rangle\in\tau}(r_i^{-u} - r_i^u)^2}$$

Here, $|\mathcal{T}|$ is the number of user-item pairs $\langle u, i\rangle$ in the test set $\mathcal{T}$. Mean squared error(MAE) is an alternative to RMSE. This approach is similar to RMSE but uses the absolute distance to measure the distance between the predicted item and actual item.

$$MAE = \sqrt{\frac{1}{|\tau|}\sum_{\langle u,i\rangle\in\tau}\left|r_i^{-u} - r_i^u\right|}$$

Here, $|\mathcal{T}|$ is the number of user-item pairs $\langle u, i\rangle$ in the test set $\mathcal{T}$. RMSE penalizes the large errors disproportionately than MAE.

Usage prediction measure: Recommendation system recommends the item for the user to use. In the usage prediction measure, whether the item suggested is used by the user or not is checked. In this study, offline evaluation of usage prediction is important as the study deals with offline data from the library. In an offline evaluation of usage prediction, from the dataset of items

that users have used, a test user is selected. Some choices of that test user are hidden and asked the recommendation system to predict the set of items for that user. Then the result is classified according to table 2. In offline cases, unused items are forced to assume that they would have not been used even if they have been recommended to the user. That means those items are not interesting to the user. This assumption might sometimes be false because the user might not have noticed that the items exist but after the recommendation system recommends the item, a user might be interested in that.

Table 2: Usage prediction scenario

	Recommended	**Not recommended**
Used	True Positive(TP)	False Negative(FN)
Not used	False Positive(FP)	True Negative(TN)

After calculating the numbers of TP, FN, FP and TN values, precision, recall, and false-positive rates are calculated.

precision = $\frac{TP}{TP + FP}$

recall (true positive rate) = $\frac{TP}{TP + FN}$

false positive rate = $\frac{FP}{FP + TN}$

The result is visualized in a curve such as *the Precision-Recall Curve* or *Receiver Operating Characteristic Curve (ROC)*. The curve comparing precision to recall is called *Precision-Recall Curve* and this emphasizes the proportion of recommended items that are preferred. The curve comparing recall (true positive) and false-positive rates is called the *Receiver Operating Characteristic Curve (ROC)*, this emphasizes the proportion of items that are not preferred that end up being recommended.

Ranking measures: In some recommendation model, the recommendation problem is studied as a ranking problem and a list of high ranked items are recommended to the user. The accuracy and relevance of the ranking should be evaluated. This can be evaluated using *reference ranking*, *utility-based ranking* and this can be also evaluated with online-based experiment.

Coverage measure: Some recommendation algorithms might provide the recommendation with high accuracy for the items where they have a huge amount of data. However, this may not be true for other items in the dataset. This is where the calculation of item coverage is vital. Item space coverage calculates the percentage of all the items in the dataset that can ever be recommended. Userspace coverage is another approach. This measures the portion of users who might never get any recommendation. The coverage here is calculated from the richness of the user's profile that is required to make a recommendation. In collaborative filtering approach, the userspace coverage could be measured as the number of items that a user needs to rate before receiving any recommendation. This measure is usually evaluated in an offline experimental setting.

Besides this, an algorithm should also be tested with cold start performance which means how the algorithm reacts with the new users. Confidence (system's trust in recommendation), trust (user's trust in recommendation), novelty (new item recommendation), serendipity (surprisingness of the successful recommendation), diversity (how diverse the recommendation is), utility (how much sales have come through the recommendation), risk in recommendation, robustness of the algorithm, privacy of users, adaptivity and scalability of the algorithm should also be measured. This all depends upon the importance of the system and the resources available [17].

2.5 Context-aware recommendation system

Most of the prevalent recommendation models focus on recommending the most relevant items to the users taking users and items as only the entities in the recommendation and do not take the contextual factors into consideration. Adomavicius et al. (2011) argue that contextual information such as time, place, company of people also play a vital role in the recommendation system and they should be taken care when modeling recommendation system.

Adomavicius et al. (2011) mention three approaches to include the context in the recommendation system: Pre-filtering, Post-filtering, and contextual model. These are illustrated in figure 5.

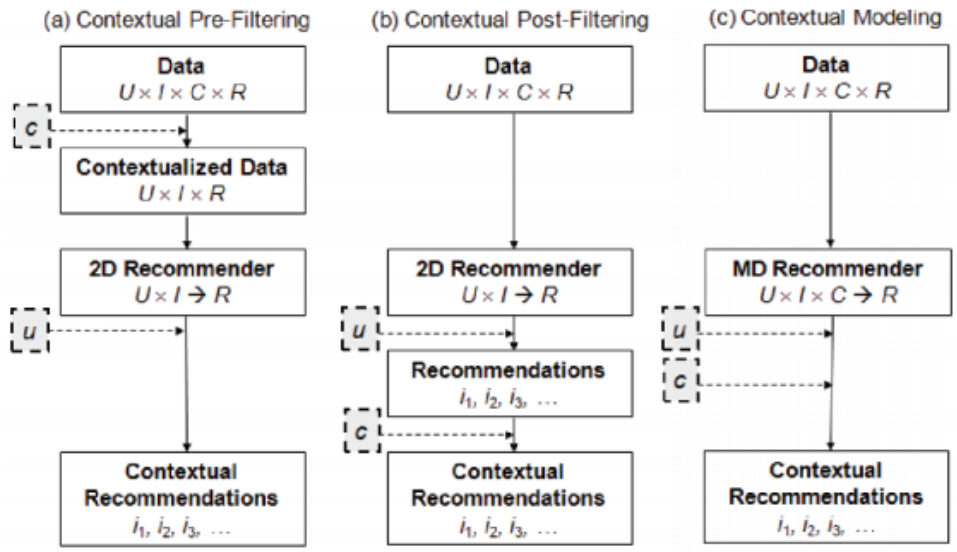

Figure 5: Context in recommendation system [18]

Pre-filtering: Contextual information is used in the dataset before modeling to select only the relevant portion of the data. For example, if a person watches only thriller movies, only the rating data of thriller movies is extracted from the dataset.

Post-filtering: In this method, contextual information is used in the final set of recommendation to constraint and re-rank. For example, there is a dataset of restaurants that has the information on restaurants all over Europe and a restaurant is to be recommended to a customer living in Finland from the dataset. If any constraints have not been set in the modeling and data preparation phase, it might be good to filter out only the Finnish restaurants from the final set of recommendations.

Contextual model: In this approach, context information is used directly in the model in learning preferences. This changes the two-dimensional data to multidimensional.

Without context:

R: User × Item → Rating

With context:

R: User × Item × Context → Rating

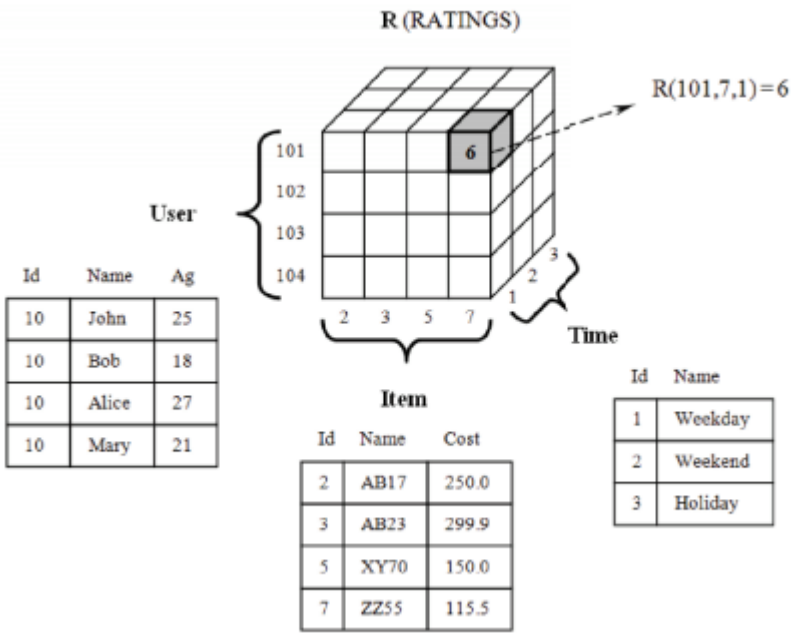

Figure 6: Multidimensional model for the User × Item × Time recommendation space[18]

Figure 6 shows a three-dimensional model when a contextual factor time is added in the model. Calculations of these models are computationally expensive. There are different factorization approaches such as factorization machines, matrix factorization and tensor factorization used in easing the calculations [18].

2.6 Recommendation system in practice

As mentioned in section 1.1, the recommendation system is getting popular and it is being implemented by many e-commerce websites, webshops, booking portals, video sharing websites and social media websites. YouTube, Netflix, eBay, Amazon is some of the leading names in the industries. This section of the thesis describes briefly how they have implemented the recommendation in their system. The current implementation might be different than the one described here because they are continuously working to improve their system. Also, it is challenging to find detailed information as the recommendation is a part of their business. However, a few of the research papers have worked on the information on those [19, 20, 21, 22].

YouTube is the largest platform for video uploading sharing and discovering at the moment of writing this thesis. It has more than a billion users. Recommending a video on YouTube is extremely challenging because of the scale, freshness, and noise. YouTube has shifted towards a deep learning approach with all the Google products. It is built on *Google Brain* which was recently introduced as an open-source by google as *TensorFlow* [19].

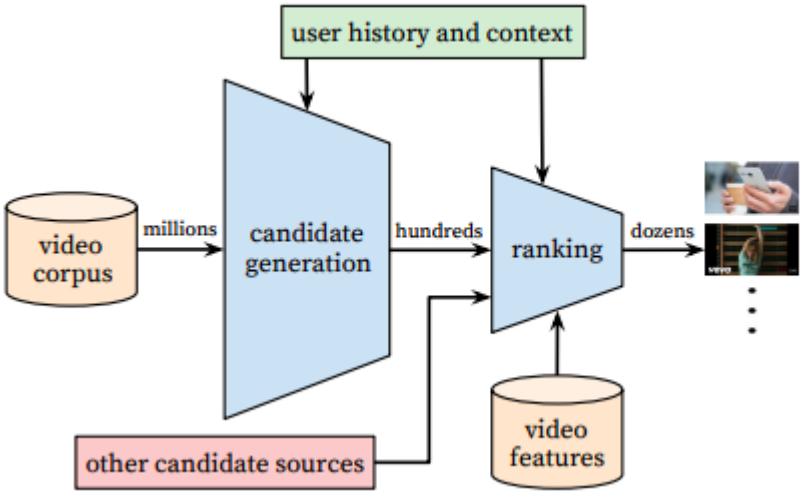

Figure 7: Recommendation system of YouTube.[19]

Figure 7 shows the recommendation system on YouTube. It consists of two neural networks, one for candidate generation and another for ranking. The candidate generation network takes input from users' behavior and history and outputs a small subset of the videos from the large corpus of YouTube videos. These generated candidates are taken as the representation for the candidate with high precision. It uses collaborative filtering to provide broad personalization [19]. The ranking network assigns a score to each video that was generated from the candidate generation algorithm using the different features that describe the user and video and serves the highest-ranking video to the user.

The recommendation on YouTube is evaluated using offline and online experimental settings. Offline metrics such as precision, recall, and ranking loss are used to guide iterative improvement to the system and the algorithm is finally evaluated using online *A/B test* with the live experiment from YouTube's system [19].

In Netflix, during the era of selling DVDs, users could rate the items using stars ranging from 1 to 5 for the items they purchase. The recommendation problem then was predicting the star to the items that the user has not purchased (or rated). Netflix organized a competition to improve those ratings [20]. Netflix has extended its business from selling DVDs to streaming the contents and

has a larger amount of data than before such as a user's demographic information, view time, and view history, that helped to build better recommendation systems.

Netflix has different algorithms to pursue the different tasks in the system. It uses the *Personalized Video Ranker* algorithm to order the entire catalog of videos for each member's profile in a personalized way. *The top-N Video Ranker* algorithm finds the best few personalized recommendations for each member of the entire catalog. *The trending Now* algorithm finds short-term and long-term trending items. *Continue Watching* orders the videos in the continuous watching row. *The video-Video Similarity* algorithm, which is an impersonalized algorithm, computes a ranked list of videos. Finally, *Page Generation* algorithm uses the outputs from the algorithms mentioned here and constructs a single page for the user [20].

The evidence algorithm is not part of the recommendation algorithm but works together with the recommendation algorithm to define the experience and help the users to determine if the video recommended to them is right or not. All those algorithms rely on statistical and machine-learning techniques that include both supervised and unsupervised learning. Supervised learning techniques include classification and regression and unsupervised learning techniques include dimensionality reduction through clustering such as topic modeling [20]. Amazon uses the item-item collaborative filtering approach which makes an automatic prediction to the users of Amazon by collecting information from several other users [22].

The eBay uses the graph-based recommendation system which is based on the user's personal taste profile describing the set of things that the user likes and the set of things that users do not like. The test profile is built on the assumption that likes and dislikes are correlated with each other [23].

2.7 Privacy issue in the recommendation system

As described in chapter 1.1, the recommendation system is dominating the user's decision making in different media screening platforms and e-commerce platforms when the recommendation is

personalized. The personalized recommendation also makes the users addictive and keeps them busy. For example, the autoplay feature of media streaming sites such as YouTube keeps on recommending new video and playing automatically. As per the Facebook founder Sean Parker, the thought process of building an application such as Facebook was to consume as much of the user's time and conscious attention as possible.

The amount of personalization of the recommendation system depends upon the accuracy of the personal data provided by the customer. More accurate personal data helps to provide a more accurate personalized recommendation to the user which is beneficial for both the users and the service providers but this raises a serious privacy and data security concern. A recommendation system undesirably and unintentionally discloses the user's personal interests and the data collected for the recommendation system can also be sold to different third parties without the user's consent. And also, if the system is compromised by external hackers, the personal data can be leaked causing serious privacy concerns [29].

To address the privacy concern, researchers have come with the concept of privacy preserved recommendation systems such as privacy-preserving collaborative filtering based recommendation and privacy-preserving content-based filtering based recommendation. Privacy-preserving collaborative filtering based recommendation is further classified into the private neighborhood-based approach and private machine-learning based approach. These methods are based either on cryptography or on a randomization technique to preserve the user's privacy [29].

The case study presented in the chapter 3 uses only the library loan log to infer rating data which is utilized to compare different recommendation system algorithms. No other personal data is used for the recommendation system. However, for the descriptive analysis of the dataset, the personal data is used but while presenting in the thesis the user id is masked.

3. Case study

3.1 Problem description

In this section, the case study performed during the writing of this thesis is presented. Different recommendation algorithms described in chapter 2 of this thesis are experimented with the dataset obtained from the Vantaa City Library which is located in Finland's metropolitan area. The data is collected systematically and the identity of the users is prevented digitally. The goal of this case study is implementing different traditional algorithms used for the recommendation system and analyze the outcome.

3.2 Consideration

Artifacts loaning behavior of a customer and the popularity of an artifact has been considered as a significant element in this case study. Practical consideration taken is limited domain knowledge available and the limited computing capacity. The dataset does not have any explicit rating or feedback given by the user to the item. Different datasets given are briefly described in the following chapter.

3.3 Datasets

Datasets provided for this case study are in the CSV (comma separated values) format. Since a non-disclosure agreement has been signed for the data, the only structure of the data and a brief exploratory analysis has been described in this chapter.

Loan data (libdata_lina.csv)

The dataset contains the loan information of the items from the library. The data is collected from the library system and it includes all the loan data of the interval 20.7.2016–22.10.2017. There are 1048575 observations and 4 features. Features include the encrypted customer ID, the ID of the borrowed artifact, borrowed date and time and the ID of the terminal from where the item was borrowed. There are 57118 unique users and 182899 unique artifacts in the dataset and a user has done maximum 1733 transactions in the period. The average user has done around 27 transactions during the period. The transaction has been done through 42 terminals. The busiest

terminal according to the dataset has done 168681 transactions and the least used terminal has done only two transactions. A terminal on average has done 36921 transactions.

Table 3: Summary of the Loan data

	Count	Mean	Std.	min	25%	50%	75%	Max
Customer	57118.00	27.15	54.35	1.00	3.00	9.00	28.00	1733.00
Artifact	182899.00	8.48	33.38	1.00	1.00	2.00	6.00	2654.00
Terminal	42.00	36921.29	44133.67	2.00	6093.00	22325.50	51569.25	168681.00

Loan return data (libdat_palautus.csv)

The dataset contains the information of the loan return of the library. This is similar to the loan data. The data is collected from the library system and it includes all the loan return data of the same interval. The dataset contains 4 features. The features are the encrypted customer ID, the ID of the returned artifact, return date and time and the ID of the terminal where the artifact was returned. There are 54092 unique users and 171200 artifacts in the dataset and a user has done maximum 2498 transactions in the period. The average user has done around 23 (~22.88) transactions during the period. The transaction is done through 58 terminals. The busiest terminal

according to the dataset has done 273707 transactions and the least used terminal has done only one transaction. A terminal on average has done 21338 transactions.

Table 4: Summary of the Loan Return data

	Count	Mean	Std.	min	25%	50%	75%	Max
Customer	54092.00	22.88	46.72	1.00	3.00	8.00	24.00	2498.00
Item	171200.00	7.23	26.61	1.00	1.00	2.00	5.00	1988.00
Terminal	58.00	21337.84	52109.32	1.00	7.25	446.00	17185.50	273707.00

Customer data (libdat_asiakas.csv / libdat_asiakas_sarnd.csv / libdat_asiakas_sakk.csv)

The dataset contains information about the 127088 customers of the library with 7 features. The features include the customer ID, the gender of the customer, the date of birth of the customer, the accumulated amount of loan of the customer before the date 31.10.2017, the current amount of loan of the customer after the date 31.10.2017, the language that customer-preferred as the service language, the postal address of customer, and the city. The number of female users is 75354 and that of the male users is 51558. 163 users did not mention their gender and 13 users mentioned their gender as others. The majority of the customers wanted to be contacted in the Finnish language, following by English and Swedish. The number of customers wanting to be contacted in Finnish, English, and Swedish was 121157, 4073 and 1855 respectively. The age of

the customer varied from 1 year to 101 years, the average being 44. The age distribution of the customer is shown in figure 8.

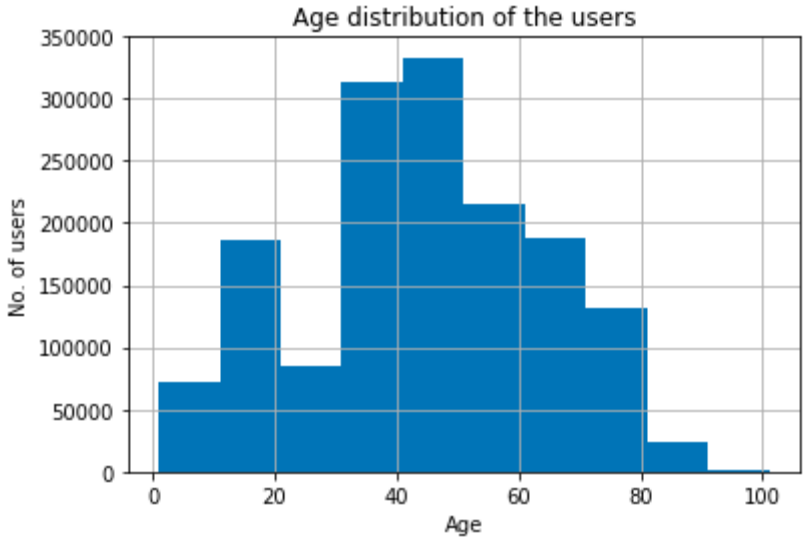

Figure 8: Age distribution of the users

Terminal data (libdat_terminaali.csv)

The dataset contains information about 46 terminals of the library with 3 features. The features include the terminal ID, name of the library and the type of terminal as automatic and service counter. There are 25 automatic and 21 service counters in the dataset.

Item data (libdat_teos.csv)

The dataset contains the information of 790653 items of the library with 9 features. The features include artifact ID, name of the author/maker, title, language ID, material ID, publication information, category ID, the appearance of the item, keywords. The number of artifacts in the Finnish language is in majority, following by English and Swedish. The number of artifacts in Finnish, English, and Swedish is 271685, 218115 and 107421 respectively.

Language data (libdat_kieli.csv)

The dataset contains the information of 485 languages with 2 features. The features include the language ID and the name of the languages.

Material data(libdat_materiaali.csv)

The dataset contains the information of 26 materials that were used in the item dataset teos.csv. It contains 2 features, the material ID and name of the material.

Category data(libdat_aiheryhma.csv)

The dataset contains the information of the 106 categories that were used in the item dataset teos.csv. It contains 2 features, the category ID and name of the categories.

3.4 Experimental setup

The case study is carried out in a MacBook with 4GB of RAM. The programming language used is python. An interactive computational environment known as Jupyter Notebook has been used to integrate scripts, texts, and visualization in a single notebook. Different libraries used include

pandas for loading and basic dataset operations, *NumPy* to work with an array, *matplotlib* for visualization *scikit-learn* for normalization of computed ratings. The algorithms are evaluated with library *Surprise,* which is a Python *scikit* for building and analyzing the recommendation system. K-fold cross-validation technique is used as a test setup which uses the whole dataset during training and testing. As an example, when the value of K is 5, then one-fifth of the data is left for testing and remaining data is used for the training. This is repeated for five times.

3.5 Preprocessing

The basic preprocessing was already completed when the dataset was received. The aim of the preprocessing done during the case study included getting a representative and unbiased subset of data that fit the computational capacity, computing the rating for the artifacts as that are not explicitly given and building user-item matrix with those ratings.

In this context, the writer of the thesis strongly believes that people of different age group consume different artifacts. To observe this behavior, users are divided into the different age group with the bin size of 10 and their consuming behavior is tabulated in table 5.

Table 5: Comparison of artifacts share by age group

Match(%)	10-20	20-30	30-40	40-50	50-60	60-70	70-80	80-90	90-100
10-20	100	46.45	61.27	66.94	49.55	33.63	22.0	6.73	0.62
20-30	40.45	100	64.53	62.68	53.1	39.75	26.07	8.95	0.89
30-40	31.09	37.61	100	65.95	49.47	36.79	24.45	8.4	0.84
40-50	28.41	30.55	55.16	100	52.41	37.3	24.83	8.56	0.81
50-60	24.58	30.25	48.36	61.26	100	47.9	30.44	11.03	1.04
60-70	21.32	28.95	45.96	55.73	61.22	100	41.22	14.84	1.38
70-80	21.65	29.47	47.42	57.59	60.39	63.98	100	23.27	2.17
80-90	21.84	33.39	53.76	65.48	72.19	75.99	76.8	100	6.58
90-100	22.41	36.72	59.23	68.95	75.43	48.49	79.39	72.82	100

From table 5, for simplicity here the explanation takes an example of no. of items in the dataset of users age between 30 to 40 and 40 to 50. There are 78665 artifacts that are borrowed by the users of the age group 40-50 and 94066 artifacts that are borrowed by the users of the age group 30-40. Users aged 30-40 also borrow 55.16% of the total artifacts borrowed by 40-50. Similarly, the age group of 40-50 borrow 65.95% of artifacts from the total artifacts borrowed by age group 40-50. The percentage keeps on decreasing either way from the common artifacts of the two age groups. For example, the common artifacts between the users of age group 30-40 and 80-90 are only 8.4%, i.e only 8.4% of the artifacts borrowed by age group 30-40 are borrowed by age group 80-90. This trend can be observed throughout the table, which indicated that setting the age of users as a parameter to select a subset of the data is safe as well as unbiased.

The next preprocessing step is to compute the ratings for the artifacts. Initially, the dataset only contained the artifact loan log and return log. There is information that a user x has loaned artifact y, which does not necessarily indicate that the user x liked artifact y. Likewise, the artifacts that are not loaned by user x does not necessarily indicate that the user x does not like those artifacts. Since it is the only user's behavior available in the dataset, a numerical rating is computed using the method mentioned below.

Firstly, the dataset is converted into binary rating 0 meaning that the user has not loaned the artifact and 1 meaning that the user has loaned the artifact as in table 6.

Table 6: Binary rating table

User	Artifact	Rating
1024945XXX	b22474XXX	1
1024945XXX	b22475XXX	0
1024945XXX	b22476XXX	1

In table 6, there are two artifacts b22474XXX and b22476XXX loaned by the user 1024945XXX and the item b22475XXX is not yet loaned (last three characters of the users and artifacts are masked). This assigns the same value 1 to all loaned artifacts and 0 to all the artifacts that are not loaned. Among the loaned items, suppose b22474XXX is a very popular artifact in the list being loaned for the highest amount of time and b22476XXX is the least famous artifact in the list being loaned for only a few times, assigning them equal rating gives equal weight to them. To solve this problem, a computed rating ranging from 1 to 10 is generated using a normalizer. The normalizer used transforms the rating by scaling each rating to the given range 1 to 10.

For the normalizing purpose, a simple technique is used called *minMaxScaler* which transforms the rating by scaling each rating to the given range. Each rating value is subtracted with the

minimum value of ratings and divided by the range of ratings. Range of ratings is the difference between maximum and minimum rating. The formula to calculate the *minMaxScaler* is:

$$RatingNormalized = \frac{Rating - RatingMin}{RatingMax - RatingMin} \times (FeatureMax - FeatureMin) + FeatureMin.$$

Where the *RatingNormalized* is the normalized rating, the Rating is the original rating, the *RatingMin* is the minimum rating from the entire rating list, the *RatingMax* is the maximum rating from the entire list. The *FeatureMin* and the *FeatureMax* are the lower and upper limit in which we want to normalize the ratings, 1 and 10 in this case. The *minMaxScaler* preserves the orientation of the data [28].

If an artifact is very popular and the user has loaned it, the user might give the highest rating 10 to the artifact. With this new rule, b22474XXX is rated the highest rating 10 and the artifact b22476XXX is rated the lowest rating 1 and the 1024945XXX does not get any rating as it is not yet lent by the user 1024945XXX. The problem of recommendation as described in chapter 2.1 is to predict the rating for this item.

Table 7: Computed rating table

User	Artifact	Rating
1024945XXX	b22474XXX	10
1024945XXX	b22475XXX	NA
1024945XXX	b22476XXX	1

The computed rating distribution is illustrated in figure 9.

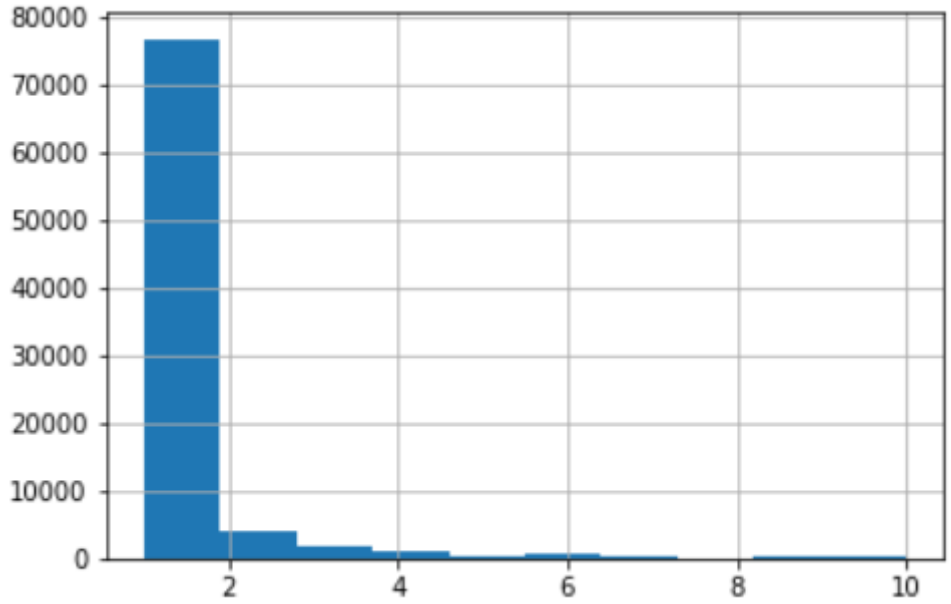

Figure 9: Rating distribution

The next step of the preprocessing is to convert the table of the user, artifacts, and rating as given in table 7 to user-item matrix. A pivot table is constructed as in table 8 filling the cells of the table with the respective rating data.

Table 8:

User/Item	b22474XXX	b22475XXX	b22476XXX
1024945XXX	10	NA	1
1024944XXX	.....	.....	.....
1024943XXX	.....	.....	.....
1024942XXX	.....	.....	.....

3.5 Recommendation Algorithms

The dataset is trained with different algorithms for prediction based on collaborative filtering and matrix factorization and evaluated using k-fold cross-validation(k=5). The metrics for validation used are the Root Mean Squared Error(RMSE) and the Mean Squared Error. Firstly, the chunk of the dataset where the age of the users is between 40-50 is taken and experimented. The results obtained are listed below.

NormalPredictor

This algorithm predicts the random rating to the artifacts. This is a basic algorithm used to compare the accuracy with other algorithms. The predicted rating is calculated from the normal distribution. The parameters of the normal distribution are estimated using *Maximum Likelihood Estimation*. The result obtained with this algorithm is tabulated in table 9.

Table 9: 5-fold cross-validation with the NormalPredictor

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	1.4827	1.4742	1.5016	1.4894	1.4563	1.4809	0.0152
MAE (testset)	0.9970	0.9935	0.9934	0.9974	0.9832	0.9929	0.0051
Fit time	0.12	0.15	0.15	0.15	0.17	0.15	0.02
Test time	0.19	0.17	0.17	0.25	0.27	0.21	0.04

BaselineOnly

This is another basic algorithm used to compare the accuracy. This is based on the average rating from the entire ratings from the database. This is calculated as follows:

$$\hat{r}_{ui} = b_{ui} = \mu + b_u + b_i$$

Where $\hat{r}_{ui}$ is predicted rating, μ is average rating, b_u is the *bias* in a user and b_i is the *bias* in an item. For the known user bias b_u is 0 and for the known item bias b_i is 0 [24]. The result obtained with this algorithm is tabulated in table 10. The mean RMSE with this algorithm is 0.37 and the mean MAE error is 0.29.

Table 10: 5-fold cross-validation with the BaselineOnly

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.3629	0.3662	0.3679	0.3653	0.3675	0.3660	0.0018
MAE (testset)	0.2858	0.2869	0.2871	0.2872	0.2874	0.2869	0.0006
Fit time	0.41	0.48	0.40	0.46	0.49	0.45	0.03
Test time	0.14	0.14	0.14	0.18	0.16	0.15	0.02

SVD

The Singular Value Decomposition (SVD) algorithm has been popular in the field of

recommendation system since the Netflix prize. The algorithm falls under the Matrix factorization. The prediction rating $\hat{r}_{ui}$ is defined as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u$$

μ = average rating

b_u = bias in user

b_i = bias in item

$q_i^T p_u$ = interaction between user u and item i showing overall interest in the item's characteristic. [25]

When the baseline ($\mu + b_u + b_i$) are not used, this is equivalent to the Probabilistic Matrix Factorization which is not experimented in this case study. If a user u is unknown, the bias b_u and factor p_u are assumed to be 0 and when an item *i* is unknown, the bias b_i and factor q_i are assumed to be 0. For this implementation of the SVD baselines are initialized to 0 and the learning rates are set to 0.005 and the regularization terms are set to 0.02.

The result obtained with this algorithm is tabulated in table 11. The mean RMSE with this algorithm is 0.29 and the mean MAE error is 0.23.

Table 11: 5-fold cross-validation with the SVD

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.2931	0.3036	0.2967	0.2957	0.2895	0.2957	0.0047
MAE (testset)	0.2278	0.2292	0.2271	0.2264	0.2264	0.2274	0.0010
Fit time	7.53	8.82	6.75	7.79	8.21	7.82	0.69
Test time	0.25	0.23	0.53	0.25	0.50	0.35	0.13

SVDpp

The SVDpp (SVD++) is a matrix factorization based algorithm which is an extension of the SVD algorithm. This algorithm also takes implicit ratings into account. Implicit ratings in this algorithm take account of the fact that a user has rated an item regardless of the rating value. The prediction rating $\hat{r}_{ui}$ as per this algorithm is defined as:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T (p_u + |I_u|^{-1/2} \sum_{j \in I_u} y_j$$

y_j is a new set of item factors capturing the implicit ratings. Other terms used in the equations are similar to SVD explained above [25]. The baseline for the SVDpp in this implementation is initialized to 0 and the learning rates are set to 0.005 and the regularization terms are set to 0.02.

The result obtained with this algorithm is tabulated in table 12. The mean RMSE with this algorithm is 0.23 and the mean MAE error is 0.17.

Table 12: 5-fold cross-validation with the SVD++

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.2282	0.2375	0.2266	0.2265	0.2274	0.2293	0.0042
MAE (testset)	0.1667	0.1664	0.1679	0.1678	0.1670	0.1672	0.0006
Fit time	242.59	224.58	206.55	200.92	199.99	214.93	16.42
Test time	3.37	4.95	3.10	3.03	2.96	3.48	0.75

NMF

The Non-negative Matrix Factorization (NMF) is a collaborative filtering algorithm based on non-negative matrix factorization which is also very similar to the SVD. The prediction rating $\hat{r}_{ui}$ according to NMF is defined as:

$$\hat{r}_{ui} = q_i^T p_u$$

Both the user factor P_u and the item factor q_i should be positive [26]. The result obtained with this algorithm is tabulated in table 13. The mean RMSE with this algorithm is 0.31 and the mean MAE error is 0.20.

Table 13: 5-fold cross-validation with the NMF

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.3138	0.3149	0.3188	0.3107	0.3159	0.3148	0.0026
MAE (testset)	0.2032	0.2035	0.2075	0.2024	0.2055	0.2044	0.0019
Fit time	9.82	9.83	9.69	9.87	9.77	9.79	0.06
Test time	0.17	0.15	0.15	0.16	0.15	0.16	0.01

KNNBasic

The KNNBasic is a basic nearest-neighbor based collaborative filtering algorithm derived from the basic nearest-neighbor approach. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u, v).r_{vi}}{\sum_{v \in N_i^k(u)} sim(u, v)}$$

This is a user-based collaborative filtering with the cosine similarity as the similarity metric. $N_i^k(u)$ are the k nearest neighbors of user u that have rated item *v*. This set is computed using cosine similarity metric. The result obtained with this algorithm is tabulated in table 14. The mean RMSE with this algorithm is 0.36 and the mean MAE error is 0.20.

Table 14: 5-fold cross-validation with the KNNBasic

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.3568	0.3565	0.3503	0.3634	0.3649	0.3584	0.0053
MAE (testset)	0.2063	0.2058	0.2055	0.2100	0.2085	0.2072	0.0018
Fit time	0.17	0.27	0.19	0.19	0.20	0.20	0.03
Test time	0.44	0.42	0.41	0.42	0.67	0.47	0.10

KNNWithMeans

The KNNWithMeans is also a basic nearest-neighbor based collaborative filtering algorithm which takes mean rating of each user into account. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = \frac{\sum_{v \in N_i^k(u)} sim(u, v).(r_{vi} - \mu_v)}{\sum_{v \in N_i^k(u)} sim(u, v)}$$

The result obtained with this algorithm is tabulated in table 15. The mean RMSE with this algorithm is 0.50 and the mean MAE error is 0.37.

Table 15: 5-fold cross-validation with the KNNWithMeans

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.5054	0.5173	0.4922	0.4978	0.4924	0.5010	0.0094
MAE (testset)	0.3733	0.3745	0.3668	0.3672	0.3684	0.3700	0.0032
Fit time	0.29	0.34	0.35	0.25	0.25	0.30	0.04
Test time	0.45	0.68	0.43	0.43	0.43	0.49	0.10

KNNWithZScore

The KNNWithZScore is also a basic nearest-neighbor based collaborative filtering algorithm which takes the z-score normalization of each user into account. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = \mu_{\mu} + \sigma_u \frac{\sum_{v \in N_i^k(u)} sim(u,v).(r_{vi} - \mu_v)/\sigma_v}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

σ_u = standard deviation of all ratings given by user u.

σ_v = the standard deviation of all ratings given to item v.

The result obtained with this algorithm is tabulated in table 16. The mean RMSE with this algorithm is 0.67 and the mean MAE error is 0.43.

Table 16: 5-fold cross-validation with the KNNWithZScore

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.6889	0.6726	0.6738	0.6714	0.6653	0.6744	0.0078
MAE (testset)	0.4417	0.4349	0.4326	0.4290	0.4319	0.4340	0.0043
Fit time	0.34	0.48	0.40	0.39	0.37	0.40	0.05
Test time	0.50	0.53	0.46	0.47	0.46	0.49	0.03

KNNBaseline

The KNNBaseline is also a basic nearest-neighbor based collaborative filtering algorithm which takes mean rating of each user into account. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = b_{ui} + \frac{\sum_{v \in N_i^k(u)} sim(u,v).(r_{vi} - b_{vi})}{\sum_{v \in N_i^k(u)} sim(u,v)}$$

Here, b_{ui} is the baseline rating by user u to item i. The result obtained with this algorithm is tabulated in table 17. The mean RMSE with this algorithm is 0.24 and the mean MAE error is 0.17.

Table 17: 5-fold cross-validation with the KNNBaseline

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.2392	0.2419	0.2367	0.2352	0.2358	0.2378	0.0025
MAE (testset)	0.1759	0.1758	0.1764	0.1749	0.1754	0.1757	0.0005
Fit time	0.69	0.64	0.59	0.61	0.55	0.62	0.05
Test time	0.98	0.48	0.67	0.48	0.49	0.62	0.19

SlopeOne

This is a simple implementation of the rating-based collaborative filtering. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = \mu_{\mu} + \frac{1}{|R_i(u)|} \sum_{j \in R_i(u)} dev(i,j)$$

The *R(u)* is the set of items *j* rated by user u that has at least one common user with *i*. The *dev(i,j)* is the average difference between ratings of *i* and ratings of *j*[27]. The result obtained with this algorithm is tabulated in table 18. The mean RMSE with this algorithm is 0.59 and the mean MAE error is 0.45.

Table 18: 5- fold cross-validation with the SlopeOne

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.5873	0.5831	0.5878	0.5971	0.5891	0.5888	0.0046
MAE (testset)	0.4545	0.4546	0.4524	0.4578	0.4540	0.4547	0.0017
Fit time	113.01	2578.26	153.34	142.64	130.94	623.64	977.40
Test time	8.54	60.13	27.72	16.64	12.34	25.07	18.67

CoClustering

This is an implementation of collaborating filtering algorithm based on co-clustering [28]. In this algorithm, users and items are classified into different clusters and co-clusters. The prediction rating $\hat{r}_{ui}$ according to this algorithm is defined as:

$$\hat{r}_{ui} = \overline{C_{ui}} + \left(\mu_{\mu} - \overline{C_u}\right) + \left(\mu_{\mu} - \overline{C_i}\right)$$

$\overline{C_{ui}}$ = Average rating of co-cluster C_{ui}

$\overline{C_u}$ = Average rating of cluster C_u

$\overline{C_i}$= Average rating of cluster C_i

The Clusters and the co-clusters are assigned using optimization methods such as k-means clustering. The result obtained with this algorithm is tabulated in table 19. The mean RMSE with this algorithm is 0.53 and the mean MAE error is 0.42.

Table 19: 5-fold cross validation with the CoClustering

	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5	Mean	Std
RMSE (testset)	0.5099	0.5152	0.5257	0.5294	0.5711	0.5303	0.0216
MAE (testset)	0.4063	0.4157	0.4180	0.4213	0.4431	0.4209	0.0122
Fit time	7.11	7.16	6.85	6.51	7.01	6.93	0.23
Test time	0.21	0.18	0.93	0.15	0.20	0.33	0.30

4. Result

In this section of the thesis, the result obtained with the portion of the dataset containing customer of age between 40-50 is visualized to make the comparison between different recommendations easy. The detailed result obtained and the scripts used during the case study is uploaded in GitHub and the link is provided in the appendix.

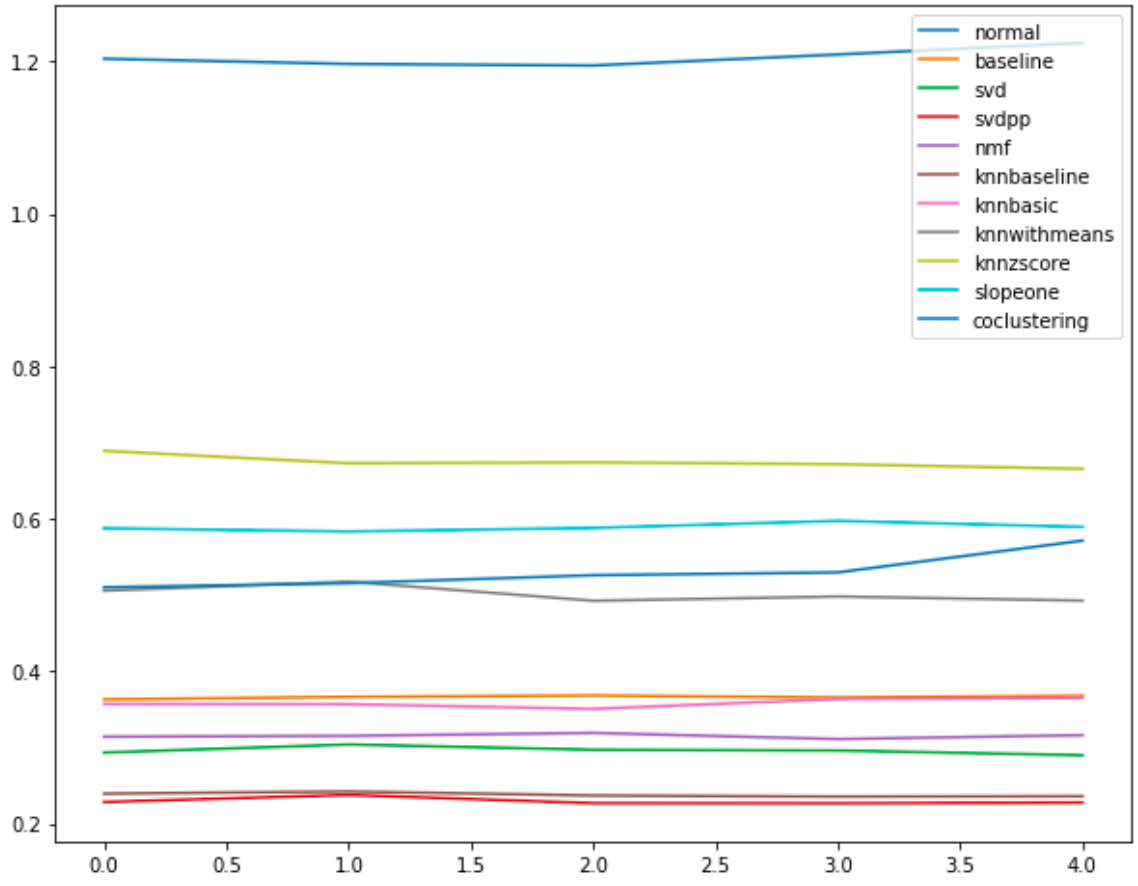

Figure 10: Comparison of the RMSE

Figure 10 is the comparison of the Root Mean Square Error (RMSE) of the different recommendation algorithms used. The result shows that SVD++ gives the least RMSE followed by the KNNBaseline and the SVD. All the recommendations used clearly beat the normal recommendation which recommends random artifacts to the user based on the rating distribution.

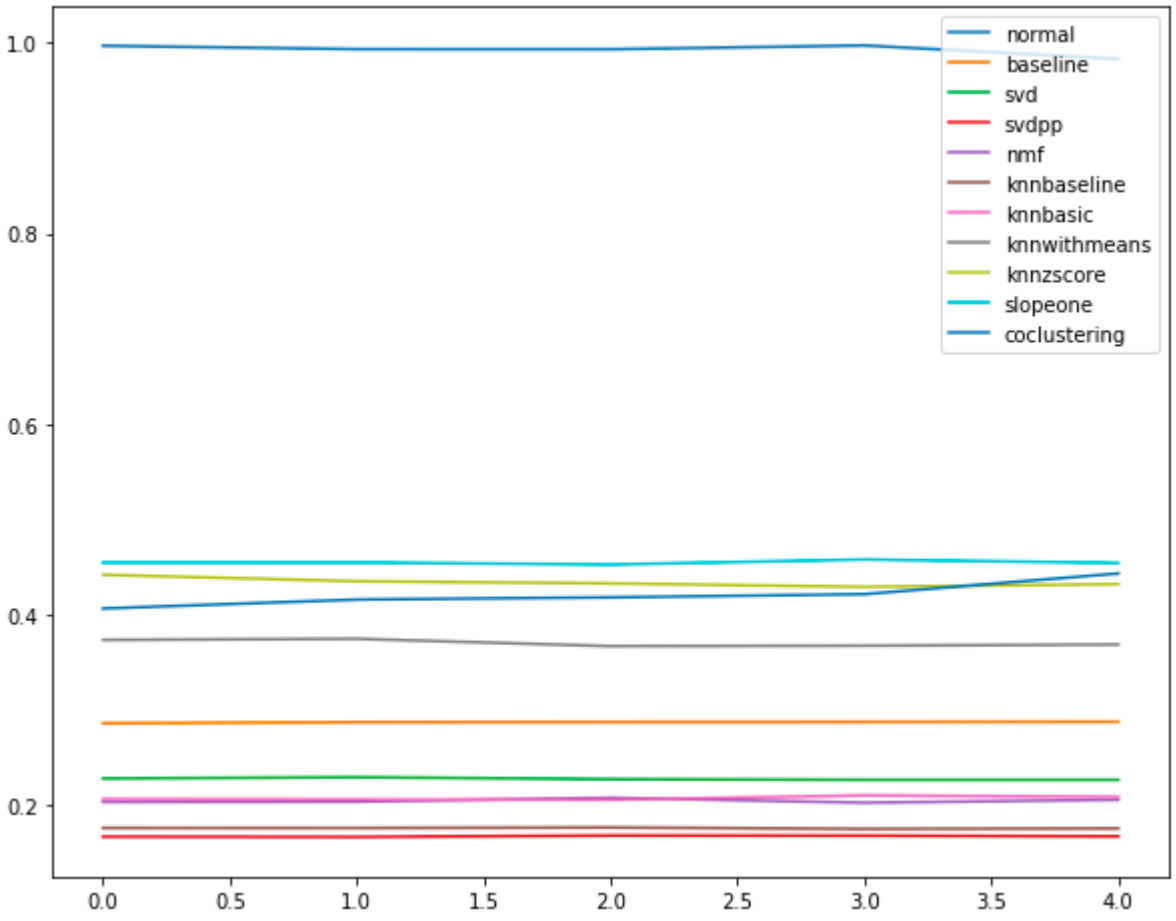

Figure 11: Comparison of the MAE

The recommendation algorithms used are also evaluated using another validation metric Mean Absolute Error(MAE) and the result is visualized in fig 10. The SVD++ beats all the recommendation algorithms giving the least MAE. Which is followed by the KNNBaseline. With this evaluation metric, KNNBasic and NMF beat SVD.

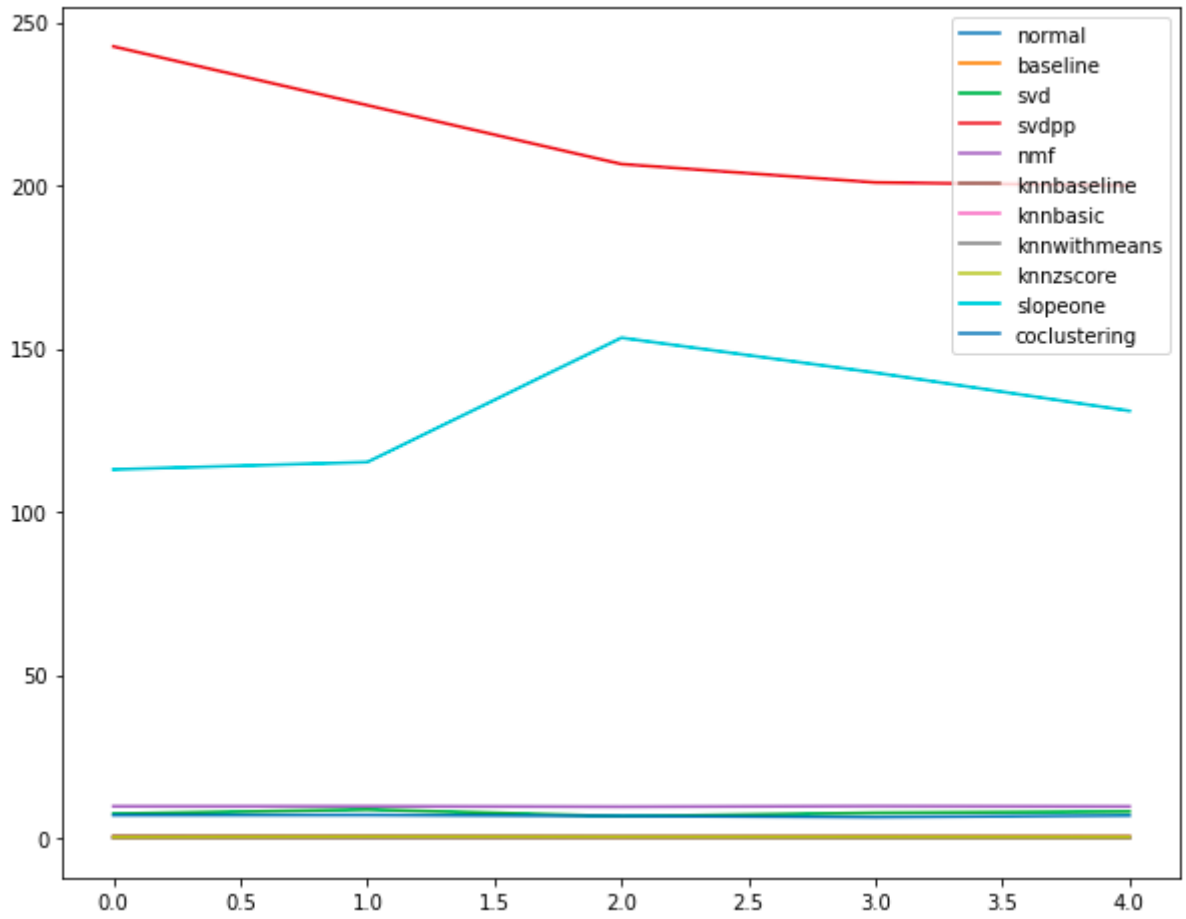

Figure 12: Comparison of the Training time

Figure 12 is the comparison of the training time. Training the recommendation algorithm SVD++ takes the highest amount of time followed by the SlopeOne and the NMF.

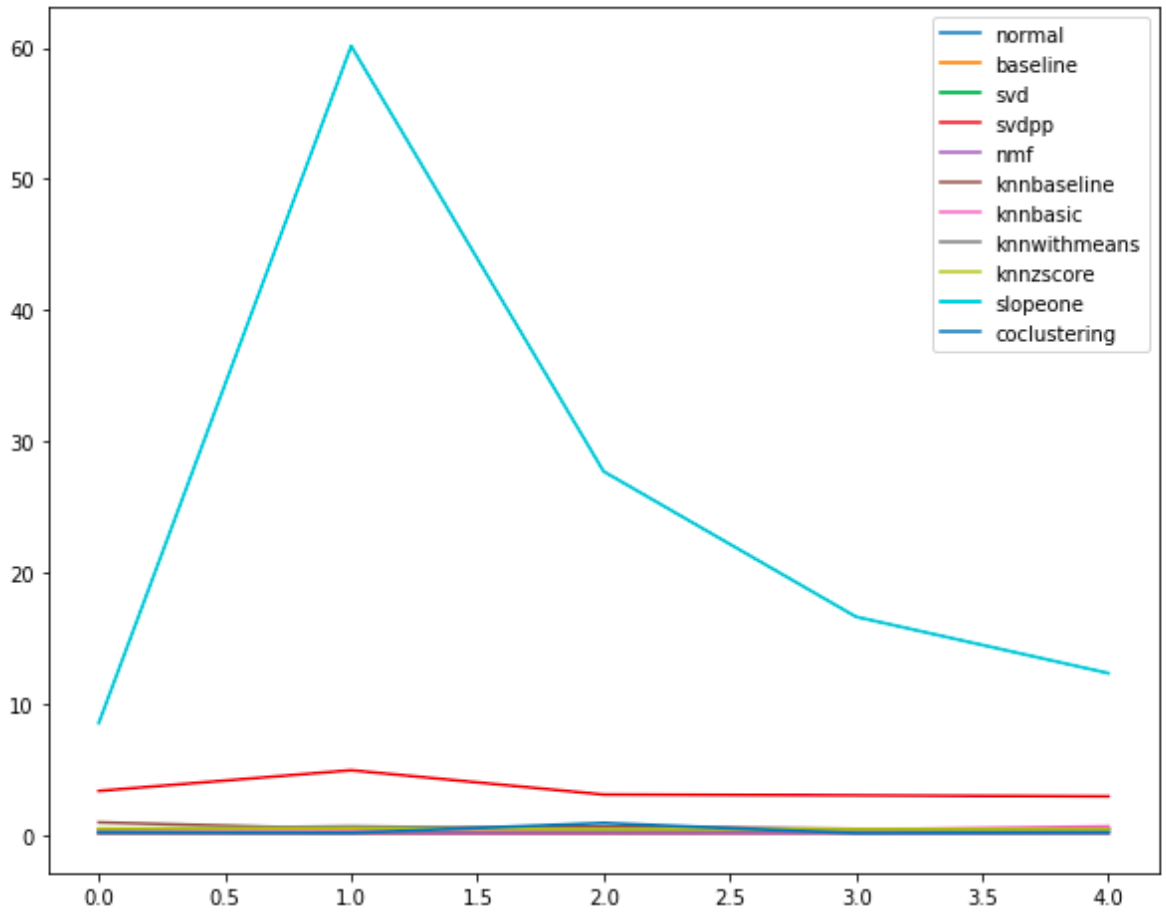

Figure 13: Comparison of the Test time

Figure 13 is the comparison of the validation time of different recommendation algorithm using 5-fold cross-validation. Validating the SlopeOne takes the highest amount of time followed by the SVD++.

5. Discussion and Conclusion

The goal of this thesis was to compare algorithms for the recommendation system on library loan data using offline data of artifacts log obtained through different libraries of a city. For this purpose, different recommendation algorithms are trained and evaluated using the evaluation metrics. Since the recommendation system built uses offline data and the machine used to train and validate the algorithm has very limited computing capacity, the time taken to train and

validate the algorithms can be ignored. In a production environment, the recommendation system is considered as a big data problem and computation is done using the cloud computing in the time of writing this thesis. With the 5-fold cross-validation, the algorithm is trained and validated for the five times which is also a reason for higher training and testing time. After ignoring the time factor, the improved version of SVD (SVD++) is performing better than other algorithms compared. The Nearest-neighbor algorithms try to recommend the artifacts that are closer to the artifacts previously consumed by the user. In the matrix factorization, the cross product of the latent factors of the users and the artifacts is the rating prediction.

A recommendation system needs to address the cold start problem. The case study performed does not address this problem. The case study is carried out with a portion of the data. The error metrics go higher when the size of the portion is increased. Therefore, the values of error metrics are not the standard one and should be taken care while comparing with other algorithms.

REFERENCES:

1. Amatriain, X., 2014, October. The recommender problem revisited. In *Proceedings of the 8th ACM Conference on Recommender systems* (pp. 397-398). ACM.Celma Ò. (2010) The Recommendation Problem. In: Music Recommendation and Discovery. Springer, Berlin, Heidelberg
2. Deza M.M., Deza E. (2014) Distances and Similarities in Data Analysis. In: Encyclopedia of Distances. Springer, Berlin, Heidelberg
3. Spertus E., Sahami M., Buyukkokten O., Evaluating similarity measures: a large-scale study in the Orkut social network, Proceeding: KDD '05 Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, Pages 678-684 Chicago, Illinois, USA, August 21 - 24, 2005
4. Amatriain X., Jaimes* A., Oliver N., Pujol J.M. (2011) Data Mining Methods for Recommender Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA
5. Burke, R., Mobasher, B., Bhaumik, R.: Limited knowledge shilling attacks in collaborative filtering systems. In Proceedings of Workshop on Intelligent Techniques for Web Personalization (ITWP'05) (2005)
6. D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," Communications of ACM, vol. 35, no. 12, pp. 61–70, 1992.
7. K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," Information Retrieval, vol. 4, no. 2, pp. 133–151, 2001.
8. 3. B. N. Miller, J. A. Konstan, and J. Riedl, "PocketLens: toward a personal recommender system," ACM Transactions on Information Systems, vol. 22, no. 3, pp. 437–476, 2004.
9. 4. G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," IEEE Internet Computing, vol. 7, no. 1, pp. 76–80, 2003.
10. 5. J. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI '98), 1998.
11. Koren Y., Bell R. (2015) Advances in Collaborative Filtering. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA
12. de Gemmis M., Lops P., Musto C., Narducci F., Semeraro G. (2015) Semantics-Aware Content-Based Recommender Systems. In: Ricci F., Rokach L., Shapira B. (eds) Recommender Systems Handbook. Springer, Boston, MA
13. Pilászy I., Tikk D., Recommending new movies: even a few ratings are more valuable than metadata, Proceeding: RecSys '09 Proceedings of the third ACM conference on Recommender systems Pages 93-100
14. Zhang Y., Yuan B., Tang K. (2017) Enhanced Pairwise Learning for Personalized Ranking from Implicit Feedback. In: He C., Mo H., Pan L., Zhao Y. (eds) Bio-inspired Computing: Theories and Applications. BIC-TA 2017. Communications in Computer and Information Science, vol 791. Springer, Singapore
15. Kohavi, R., Longbotham, R., Sommerfield, D., Henne, R.M.: Controlled experiments on the web: survey and practical guide. Data Min. Knowl. Discov. 18(1), 140–181 (2009). DOI http://dx.doi.org/10.1007/s10618-008-0114-1
16. Shani G., Gunawardana A. (2011) Evaluating Recommendation Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA

17. Adomavicius G., Tuzhilin A. (2011) Context-Aware Recommender Systems. In: Ricci F., Rokach L., Shapira B., Kantor P. (eds) Recommender Systems Handbook. Springer, Boston, MA
18. Covington, P., Adams, J. and Sargin, E., 2016, September. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198). ACM.
19. Carlos A. Gomez-Uribe and Neil Hunt. 2015. The Netflix recommender system: Algorithms, business value, and innovation. ACM Trans. Manage. Inf. Syst. 6, 4, Article 13 (December 2015), 19 pages. DOI: http://dx.doi.org/10.1145/2843948
20. Netflix Prize 2009, https://www.netflixprize.com/ [Accessed 7 2 2018)
21. Smith, B. and Linden, G., 2017. Two decades of recommender systems at Amazon. com. IEEE Internet Computing, 21(3), pp.12-18.
22. Pinckney, T., 2013a. NYC* 2013 - "Graph-based Recommendation Systems at eBay" by Thomas Pinckney. [Online] Available at: https://www.youtube.com/watch?t=2400&v=Tg3dP2fZGSM [Accessed 7 2 2018]
23. Koren, Y. 2010. Factor in the neighbors: Scalable and accurate collaborative filtering. ACM Trans. Knowl. Discov. Data. 4, 1, Article 1 (January 2010), 24 pages. DOI = 10.1145/1644873.1644874 http://doi.acm.org/10.1145/1644873.1644874
24. Y. Koren, R. Bell, and C. Volinsky, "Matrix Factorization Techniques for Recommender Systems," in Computer, vol. 42, no. 8, pp. 30-37, Aug. 2009. doi: 10.1109/MC.2009.263. URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5197422&isnumber=5197410
25. X. Luo, M. Zhou, Y. Xia and Q. Zhu, "An Efficient Non-Negative Matrix-Factorization-Based Approach to Collaborative Filtering for Recommender Systems," in IEEE Transactions on Industrial Informatics, vol. 10, no. 2, pp. 1273-1284, May 2014. doi: 10.1109/TII.2014.2308433 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6748996&isnumber=6809862
26. Lemire, Daniel & Maclachlan, Anna. (2007). Slope One Predictors for Online Rating-Based Collaborative Filtering. Proceedings of the 2005 SIAM International Conference on Data Mining, SDM 2005. 5. 10.1137/1.9781611972757.43.
27. T. George and S. Merugu, "A scalable collaborative filtering framework based on co-clustering," Fifth IEEE International Conference on Data Mining (ICDM'05), Houston, TX, 2005, pp. 4 pp.-. doi: 10.1109/ICDM.2005.14 URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1565742&isnumber=33217
28. Scikit-learn.org. (2019). sklearn.preprocessing.MinMaxScaler — scikit-learn 0.21.2 documentation. [online] Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html#sklearn.preprocessing.MinMaxScaler [Accessed 14 Jul. 2019].
29. D. Li, Q. Lv, L. Shang, N. Gu Efficient privacy-preserving content recommendation for online social communities Neurocomputing, 219 (2017), pp. 440-454

Appendix

https://github.com/usatyal/Recommender-system-case-study-/blob/master/Case_study.ipynb